SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 1–15, January 1994

ERROR BOUNDS FOR DYNAMIC RESPONSES IN FORCED VIBRATION PROBLEMS*

CHRISTIAN CABOS†

Abstract. When using mode superposition in large applications, generally only relatively few approximate eigenmodes are linearly combined. Block Lanczos iteration is an efficient method of determining such modes. In this paper new a posteriori bounds are developed that estimate the error when approximating the exact result of mode superposition with a linear combination of the output vectors of block Lanczos iteration. Mode superposition can be regarded as a way of computing $g(S)f$, a function g of a selfadjoint matrix S applied to a vector. One formula is developed that estimates the norm of the unknown error vector. A second inequality gives a bound for the error when computing linear functionals $\langle v, g(S)f\rangle$ of the response. The error bounds require that f and possibly v are contained in the Lanczos starting block and that all Ritz vectors are used to compute the result. No gaps in the spectrum of S need to be known. The bounds can be evaluated at a small cost compared to the eigenpair extraction in large systems. In a forced response calculation for a container ship with $\approx$ 38,000 degrees of freedom the error is overestimated by two to four orders of magnitude.

Key words. mode superposition, error bounds, Lanczos method, matrix functions, forced vibrations

AMS subject classifications. 65F30, 65L70, 73K12, 70J35, 65F50, 65F15

1. Introduction.

Certain systems of linear ordinary differential equations with constant coefficients can be decoupled by the method of mode superposition. The equations of motion for a linear system in structural dynamics are one example where the method has proven particularly useful when the system is large. The reason is that often only few eigenmodes need to be superposed to approximate the response to a dynamic load. The block Lanczos algorithm has been applied in this context [9] because it can efficiently determine few modes near some specified frequency. In this paper a posteriori bounds are developed that estimate the error when approximating the exact result of mode superposition with a linear combination of the output vectors of block Lanczos iteration. The term eigenmodes will be used as a synonym for eigenvectors.

Suppose that mode superposition is carried out with the input vector f representing an external load. Then since the exact calculation consists in amplifying each eigenvector component of f by some function g of the respective eigenvalue λ_i, the whole process can be regarded as computing $u = g(S)f$, a matrix function applied to a vector. S is the selfadjoint system matrix with eigenvalues λ_i, $i = 1, \ldots, n$. The exact vector u, called *response* in the following, will be approximated by the vector $\tilde{u}$ obtained by superposing few approximate eigenmodes.

Setting $g(x) = 1/x$, for example, $u = S^{-1}f$ is computed. In [15], among other functions, $S^{-1}f$ and $\exp(S)f$ are approximated in this way. In [4] the functions $\exp(-tS)f$, $\exp(-tS^{0.5})f$, and $\cos(tS^{0.5})f$ arising in partial differential equations are examined and a priori bounds are given for these functions. The number t is a parameter. An expansion for the error vector when approximating $\exp(S)f$ is derived in [12] and a posteriori norm estimates are found by truncating this expansion. All these articles refer to iterations with blocksize one.

The function $g(x) = \frac{1}{\alpha+\gamma x}$ with complex α and γ leads to the evaluation of the modally damped response to a harmonic load in structural dynamics. A common procedure in the field of forced vibrations consists in superposing only those modes that have been obtained with a given accuracy. In 1982, Wilson, Yuan, and Dickens [16] suggested not using only

*Received by the editors April 27, 1992; accepted for publication (in revised form) December 31, 1992. This work was partially supported by Bundesministerium für Forschung und Technologie.

†Germanischer Lloyd, P.O.B. 111606, D-20416 Hamburg, Germany.

these "exact" eigenvectors of the structural system. Rather, they superposed all so-called Ritz vectors generated by the Lanczos algorithm with starting vector f, where f represents the static displacement due to the spatial distribution of the dynamic load. Numerical experiments showed that the error in calculating shear and moments at specific points was significantly smaller when using Ritz vectors instead of the same number of exact eigenvectors. Application of this idea to the dynamic analysis of large structures (more than 10^5 degrees of freedom) showed a factor 2–3 improvement in throughput time [1]. In [9] the idea is extended to the block Lanczos method. The error bounds given below rely on the fact that all Ritz vectors are superposed.

Since the determination of eigenmodes is very expensive when the systems are large, error bounds can provide such useful information as when to stop the Lanczos iteration. As stated in [2], in the context of forced vibrations there is no justifiable error estimate showing how many Lanczos vectors are needed to approximate the response to a given accuracy. The stopping criterion used in [8] and [9] is based on participation factors $E_k^\top F$ of the dynamic load $Fe^{i\omega t}$. Here E_k is the kth Lanczos block obtained and ω is the circular frequency of the dynamic load. The Lanczos iteration is stopped after step r if the norm of $E_{r+1}^\top F$ falls below a specified limit. However, the relation between the participation factors and the error in the response is not clear [2]. In [7] the participation of the last Lanczos vector in the approximate solution of heat conduction problems is used for an error criterion.

This paper is organized as follows. After introducing notation, a formula for the norm of the error in the calculated response is given. This is a measure for the (mean square) overall accuracy of the result. However, in engineering applications a common task is the evaluation of the response at a specific degree of freedom or a linear combination thereof (e.g., element stresses). This can be described as calculating linear functionals of the response. A second bound is developed for estimating the error in evaluating such linear functionals of the response.

An important application is the computation of forced vibrations in structural dynamics. Evaluation of the error bounds can be simplified in case of Rayleigh's damping. As a numerical example the forced vibrations of a container ship are examined with respect to their accuracy. The finite element model used contains approximately 38,000 degrees of freedom.

2. Overview of block Lanczos method. To introduce some notation, the basic equations for the block Lanczos method will be given. For a more detailed description of this algorithm see, e.g., [5] and the references therein. The scalar product used below may be of generalized form $\langle a, b\rangle := a^*Mb$ where M is a Hermitian positive definite matrix and * denotes the complex conjugate transpose. $\langle ., .\rangle$ will also be used if a and b have more than one column. In this case $\langle a, b\rangle$ is a matrix. The norm induced will be called $\|a\|_M^2 := \langle a, a\rangle$ to distinguish it from the euclidean norm $\|b\|$. Orthonormality and selfadjointness always refer to this generalized scalar product if vectors or systems of dimension n are involved.

Assume that E_1 is the $n \times b$ orthonormal *starting block* for the Lanczos algorithm with selfadjoint $n \times n$ iteration matrix S. The integer b is called the *blocksize*. By a combination of iteration and orthogonalization the Lanczos method generates blocks of vectors $E_1, \ldots, E_r$. These are used as input vectors for the Ritz method. The Lanczos algorithm enforces the following recursion among successive blocks:

$$E_{r+1}B_{r+1} = SE_r - E_rA_r - E_{r-1}B_r^\top. \tag{1}$$

Here r denotes the total number of Lanczos steps so far and $E = (E_1, E_2, \ldots, E_r)$ is the matrix of the first $m = b \cdot r$ Lanczos vectors. Typical values of n, m, and b used in this paper are $n = 40{,}000$, $m = 200$, $b = 10$. The columns of E are mutually orthonormal and span the block Krylov space $\mathcal{E} = \text{span}(E)$. The $b \times b$ block coefficients A_i and B_i

can be assembled to form the symmetric block tridiagonal matrix

$$T = \langle E, SE \rangle. \tag{2}$$

The A_i are the diagonal blocks; $B_i^\top$ and B_i are on the sub- and superdiagonal, respectively. An eigendecomposition $TQ = Q\tilde{\Lambda}$ of T with $\tilde{\Lambda} = \operatorname{diag}(\tilde{\lambda}_i, i = 1, \ldots, m)$ produces *Ritz values* $\tilde{\lambda}_i$, which approximate the exact eigenvalues λ_j, $j = 1, \ldots, n$ of S. The *Ritz vectors* $\tilde{X} = EQ$ are approximations of the exact eigenvectors X. They also span $\mathcal{E}$.

It is of particular importance for the error bounds given below that in the absence of roundoff errors one has access to the exact size of the residual vector $(S - \tilde{\lambda}_k)\tilde{X}_k$ for each k. If Q_{rk} denotes the b last elements of column k of Q (i.e., its rth block row), then with

$$\beta_{rk} := B_{r+1} Q_{rk}, \tag{3}$$

we have

$$(S - \tilde{\lambda}_k)\tilde{X}_k = E_{r+1}\beta_{rk}. \tag{4}$$

This can be derived by multiplying the Lanczos recursion $SE = ET + (0, \ldots, 0, 0, E_{r+1}B_{r+1})$ by Q from the right. The b-dimensional vector β_{rk} can be computed during the Lanczos iteration at negligible cost once Q has been found.

3. The error in the response. In this section the overall error in approximating the response $g(S)f$ is examined. That is, an estimate is sought for the norm of the error vector

$$u - \tilde{u} = g(S)f - \tilde{X}g(\tilde{\Lambda})\langle \tilde{X}, f \rangle. \tag{5}$$

Approximate mode superposition is a three-step process:

- project f onto the Ritz vectors $\tilde{X}_k$,
- multiply by functions $g(\tilde{\lambda}_k)$ of the Ritz values $\tilde{\lambda}_k$, and
- linearly combine the Ritz vectors to obtain $\tilde{u}$.

One possibility would therefore be to develop bounds that estimate the errors in each of these steps. These quantities must be combined to produce the final result. As a basis, bounds on the Ritz values and Ritz vectors would have to be used. One major disadvantage of this technique is that to find bounds on the Ritz vectors, gaps in the spectrum of S must be known (see [6] or [11]). Another drawback is that since all Ritz vectors should be included in the calculation, large intermediate error quantities appear. The technique developed below combines all steps of mode superposition into a single estimate thereby avoiding the above disadvantages.

For a differentiable function g, define Δg by

$$\Delta g(x, y) := \begin{cases} \frac{g(x)-g(y)}{x-y} & \text{if } x \neq y, \\ g'(x) & \text{if } x = y. \end{cases} \tag{6}$$

THEOREM 3.1. *Suppose that r steps of the Lanczos iteration with selfadjoint system matrix S and blocksize b produce the Ritz values $\tilde{\Lambda} = \operatorname{diag}(\tilde{\lambda}_k)$ and Ritz vectors $\tilde{X} = (\tilde{X}_k)$, $k = 1, \ldots, m$. Let f be completely represented in the space of Ritz vectors, $f \in \mathcal{E}$, and g be a differentiable function on the real axis. Both f and g may be real or complex valued. Then the norm of the error when approximating $u = g(S)f$ by $\tilde{u} = \tilde{X}g(\tilde{\Lambda})\langle \tilde{X}, f \rangle$ is bounded by*

$$\|u - \tilde{u}\|_M^2 \leq b \max_{j=1}^{n} \left\| \sum_{k=1}^{m} \Delta g(\lambda_j, \tilde{\lambda}_k)\beta_{rk} f_k \right\|^2, \tag{7}$$

where $f_k := \langle \tilde{X}_k, f\rangle$. *The numbers* λ_j, $j = 1, \ldots, n$, *are the exact eigenvalues of* S, *and* β_{rk} *is given by* (3).

Proof. Expand f in terms of the Ritz vectors

$$u - \tilde{u} = \sum_{k=1}^{m} (g(S) - g(\tilde{\lambda}_k))\tilde{X}_k f_k. \tag{8}$$

The norm of this error vector will be calculated by summing up the squares of its X_j-components. The X_j-component of the kth summand above has the form

$$\langle X_j, (g(S) - g(\tilde{\lambda}_k))\tilde{X}_k f_k\rangle = (g(\lambda_j) - g(\tilde{\lambda}_k))\langle X_j, \tilde{X}_k f_k\rangle. \tag{9}$$

This can be seen using the definition $g(S) := \sum_{i=1}^{n} X_i g(\lambda_i)\langle X_i, .\rangle$. If $\lambda_j \neq \tilde{\lambda}_k$, then a factor $(S - \tilde{\lambda}_k)/(\lambda_j - \tilde{\lambda}_k)$ can be added with similar reasoning

$$(g(\lambda_j) - g(\tilde{\lambda}_k))\langle X_j, \tilde{X}_k f_k\rangle = \frac{g(\lambda_j) - g(\tilde{\lambda}_k)}{\lambda_j - \tilde{\lambda}_k}\langle X_j, (S - \tilde{\lambda}_k)\tilde{X}_k f_k\rangle. \tag{10}$$

In case $\lambda_j = \tilde{\lambda}_k$, the left-hand side of (10) is zero and so is $\langle X_j, (S - \tilde{\lambda}_k)\tilde{X}_k f_k\rangle = \langle(\lambda_j - \tilde{\lambda}_k)X_j, \tilde{X}_k f_k\rangle$. Therefore, for $\lambda_j = \tilde{\lambda}_k$ the difference quotient in (10) can be replaced by any finite number, particularly $g'(\tilde{\lambda}_k)$. Taking advantage of the definition of Δg, we combine (9) and (10) to obtain

$$\langle X_j, (g(S) - g(\tilde{\lambda}_k))\tilde{X}_k f_k\rangle = \Delta g(\lambda_j, \tilde{\lambda}_k)\langle X_j, (S - \tilde{\lambda}_k)\tilde{X}_k f_k\rangle, \tag{11}$$

which is valid for all λ_j and all values of $\tilde{\lambda}_k$.

From the Lanczos recursion (see (4)) we know that $(S - \tilde{\lambda}_k)\tilde{X}_k = E_{r+1}\beta_{rk}$. The X_j-component of the sum over k hence becomes

$$\sum_{k=1}^{m} \langle X_j, (g(S) - g(\tilde{\lambda}_k))\tilde{X}_k f_k\rangle = \langle X_j, E_{r+1}\rangle \sum_{k=1}^{m} \Delta g(\lambda_j, \tilde{\lambda}_k)\beta_{rk} f_k. \tag{12}$$

The norm of the error can be found by summing up the squares of the absolute values over j

$$\begin{aligned} \|u - \tilde{u}\|_M^2 &= \sum_{j=1}^{n} \left| \langle X_j, E_{r+1}\rangle \sum_{k=1}^{m} \Delta g(\lambda_j, \tilde{\lambda}_k)\beta_{rk} f_k \right|^2 \\ &\leq \sum_{j=1}^{n} \|\langle X_j, E_{r+1}\rangle\|^2 \left\| \sum_{k=1}^{m} \Delta g(\lambda_j, \tilde{\lambda}_k)\beta_{rk} f_k \right\|^2, \end{aligned} \tag{13}$$

where the Cauchy–Schwarz inequality has been applied. $\langle X_j, E_{r+1}\rangle$ is a b-dimensional row vector. Taking the maximum over j in (13) we get the desired result since $\sum_{j=1}^{n} \|\langle X_j, E_{r+1}\rangle\|^2 = b$. □

In case we have no additional information about the spectrum of S, we must evaluate

$$\|u - \tilde{u}\|_M \leq \sqrt{b} \sup_{\xi \in \mathbb{R}} \left\| \sum_{k=1}^{m} \Delta g(\xi, \tilde{\lambda}_k)\beta_{rk} f_k \right\|. \tag{14}$$

Each maximization step in (14) produces the cost of $O(bm)$ flops, the sum represents a b-dimensional column vector. The maximum can, of course, be restricted to known subsets of

$\mathbb{R}$ containing the whole spectrum of S. For example, all exact eigenvalues are nonnegative if one calculates forced vibrations.

In Fig. 1 a typical shape of the function $h(\xi) := \left\| \sum_{k=1}^{m} \Delta g(\xi, \tilde{\lambda}_k)\beta_{rk} f_k \right\|$, which is maximized in (14), is shown (dashed line). The drawn lines correspond to the example given in §7.2 with $\frac{\omega}{2\pi} = 6$ Hz and are plotted against eigenfrequency. Frequency ν can be transformed to ξ by $\xi = (2\pi\nu)^2$.

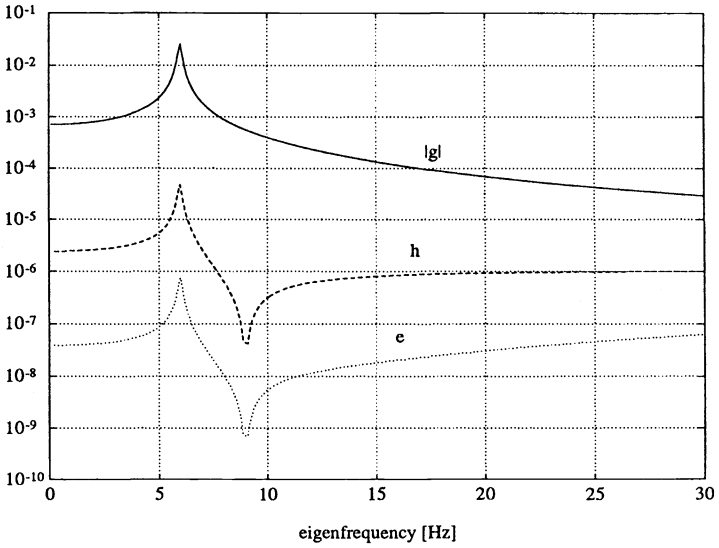


FIG. 1. *Typical shapes of* $|g|$, h, *and* e *plotted against eigenfrequency.*

To insure that $f \in \mathcal{E}$, the vector f should be included in the starting block of the Lanczos iteration. The fastest way of computing the response vector u is to combine the Lanczos vectors E without ever forming Ritz vectors. This also has the advantage that no additional cost of order n is introduced when all Ritz components are used. Ignoring some of the computed Ritz vectors, e.g., using only exact eigenvectors, therefore means no significant time savings but less accurate results as shown in the examples.

Now assume that a vector $f_\perp$ that is orthogonal to $\mathcal{E}$ is added to f. As long as we do not know about gaps in the spectrum, we must always assume that $f_\perp$ is parallel to an eigenvector X_p and that g attains its maximum at λ_p. So, in this case, the term $\|f_\perp\|_M \sup_{\xi\in\mathbb{R}} |g(\xi)|$ would have to be added to (14).

4. The error in evaluating functionals of the response. The response vector is evaluated at a single point or at a linear combination of points by applying a linear functional v to it. The error in the result is

$$\langle v, u\rangle - \langle v, \tilde{u}\rangle = \big\langle v, g(S)f - \tilde{X}g(\tilde{\Lambda})\langle \tilde{X}, f\rangle \big\rangle. \tag{15}$$

If f is contained in $\mathcal{E}$ but v possibly is not, this error can be estimated using the Cauchy–Schwarz inequality

$$|\langle v, u\rangle - \langle v, \tilde{u}\rangle| \le \|v\|_M \, \|g(S)f - \tilde{X}g(\tilde{\Lambda})\langle \tilde{X}, f\rangle\|_M. \tag{16}$$

Now the results of the last section can be applied to the second factor. $|\langle v, u\rangle - \langle v, \tilde{u}\rangle|$ can be regarded as a seminorm of the error vector $u - \tilde{u}$. As an abbreviation of the names Cauchy and Schwarz, inequality (16) will be called *CS seminorm error bound* in the sequel.

If, apart from f, v is also completely represented in $\mathcal{E}$, the actual error is generally smaller and better bounds can be given. To take advantage of this improvement, f and v should be included in the starting block of the Lanczos iteration. Similar to the last section, we define for a twice differentiable function g

$$\Delta^2 g(x,y,z) := \begin{cases} \frac{\Delta g(x,z)-\Delta g(y,z)}{(x-y)} & \text{if } x \neq y, \\ \frac{g'(y)-\Delta g(y,z)}{(y-z)} & \text{if } x = y \text{ and } x \neq z, \\ g''(x)/2 & \text{if } x = y = z. \end{cases} \tag{17}$$

THEOREM 4.1. *Apart from the assumptions in Theorem* 3.1 *let v also be completely represented in the space of Ritz vectors, $v \in \mathcal{E}$. The vector v may be real or complex valued and define $v_k := \langle \tilde{X}_k, v\rangle$. The function g shall be twice differentiable. Then the absolute value of the error in approximating $\langle v, u\rangle$ is bounded by*

$$\left|\langle v,u\rangle - \langle v,\tilde{u}\rangle\right| \leq b \cdot \max_{j=1}^{n} \left\| \sum_{l=1}^{m}\sum_{k=1}^{m} \bar{v}_l \beta_{rl} \Delta^2 g(\lambda_j, \tilde{\lambda}_l, \tilde{\lambda}_k) \beta_{rk}^{\top} f_k \right\|. \tag{18}$$

Proof. Both v and f are expanded in terms of Ritz vectors

$$\langle v, u-\tilde{u}\rangle = \sum_{l=1}^{m}\sum_{k=1}^{m} \left\langle \tilde{X}_l v_l, (g(S) - g(\tilde{\lambda}_k))\tilde{X}_k f_k \right\rangle. \tag{19}$$

Using the orthogonality among Lanczos blocks and relation (4), one can see that $(S-\tilde{\lambda}_k)\tilde{X}_k$ is orthogonal to $\mathcal{E}$ for all k. Consequently, each scalar product can be modified

$$\left\langle \tilde{X}_l v_l, (g(S) - g(\tilde{\lambda}_k))\tilde{X}_k f_k \right\rangle = \left\langle \tilde{X}_l v_l, (g(S) - g(\tilde{\lambda}_k) - b_{l,k}(S-\tilde{\lambda}_k))\tilde{X}_k f_k \right\rangle \tag{20}$$

with arbitrary numbers $b_{l,k}$. Choose $b_{l,k}$ as

$$b_{l,k} := \Delta g(\tilde{\lambda}_l, \tilde{\lambda}_k). \tag{21}$$

Without adding this term, the difference quotient in (23) would be unbounded near Ritz values. This in turn would make it impossible to find a sensible bound for the error.

The scalar product will be evaluated by summing up X_j-coordinates. Observe that if $\lambda_j \neq \tilde{\lambda}_l$ and $\lambda_j \neq \tilde{\lambda}_k$,

$$\langle \tilde{X}_l v_l, X_j\rangle \left\langle X_j, (g(S) - g(\tilde{\lambda}_k) - b_{l,k}(S-\tilde{\lambda}_k))\tilde{X}_k f_k \right\rangle \tag{22}$$

$$= \langle (S-\tilde{\lambda}_l)\tilde{X}_l v_l, X_j\rangle \frac{g(\lambda_j) - g(\tilde{\lambda}_k) - b_{l,k}(\lambda_j - \tilde{\lambda}_k)}{(\lambda_j - \tilde{\lambda}_l)\cdot(\lambda_j - \tilde{\lambda}_k)} \langle X_j, (S-\tilde{\lambda}_k)\tilde{X}_k f_k\rangle. \tag{23}$$

If $\lambda_j = \tilde{\lambda}_l$ or $\lambda_j = \tilde{\lambda}_k$, one of the scalar products in (23) is zero and so is (22). Therefore, the difference quotient can be replaced by an arbitrary number in these cases. Using the definition of $\Delta^2 g$, (22) now equals

$$\langle (S-\tilde{\lambda}_l)\tilde{X}_l v_l, X_j\rangle \Delta^2 g(\lambda_j, \tilde{\lambda}_l, \tilde{\lambda}_k) \langle X_j, (S-\tilde{\lambda}_k)\tilde{X}_k f_k\rangle \tag{24}$$

for all λ_j, $\tilde{\lambda}_l$, and $\tilde{\lambda}_k$.

At this point information from the Lanczos method (4) is introduced:

$$\langle X_j, (S-\tilde{\lambda}_k)\tilde{X}_k f_k\rangle = \langle X_j, E_{r+1}\rangle \beta_{rk} f_k = f_k \beta_{rk}^{\top} \langle E_{r+1}, X_j\rangle \tag{25}$$

and the same can be done with $\langle (S-\tilde{\lambda}_l)\tilde{X}_l v_l, X_j\rangle$. Summing up (23) over l, k, and j, we get

$$\begin{aligned}\langle v,u\rangle - \langle v,\tilde{u}\rangle &= \sum_{l=1}^{m}\sum_{k=1}^{m}\sum_{j=1}^{n}\langle X_j, E_{r+1}\rangle \bar{v}_l\beta_{rl}\Delta^2 g(\lambda_j,\tilde{\lambda}_l,\tilde{\lambda}_k)\beta_{rk}^{\top}f_k\langle E_{r+1},X_j\rangle \\ &= \sum_{j=1}^{n}\langle X_j,E_{r+1}\rangle\Bigg(\sum_{l=1}^{m}\sum_{k=1}^{m}\bar{v}_l\beta_{rl}\Delta^2 g(\lambda_j,\tilde{\lambda}_l,\tilde{\lambda}_k)\beta_{rk}^{\top}f_k\Bigg)\langle E_{r+1},X_j\rangle. \qquad (26)\end{aligned}$$

Note that up to now we have equality. Each summand of the outer sum can be estimated by taking norms

$$|\langle v,u\rangle - \langle v,\tilde{u}\rangle| \le \sum_{j=1}^{n}\|\langle E_{r+1},X_j\rangle\|^2\left\|\sum_{l=1}^{m}\sum_{k=1}^{m}\bar{v}_l\beta_{rl}\Delta^2 g(\lambda_j,\tilde{\lambda}_l,\tilde{\lambda}_k)\beta_{rk}^{\top}f_k\right\|. \qquad (27)$$

More exactly, the norm of the double sum could be replaced by the radius of the numerical range of this matrix. Because $\sum_{j=1}^{n}\|\langle X_j,E_{r+1}\rangle\|^2 = b$, we can take the maximum over the right factors to obtain the desired result. □

Inequality (18) will be called *full seminorm error bound* in the sequel. Again, if we have no additional information about the spectrum of S

$$|\langle v,u\rangle - \langle v,\tilde{u}\rangle| \le b\cdot\sup_{\xi\in\mathbb{R}}\left\|\sum_{l=1}^{m}\sum_{k=1}^{m}\bar{v}_l\beta_{rl}\Delta^2 g(\xi,\tilde{\lambda}_l,\tilde{\lambda}_k)\beta_{rk}^{\top}f_k\right\|. \qquad (28)$$

As in the last section one should, of course, take advantage of any such information, e.g., about gaps in the spectrum. Figure 1 shows the typical shape of the norm $e(\xi)$ of the double sum (dotted line) which is maximized in (28). Reference is again made to the numerical example given in §7.2. The double sum in (28) represents a $b\times b$ matrix. Instead of the euclidean matrix norm, the Frobenius norm is used in all calculations in this paper.

Care must be taken for calculating the differential quotients Δg and $\Delta^2 g$. Cancellation can occur. In case the numerator and the denominator approach zero, the quotient must be substituted by the appropriate derivative.

The factor $\Delta^2 g(\xi,\tilde{\lambda}_l,\tilde{\lambda}_k)$ can be regarded as an $m\times m$ matrix with indices l and k. Therefore, the function that must be maximized in (28) produces the cost of $O(bm^2)$ operations per evaluation. Since for $\xi\neq\tilde{\lambda}_l$ we have

$$\Delta^2 g(\xi,\tilde{\lambda}_l,\tilde{\lambda}_k) = \frac{1}{\xi-\tilde{\lambda}_l}(\Delta g(\xi,\tilde{\lambda}_k) - \Delta g(\tilde{\lambda}_l,\tilde{\lambda}_k)), \qquad (29)$$

the matrix multiplication can be economized. The matrix product

$$\sum_{k=1}^{m}\Delta g(\tilde{\lambda}_l,\tilde{\lambda}_k)\beta_{rk}^{\top}f_k \qquad (30)$$

must be found and stored once. Factorizing the rest of the computation into a k- and an l-part, only $O(bm)$ operations remain for any subsequent ξ in the maximization process.

5. Application to the calculation of forced vibrations. Calculating forced vibrations leads to evaluating functions of a matrix S applied to the load vector. Suppose that the dynamic behaviour of a structural system is described by the linear second-order differential equation

$$M\ddot{u}(t) + C\dot{u}(t) + Ku(t) = Fe^{i\omega t}, \qquad (31)$$

where M, C, and K are symmetric mass, damping, and stiffness matrix, respectively. The vector F represents the time-independent spatial distribution of the load and ω is its circular frequency. The solution $u(t) = ue^{i\omega t}$ is the time-dependent vector of nodal displacements and t represents time. A dot above a symbol denotes differentiation with respect to t. The damping matrix C shall be diagonalizable simultaneously with M and K. That is, modal damping will be applied. Formally C is a function of M and K, $C = Mc(M^{-1}K)$. Rayleigh's damping is obtained by setting $c(x) = a_1 + a_2 x$ with constants a_1 and a_2. C is never formed explicitly but is rather applied for each mode.

Equation (31) can be solved by obtaining a complete generalized eigendecomposition $KX_j = \mu_j MX_j$, $j = 1, \ldots, n$, of the matrix pencil (K, M). One may assume that the eigenvectors X_j form a complete orthonormal set. The scalar product used is $\langle a, b\rangle := a^* Mb$. In eigenvector coordinates, (31) decouples and the solution $u(t)$ can be given explicitly (see, e.g., [3])

$$u(t) = \sum_{j=1}^{n} X_j \frac{1}{\mu_j + i\omega c(\mu_j) - \omega^2} X_j^\top F e^{i\omega t}, \tag{32}$$

or when applying an arbitrary shift σ,

$$u(t) = \sum_{j=1}^{n} X_j \frac{\mu_j - \sigma}{\mu_j + i\omega c(\mu_j) - \omega^2} \langle X_j, (K - \sigma M)^{-1} F\rangle e^{i\omega t}. \tag{33}$$

When solving the generalized eigenproblem the Lanczos method is usually applied in "shift and invert mode" [13], [10], that is, $S := (K - \sigma M)^{-1} M$ is chosen as iteration matrix. Then S is selfadjoint with respect to $\langle ., .\rangle$ and convergence to eigenpairs is best for $\mu \approx \sigma$. The eigenvalues of the transformed system S are $\lambda_j = 1/(\mu_j - \sigma)$ and the eigenvectors X_j remain the same. Therefore, the solution of (31) can be written as

$$u(t) = g_{K,\omega}(S)(K - \sigma M)^{-1} F e^{i\omega t} \quad \text{with} \quad g_{K,\omega}(\lambda) := \frac{1}{1 + \lambda(\sigma - \omega^2 + i\omega c(1/\lambda + \sigma))}. \tag{34}$$

For $\sigma = 0$ this function $g_{K,\omega}$ is called "dynamic amplification factor." Alternatively,

$$u(t) = g_{M,\omega}(S) M^{-1} F e^{i\omega t} \quad \text{with} \quad g_{M,\omega}(\lambda) := \lambda g_{K,\omega}(\lambda), \tag{35}$$

which has the disadvantage that M^{-1} must be applied to generate the starting vector $M^{-1}F$. The input vector f will be defined either by $f = (K - \sigma M)^{-1} F$ or by $f = M^{-1} F$, whichever method as given above is applied. The index ω will frequently be dropped.

For viscous nonmodal damping the response u generally cannot be expressed in the form $g(S) f$ for some function g. Hence the theory of the previous sections does not apply.

The functions g given by (34) and (35) exhibit a sharp peak for $\lambda \approx \frac{1}{\omega^2 - \sigma}$ if σ is not near to ω^2. A semilogarithmic graph of $|g|$ for the example in §7.2 and $\frac{\omega}{2\pi} = 6$ Hz is given in Fig. 1 (solid line). Note that $|g|$ is plotted over eigenfrequency ν. Plotted against $\lambda = 1/((2\pi\nu)^2 - \sigma)$, the peak is narrower still. On one hand, this shape of g makes it possible to obtain good approximations of the response with relatively few eigenmodes corresponding to values around ω^2. On the other hand, finding good error bounds is difficult in this case if they rely on the approximation of g by a polynomial of degree r as in [4].

6. Rayleigh's damping. In the case of Rayleigh's damping, the expressions $\Delta g(\xi, \tilde{\lambda}_k)$ and $\Delta^2 g(\xi, \tilde{\lambda}_l, \tilde{\lambda}_k)$ can be written as a product and closed error formulas can be found. Set $c(\mu) := a_1 + a_2 \mu$ with constants a_1 and a_2, then g_M has the form $g_M(\lambda) = \frac{\lambda}{\alpha + \gamma\lambda}$. Here the

numbers α and γ are defined by $\alpha := 1 + i\omega a_2$ and $\gamma := \sigma - \omega^2 + i\omega(a_1 + a_2\sigma)$. Now the identities

$$\Delta g_M(x, y) = \frac{\alpha}{(\alpha + \gamma x)(\alpha + \gamma y)} \tag{36}$$

and

$$\Delta^2 g_M(x, y, z) = \frac{-\alpha\gamma}{(\alpha + \gamma x)(\alpha + \gamma y)(\alpha + \gamma z)} \tag{37}$$

can be derived easily. By pulling the ξ-dependent part out of inequalities (14) and (28) and by explicitly calculating the maximum of $|\frac{1}{\alpha+\gamma\xi}|$, one obtains the formulas

$$\|u - \tilde{u}\|_M \leq \sqrt{b}\, \frac{|\alpha\gamma|}{|\Im(\bar{\alpha}\gamma)|} \left\| \sum_{k=1}^{m} \frac{f_k}{\alpha + \gamma\tilde{\lambda}_k} \beta_{rk} \right\| \tag{38}$$

and

$$|\langle v, u\rangle - \langle v, \tilde{u}\rangle| \leq b\, \frac{|\alpha\gamma^2|}{|\Im(\bar{\alpha}\gamma)|} \left\| \left(\sum_{l=1}^{m} \beta_{rl} \frac{\bar{v}_l}{\alpha + \gamma\tilde{\lambda}_l} \right) \left(\sum_{k=1}^{m} \frac{f_k}{\alpha + \gamma\tilde{\lambda}_k} \beta_{rk}^{\top} \right) \right\|. \tag{39}$$

Both bounds are valid under the assumptions of Theorems 3.1 and 4.1 with $g = g_M$. For $g = g_K$, the right-hand sides must be multiplied by a factor $|\frac{\gamma}{\alpha}|$. The symbol $\Im$ denotes the imaginary part of a complex number. Both expressions can be evaluated with a small multiple of bm flops.

7. Numerical examples. The above methods have been applied to a forced response calculation for a container ship. The ship's finite element model contained 38,292 degrees of freedom (a hidden line plot is given in Fig. 2). The stiffness matrix K was a band matrix with a mean bandwidth of 400. The main point of investigation was the response to moments produced by gas forces in the engine. Three different spatial load distributions have been included in the calculation. The response was evaluated at 9 degrees of freedom. To include all these vectors, the blocksize was chosen to be 12. Twenty iterations have been performed and the shift σ was chosen corresponding to a frequency of 9 Hz. The implementation of the Lanczos algorithm contained "partial reorthogonalization" as described in [14].

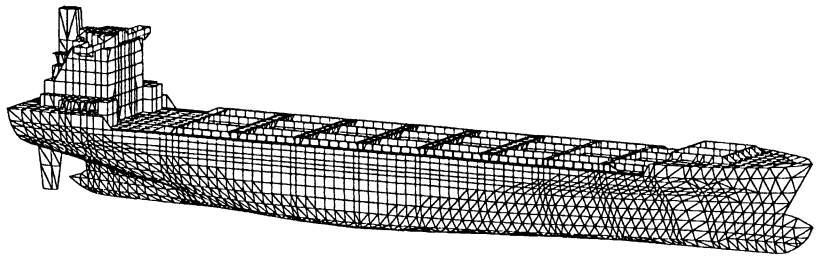


FIG. 2. *Finite element model of container ship.*

A lumped (diagonal) mass matrix was used. Therefore, M^{-1} could be applied and the method chosen was the one defined by g_M in (35). Numerical experience with different examples shows that the error bounds become slightly better when using the method given by g_K in (34). This improvement is achieved by one additional (expensive) application of S. The error bounds are, of course, equally applicable to the case of a nondiagonal mass matrix. Then evaluation of M^{-1} should be avoided and hence the g_K-method would have to be used.

All graphs in this section are plotted against frequency of excitation; that is, the parameter ω is varying in $g_{M,\omega}(\lambda)$.

7.1. Rayleigh's damping. The critical damping ratio was chosen to be linear in frequency, 0% at 0 Hz and 3% at 10 Hz. Figure 3 shows the relative norm errors after various numbers of iteration steps. More precisely, the lines show $\|u - \tilde{u}\|_M / \|\tilde{u}\|_M$, the error bound as given by (38) divided by the norm of the calculated response. As expected the results are most accurate around the shift σ.

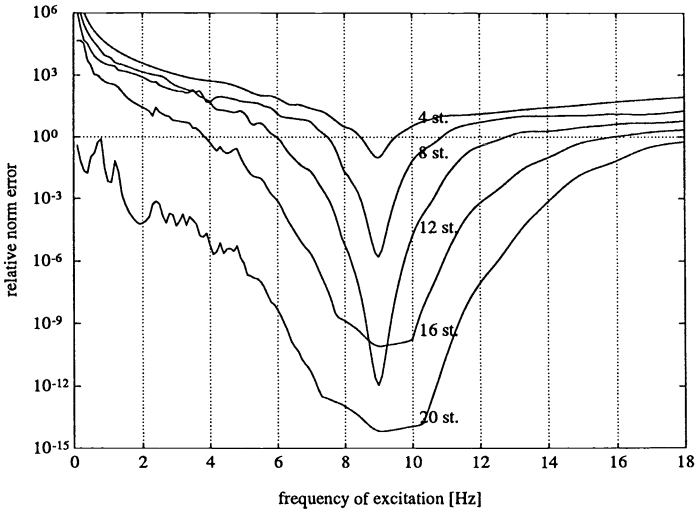


FIG. 3. *Relative norm error bounds after various numbers of iteration steps.*

For the comparisons in Figs. 4–6, the response was evaluated at a particular degree of freedom on the ship's bridge. The solid lines in Fig. 4 show the relative errors found using (39) after 8, 14, and 20 iteration steps. The CS seminorm error bounds are given by the dashed lines. Clearly, it is advisable to include the vectors v of measurement degrees of freedom (which define the evaluation functionals) in the starting block to be able to prove additional accuracy in the results.

In Fig. 5 the full seminorm error bounds (solid lines) are compared to the actual error (dashed lines). The term *actual error* will be used for the difference to the response after 20 steps (plus its error estimate) in the following. The error bound after 20 steps is given for reference. Between 6 and 16 Hz the error is overestimated by two to three orders of magnitude. Due to finite precision (about 16 decimals), the actual error does not fall below a certain limit (here 10^{-14}) although the error bounds predict better approximation.

Figure 6 shows the effect of ignoring bad Ritz vectors. In these examples, the term *bad* will be used for those Ritz pairs with negative values and with values above 30 Hz. Conventionally, such Ritz pairs would not be included in a forced response calculation with the given frequencies of excitation. According to the relative error bound $\|\beta_{rk}\|/|\lambda_k|$ (see [11, p. 69]) all mentioned Ritz values had been found with a relative precision less than 70%. The solid lines give the actual errors if all Ritz vectors are used for calculating the response. Omitting bad Ritz pairs leads to the errors shown by the dashed lines. One can say that including inexact eigenvectors leads to more accurate results at those points where a modest degree of accuracy has already been achieved.

7.2. Quadratic damping below 10 Hz. Here the critical damping ratio was chosen to be a quadratic function of frequency, 0.5% at 0 Hz, 0.9% at 4 Hz, and 3% at 10 Hz. Above 10 Hz the same damping function as in the previous section was used. Force vector f and

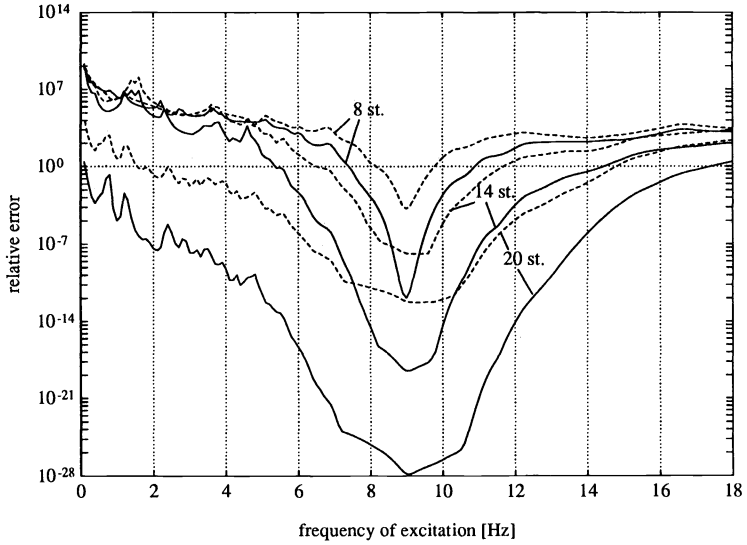


FIG. 4. *Comparison of error bounds for the evaluation of the response at a specific degree of freedom. Dashed line*: CS *seminorm error bounds. Solid line*: *Full seminorm error bounds.*

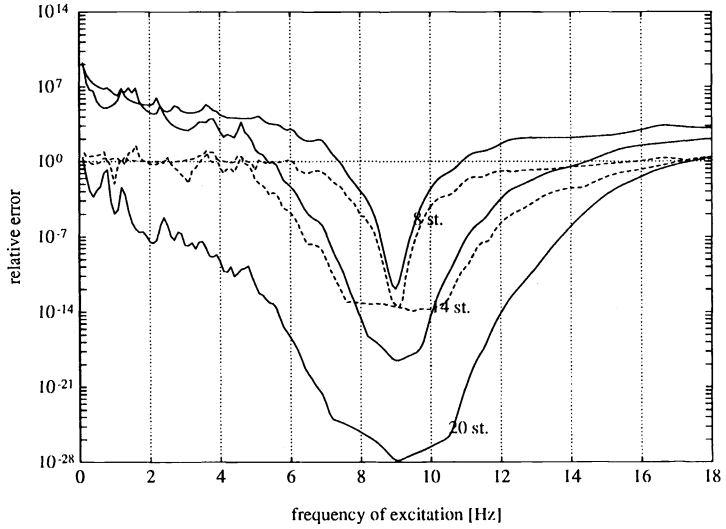


FIG. 5. *Comparison full seminorm error bound—actual error. Dashed line*: *Actual error. Solid line*: *Full seminorm error bound.*

measurement vector v were the same as above. The maximum in the error bounds was taken over all of $\mathbb{R}^+$. After 20 steps all eigenvalues between 0 and 9 Hz had been found. Consequently, gaps in this interval could be excluded from the maximum in this case.

Figure 7 shows that the norm error is higher than for Rayleigh's damping. More steps are needed to obtain a given accuracy. In Fig. 8 the full seminorm error bound after 16 steps (solid line) is compared to the actual error (dashed line) as explained in the previous example. The response error after 20 steps is again given for reference. Between 6 and 13 Hz the error is overestimated by about four orders of magnitude. The dashed-dotted line gives the actual errors when ignoring bad Ritz vectors in the same way as in the previous example.

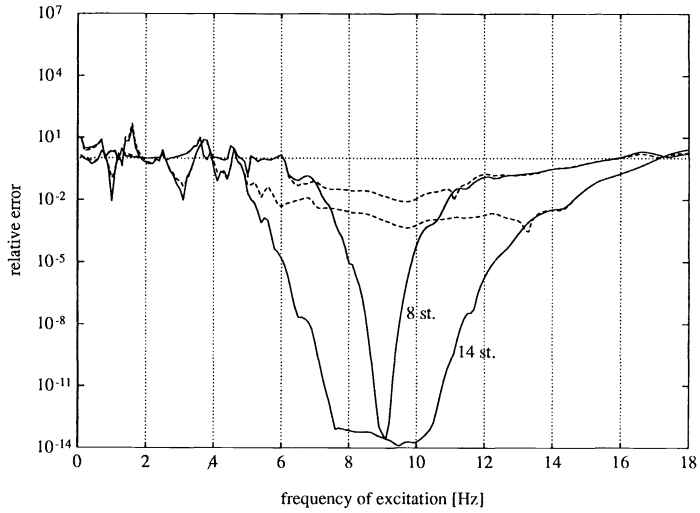


FIG. 6. *Comparison of actual errors with and without ignoring bad Ritz vectors. Solid line: Actual error. Dashed line: Actual error, bad vectors ignored.*

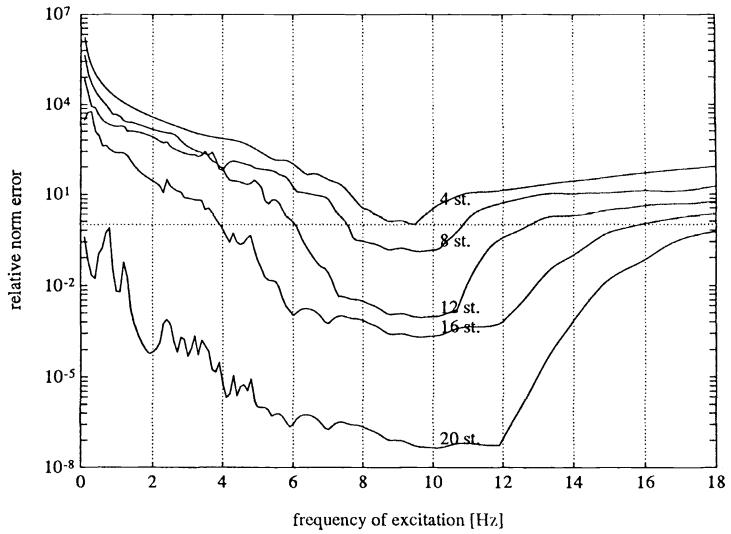


FIG. 7. *Relative norm error bounds after various numbers of iteration steps.*

7.3. Reduced number of relevant eigenmode components in the input vectors. This example shows that the error bounds can also provide useful information in a frequency range where several eigenpairs have not yet been found. S was chosen to be diagonal and M was set to the identity matrix. With other words, the canonical scalar product has been used. System dimension was $n = 1000$. The first 320 entries of the diagonal of S were set to the eigenvalues below 25 Hz of the container ship above. Six hundred and eighty eigenvalues higher than 25 Hz have been chosen at random with roughly the same spacing as the first 320 values. Diagonal test matrices in connection with the Lanczos algorithm have, for example, been used in [15]. Since the Lanczos algorithm only applies to a given system, it does not take advantage of the special form of S.

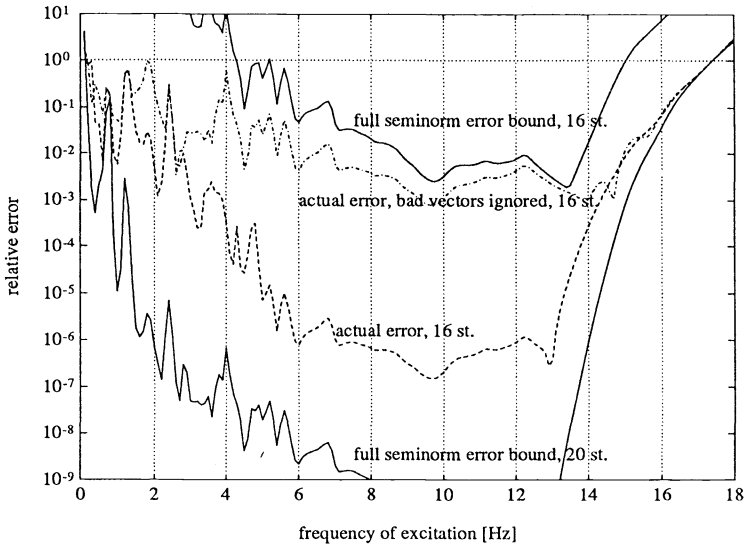


FIG. 8. *Comparison full seminorm error bound—actual errors with and without ignoring bad Ritz vectors.*

The vectors f and v were chosen at random such that about 350 of the entries of f and 220 of those of v had absolute values between 10^{-1} and 10^{-5}. The other entries had absolute values around 10^{-8}. Since f was complex and v was real, the blocksize was chosen to be $b = 3$. The starting block contained v and the real and imaginary parts of f. Lanczos iteration was stopped after ten steps and damping was chosen as in §7.1.

In Fig. 9 the exact eigenvalues are shown by a plus (+) sign. An x marks an eigen value found by the Lanczos iteration. Its vertical position represents the estimate $\|\beta_{rk}\|/|\lambda_k|$ for its relative accuracy (see above). Between 12 and 15 Hz only 14 out of 29 eigenvalues and -vectors have been approximated. Nevertheless, the error bounds show that the calculated response has a relative error less than 10% in this interval. This is true for the norm (dashed line) as well as for the functional of the response (solid line, full seminorm error bound).

The described effect can become useful if the structure under consideration contains a large number of eigenmodes that are not relevant near the points of excitation and evaluation, but correspond to eigenvalues near ω^2. In particular this refers to vibrating subsystems. The error bounds can predict good approximation of the response even if these modes have not been found. As a result, the Lanczos iteration can be stopped earlier as when using a conventional stopping criterion like "calculate all modes in the desired frequency range."

8. Conclusions. Using the formulas developed above, it is possible to obtain a posteriori error bounds for the process of mode superposition. The bound for the norm error gives the overall error of the approximate solution if the input vector is included in the starting block of the Lanczos algorithm. In case the response is evaluated at a single degree of freedom or at a linear combination thereof, the second bound gives more favourable results. Then the functional for this evaluation should also be included in the starting block.

The error estimates do not make use of bounds on the Ritz vectors. They directly measure the accuracy of the whole process of approximating the response. This is in contrast to the approach of dividing it into the three usual steps of mode superposition. No specific information (e.g., gaps) about the spectrum of the system matrix needs to be known.

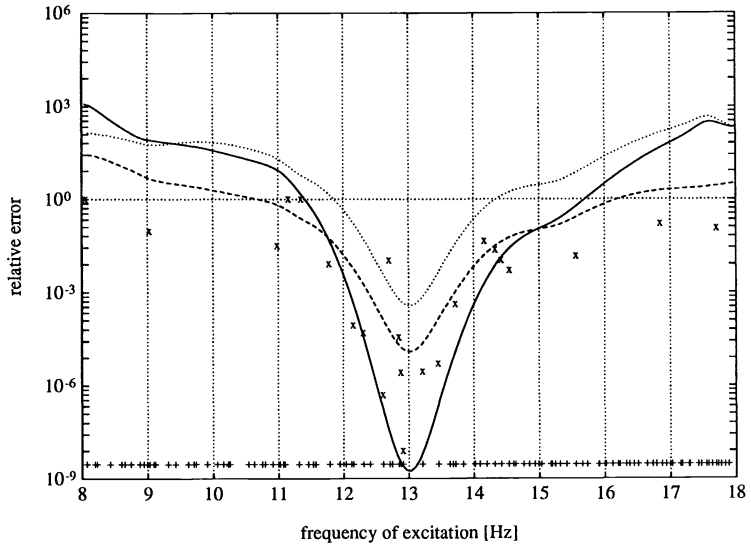


FIG. 9. *Response approximation when input vectors contain* $\approx$ *25% relevant eigenmode components. Solid: full seminorm error bound. Dashed: norm error bound. Dotted: CS seminorm error bound.* +: *exact eigenvalues.* x: *found eigenvalues.*

The error bounds can be included in an implementation of the Lanczos method. There they provide a stopping criterion for the iteration that reflects the quality of the overall result instead of measuring intermediate quantities. It might well be that the response is predicted with satisfactory precision although the Ritz vectors would not be regarded as sufficiently accurate. In the case of forced vibrations it is possible that they predict good approximation in a frequency range where some modes have not yet been found by the Lanczos iteration. These missing modes would be those with a very small component in the starting block. In other words, the iteration can be stopped at an early stage, thereby reducing computational cost when contrasted to a conventional stopping criterion such as "calculate all modes in the desired frequency range."

The effect of improving results by including all Ritz vectors in the calculation has already been stated in the literature. The numerical examples given show that the improvement mainly occurs at points where a moderately accurate result has already been achieved. In case Lanczos vectors are combined directly to find the response, ignoring some of the Ritz vectors means no significant time savings but less accurate calculations.

Numerical examples show that if the bounds predict good approximation, the error is overestimated by about two to four orders of magnitude in large-scale engineering applications. The cost for their calculation is $O(bm)$ floating-point operations in the case of norm errors where m is the number of Ritz values found and b is the blocksize. The cost for bounding seminorms of the error is dominated by a small multiple of bm^2 flops. If Rayleigh's damping is applied, all bounds can be evaluated with a small multiple of bm flops.

It is straightforward to apply the results to different matrix functions such as the matrix exponential.

Acknowledgments. The author would like to thank Dr. H. G. Matthies for many valuable discussions and Dr. R. Ansorge and Dr. H. Voß for their advice and support.

REFERENCES

[1] R. ARNOLD, R. CITERLEY, M. CHARGIN, AND D. GALANT, *Application of Ritz vectors for dynamic analysis of large structures*, Comput. & Struct., 21 (1985), pp. 461–467.

[2] H. C. CHEN AND R. L. TAYLOR, *Using Lanczos vectors and Ritz vectors for computing dynamic responses*, Engrg. Comput., 6 (1989), pp. 151–157.

[3] R. W. CLOUGH AND J. PENZIEN, *Dynamics of Structures*, McGraw-Hill, New York, 1975.

[4] V. DRUSKIN AND L. KNIZHNERMAN, *Two polynomial methods of calculating functions of symmetric matrices*, U.S.S.R. Comput. Math. and Math. Phys., 29 (1989), pp. 112–121.

[5] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, and London, 1989.

[6] H. G. MATTHIES, *Computable error bounds for the generalized symmetric eigenproblem*, Comm. Appl. Numer. Methods, 1 (1985), pp. 33–38.

[7] B. NOUR-OMID, *Lanczos method for heat conduction analysis*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 251–262.

[8] B. NOUR-OMID AND R. CLOUGH, *Dynamic analysis of structures using Lanczos coordinates*, Earthquake Engrg. Struct. Dynamics, 12 (1984), pp, 565–577.

[9] ———, *Block Lanczos method for dynamic analysis of structures*, Earthquake Engrg. Struct. Dynamics, 13 (1985), pp. 271–275.

[10] B. NOUR-OMID, B. PARLETT, T. ERICSSON, AND P. S. JENSEN, *How to implement the spectral transformation*, Math. Comput., 48 (1987), pp. 663–673.

[11] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, 1980.

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

[13] D. SCOTT, *The advantages of inverted operators in Rayleigh–Ritz approximations*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 68–75.

[14] H. SIMON, *The Lanczos algorithm with partial reorthogonalization*, Math. Comput., 42 (1984), pp. 115–142.

[15] H. VAN DER VORST, *An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.

[16] E. WILSON, M. YUAN, AND J. DICKENS, *Dynamic analysis by direct superposition of Ritz vectors*, Earthquake Engrg. Struct. Dynamics, 10 (1982), pp. 813–821.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 16–35, January 1994

GLOBAL OPTIMIZATION DECOMPOSITION METHODS FOR BOUNDED PARAMETER MINIMAX RISK EVALUATION*

ERIC GOURDIN†, BRIGITTE JAUMARD‡, AND BRENDA MACGIBBON§

Abstract. There has been much recent statistical research in the area of inference under constraints. The problem considered here is that of bounded parameter estimation, in particular that of normal and Poisson means, using minimaxity as the criterion of evaluation. Because of the ease of calculation of linear minimax rules, the ratio of these risks to the nonlinear minimax risks for these problems is also studied. To find the minimax solution, the dual problem of finding the least favorable prior distribution is often considered. On bounded parameter spaces the least favorable prior is often discrete, so the finding of the minimax estimator and its risk is equivalent to a global optimization problem with constraints. Previously published numerical specifications of the priors have used iterative (often heuristic) procedures. Two global optimization procedures are proposed. The first is based on multivariate Lipschitz optimization and makes use of bounds on the first-order derivatives. The second is a decomposition procedure that utilizes the partial concavity of the Bayes risk function. Both procedures are compared, and the decomposition method appears to be much more efficient. It is shown that the Ibragimov–Hasminskii constant, the maximum of the ratio of linear to nonlinear minimax risks, is different for the Poisson and normal problems.

Key words. decomposition, Ibragimov–Hasminskii constant, bounded parameter, minimax, discrete least favorable prior

AMS subject classifications. 90C26, 62-04, 62F10

1. Introduction.

In the framework of statistical decision theory, many interesting problems can be resolved by finding minimax procedures. These problems include the areas of estimation, hypothesis testing, experimental design, and robust statistics, among others. Recently, the problem of estimating the mean of a standard Gaussian shift experiment, when it is known to be in an infinite-dimensional bounded convex subspace, has been studied by Pinsker [27] in the ellipsoid case, and by Donoho, Liu, and MacGibbon [7] for more general bodies.

More precisely, it is supposed that

$$x_i = \theta_i + \varepsilon_i, \qquad i = 0, 1, 2, \ldots,$$

where the ε_i are independent identically distributed $N(0, \sigma^2)$, σ^2 known and θ_i unknown, but known to lie in an orthosymmetric quadratically convex ℓ_p ($p \geq 2$) body Θ. An estimator $\hat{\theta}^*$ of θ is minimax with respect to quadratic loss if

$$\sup_{\theta \in \Theta} E\left(\sum (\hat{\theta}_i^* - \theta_i)^2\right) = \inf_{\hat{\theta}} \sup_{\theta \in \Theta} E\left(\sum (\hat{\theta}_i - \theta_i)^2\right).$$

The estimator $\hat{\theta}^*$ is linear minimax if it is linear and satisfies the above equation with the infimum over $\hat{\theta}$ referring only to linear procedures. The resulting risk is called the linear minimax risk. The fact that linear procedures are much easier to calculate raises interest in the study of the ratio of linear to nonlinear minimax risk. Donoho, Liu, and MacGibbon [7] showed that this ratio for the orthosymmetric quadratically convex ℓ_p body problem ($p \geq 2$)

*Received by the editors October 29, 1990; accepted for publication (in revised form) January 25, 1993.

†École Polytechnique de Montréal, Département de Mathématiques Appliquées, Succursale A, Case Postale 6079, Montréal (Québec) Canada H3C 3A7 (eric@crt.umontreal.ca).

‡GERAD and École Polytechnique de Montréal, Département de Mathématiques Appliquées, Succursale A, Case Postale 6079, Montréal (Québec) Canada H3C 3A7 (brigitt@crt.umontreal.ca). This research was supported by the Natural Sciences and Research Council of Canada grant GP0036426 and Formation de Chercheurs et d'Aide à la recherche, Quebec grant 90NC0305.

§GERAD and UQAM Département de Mathématiques et d'Informatique, Succursale A, Case Postale 8888, Montréal (Québec) Canada H3C 3P8. This research was supported by the Natural Sciences and Research Council of Canada grant OGP0092037 and Formation de Chercheurs et d'Aide à la recherche, Québec grant 89EQ2452.

could be bounded by the ratio for the hardest one-dimensional subproblem. Thus the problem is reduced to studying the ratio $\mu_N(m)$ of linear minimax risk to nonlinear minimax risk for the estimation of a bounded normal mean; that is, estimating the mean of a normal population $N(\theta, 1)$ given that $|\theta| \leq m$, on the basis of one observation $X = x$. It is well known that the minimax linear risk is m^2/m^2+1, and if $r_N(m)$ denotes the minimax risk among all estimators, then the ratio of interest $\mu_N(m)$ is defined by

$$\mu_N(m) = \frac{m^2}{(1+m^2)r_N(m)}.$$

Let $\mu_N^* = \sup_m \mu_N(m)$. This constant, the worst-case ratio, is referred to as the Ibragimov–Hasminskii constant [17], since they were the first to study the behaviour of $\mu_N(m)$ and to prove that μ_N^* is finite.

Casella and Strawderman [6] were the first to provide analytic and numerical results for the minimax risk $r_N(m)$ for the one-dimensional normal mean problem. To find the minimax solution, the dual problem of finding the least favorable prior distribution for the corresponding Bayes problem is often considered. The duality theory ensures that, if there exist solutions to both problems, then the Bayes procedure with respect to the least favorable prior distribution will be minimax (see Wald [33], Ferguson [11], Berger [3], Kempthorne [20], Brown [5]).

For many of these problems, particularly for constrained inference (see, e.g., Ghosh [13]), the least favorable prior distribution is a discrete measure with finite support. Thus, the finding of a minimax solution becomes a global optimization of a nonlinear, nonconvex function. Kempthorne [20] was the first to describe a convergent iterative procedure for the numerical specification of such priors. Donoho, Liu, and MacGibbon [7] used Brown's identity (see Bickel [4]) involving the Bayes risk and Fisher information to approximate an upper bound for the minimax risk $r_N(m)$ in the bounded normal mean problem, thus reducing the optimization to that of minimizing a convex functional subject to convex constraints. The numerical specification was achieved using the optimization system NPSOL of Gill, Murray, Saunders, and Wright [12]. Within four-digit accuracy, Donoho, Liu, and MacGibbon [7] showed that the Ibragimov–Hasminskii constant was bounded by 1.2497. A more recent numerical study by Feldman and Brown [10] suggests that the constant is approximately 1.2465.

Johnstone and MacGibbon [18] studied the problem of estimating a bounded Poisson parameter under normalized quadratic loss. Using an iterative procedure similar to Kempthorne's algorithm involving local optimization techniques (see Kempthorne [20]), they calculated the minimax risk and obtained an upper bound of 1.251 for the ratio of nonlinear to linear minimax risk for this Poisson problem.

An interesting question now becomes whether or not this ratio is the same for the normal and Poisson problems. Since the specification of the minimax solution is a global optimization problem, it seems essential to develop such techniques with sufficient accuracy to resolve this issue.

We first designed a new Lipschitz algorithm concerned with the following global optimization problem :

$$(P)\begin{cases} \text{maximize} & f(x), \\ \text{subject to :} & g_i(x) \leq 0, \qquad i = 1, 2, \ldots, m, \\ & x \in \mathbb{R}^n, \end{cases}$$

where the functions f and g_i $(i = 1, 2, \ldots, m)$ satisfy a Lipschitz condition, but are not necessarily either concave or linear.

Such problems are very difficult to solve in practice. Most often, they are solved heuristically, using nonlinear optimization algorithms that only yield a local optimum or an unproved global optimum. However, more efficient algorithms can be designed if specific properties are taken into account, e.g., polynomial expressions of the constraints (see, e.g., Duffin, Peterson, and Zener [8], Ecker [9]) or convexity/linearity of the constraints functions (see, e.g., Horst and Tuy [16], Rosen [31]). The multivariate Lipschitz algorithm we propose, although exact and theoretically convergent, did not allow us to reach highly precise results in a reasonable amount of computing time. Therefore, we considered a second algorithm.

The Bayes risk function involved in the minimax estimation problem presents an interesting property of partial concavity (see Kempthorne [20], Johnstone and MacGibbon [18]). To take advantage of this property, we improved our first algorithm, using decomposition techniques.

The second algorithm deals with the following partially concave global optimization problem:

$$(\tilde{P}) \begin{cases} \text{maximize} & f(x, y), \\ \text{subject to :} & g_i(x, y) \leq 0, \qquad i = 1, 2, \ldots, m, \\ & x \in \mathbb{R}^{n_x},\ y \in \mathbb{R}^{n_y}, \end{cases}$$

where the functions f and g_i $(i = 1, 2, \ldots, m)$ satisfy a Lipschitz condition with respect to x, and are continuous and concave with respect to y.

Both algorithms are applied to the evaluation of the minimax risk for a bounded Poisson mean with a suitably normalized quadratic loss function. The second algorithm appears to be much more efficient than the first one.

The paper is organized as follows. The next section is devoted to the two algorithms. First, we introduce the Lipschitz optimization theory (§2.1); then we explain our new multivariate Lipschitz optimization algorithm (§2.2). Finally, we present the decomposition algorithm (§2.3). Section 3 will be devoted to the minimax estimation of bounded Poisson and normal means and the calculation of the Ibragimov–Hasminskii constants for these problems. Computational experiences, including some on the minimax risk evaluation for the bounded normal and Poisson mean problems, are reported in §4. Conclusions are drawn in §5.

2. Global optimization methods.

2.1. Introduction to Lipschitz optimization. Piyavskii's algorithm [28], [29] (see also Shubert [32]) is one of the most classical algorithms of univariate Lipschitz optimization. It addresses the problem of maximizing a continuous univariate function f over an interval [a,b], assuming it satisfies a Lipschitz condition

$$\forall x, y \in [a, b], \quad | f(x) - f(y) | \leq L | x - y |,$$

where L is a constant.

The algorithm generates a sampling sequence of points $x^1, x^2, \ldots, x^k$, where the function f is evaluated, and builds a piecewise linear upper bounding function, also called a sawtooth cover due to its shape, and consists of the lower envelope of functions $F_i(x) = f(x^i) + L | x - x^i |$ for $i = 1, 2, \ldots, k$.

Although it is not a best possible algorithm, it can be modified so that it becomes nearly optimal; see Hansen, Jaumard, and Lu [15].

Piyavskii's algorithm can be generalized (see Piyavskii [29], Mladineo [26]) to handle the problem of maximizing a continuous multivariate function $f(x)$, $x \in \mathbb{R}^n$, over an n-cube $C = \{x \in \mathbb{R}^n \mid a \leq x \leq b\}$, assuming f satisfies a Lipschitz condition

$$\forall x, y \in C \quad | f(x) - f(y) | \leq L \| x - y \|,$$

where L is a constant and $\| \cdot \|$ defines the Euclidean norm. Other algorithms, using different partitioning techniques, have been proposed, see, e.g., Meewela and Mayne [23], [24].

The graph of the upper bounding function generalizing the univariate sawtooth cover to the multidimensional case, is made up of several intersecting cones that approximate the graph of f. In using Piyavskii's sampling rule for computations, one must find the peaks of the approximating surface, which are cone intersections, and, of those, the one that is the highest. This is rather difficult in practice, as it reduces to a system of $n + 1$ equations (n linear and one quadratic).

Indeed, it is easier to build a piecewise constant upper bounding function, where each cone is approximated by its cover. This new algorithm is described in the next section.

2.2. A multivariate Lipschitz optimization algorithm (MLip). The algorithm described in this section is designed for multivariate Lipschitz optimization problems:

$$\begin{cases} \text{maximize} & f(x), \\ \text{subject to:} & x \in C = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n], \end{cases}$$

where f satisfies a Lipschitz property with a constant L.

2.2.1. Description of the algorithm. The algorithm subdivides the initial n-cube into an increasing number of n-rectangular cells and approximates the objective function by a constant in each cell.

More precisely, suppose that at a given stage of the algorithm, the original n-cube C is already subdivided in k cells $C^1, C^2, \ldots, C^k$, such that

$$C = \bigcup_{i=1}^{k} C^i,$$

$$\forall i \quad C^i = \{x \in \mathbb{R}^n \mid a^i \leq x \leq b^i\}.$$

Also suppose that a constant upper bound F^i of function f has been computed in each cell C^i. We call each pair (F^i, C^i) a subproblem of the original problem, and we denote it by P^i. The algorithm finds the subproblem $P^r = (F^r, C^r)$ of highest upper bound

$$F^r = \max_{i=1,\ldots,k} F^i.$$

This value is also the current best overestimation of f^* (the maximum of f over C):

$$F_{\text{opt}}^{k+1} = F^r.$$

The corresponding cell C^r is sliced along its longest edge in p equal subcells $C^{r_1}, C^{r_2}, \ldots, C^{r_p}$. In practice, $p = 3$ performed better than $p = 2$, odd values were better than even ones, and large values tended to increase the memory requirements. It was decided that the choice of $p = 3$ is a good compromise. The midpoint x^{r_j} of each subcell C^{r_j} is found and $f(x^{r_j})$ is computed. The value of the $f(x^{r_j})$ for $j = 1, 2, \ldots, p$, is used to update f_{opt}, the current best underestimation of f^*:

$$f_{\text{opt}}^{k+1} \longleftarrow \max\Big(\max_{i=1,2,\ldots,k} f_{\text{opt}}^i, f(x^{r_1}), f(x^{r_2}), \ldots, f(x^{r_p}) \Big).$$

The size D^{r_j}, i.e., the length of the diagonal of each subcell, is also computed, and a new constant upper bound is found in each subcell, according to

$$F^{r_j} = f(x^{r_j}) + \frac{1}{2} L D^{r_j}.$$

Note that each F^{r_j}, $j = 1, 2, \ldots, p$, improves strictly on F^r. Subproblem $P^r = (F^r, C^r)$ is then replaced, in the list $\mathcal{L}$ of all subproblems, by the p subproblems

$$P^{r_j} = (F^{r_j}, C^{r_j}), \qquad j = 1, 2, \ldots, p.$$

Before considering a new cell, the algorithm discards from the list $\mathcal{L}$ all subproblems with an upper bound smaller than f_{opt}, as they cannot contain a vector x^* of optimal value $f(x^*) = f^*$. The process is carried out until $F_{\text{opt}} - f_{\text{opt}}$ becomes less than a chosen tolerance ε.

Although the upper bounding function built by this algorithm is less precise than the sawtooth cover considered in the multidimensional Lipschitz algorithm introduced in the previous section, experiments show that this does not affect the efficiency of the method in practice, both in terms of computing time and in terms of function evaluations. In most of the cases, it even leads to better results.

2.2.2. Algorithm MLip. The details of the algorithm are given below.

Notations:
L is a Lipschitz constant for f over C,
$\mathcal{L}$ is the list of active (that is, not yet discarded) subproblems.
At the current iteration, a subproblem $P^k = (F^k; C^k)$ is characterized by:
C^k, the cell $[a_1^k, b_1^k] \times [a_2^k, b_2^k] \times \cdots \times [a_n^k, b_n^k]$,
F^k, an upper bound of f in cell C^k,
$\ell_i^k = b_i^k - a_i^k$, the length of the ith edge of the cell C^k,
$D^k = \sqrt{(\ell_1^k)^2 + (\ell_2^k)^2 + \cdots + (\ell_n^k)^2}$, the size of cell C^k.

Initialization

$C^1 \longleftarrow [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$;

$x^1 \longleftarrow \left(a_1 + \frac{1}{2}\ell_1, a_2 + \frac{1}{2}\ell_2, \ldots, a_n + \frac{1}{2}\ell_n\right)$;

$D^1 \longleftarrow \sqrt{\ell_1^2 + \ell_2^2 + \cdots + \ell_n^2}$;

$x_{\text{opt}} \longleftarrow x^1$;

$f_{\text{opt}} \longleftarrow f(x_{\text{opt}})$;

$F^1 \longleftarrow f(x^1) + \frac{1}{2} L D^1$;

$F_{\text{opt}} \longleftarrow F^1$;

$P^1 \longleftarrow (F^1; C^1)$;

$\mathcal{L} \longleftarrow \{P^1\}$;

$p \longleftarrow 3$;

Upper bounding cover

While $F_{\text{opt}} - f_{\text{opt}} > \varepsilon$ **do**

Let P^r be the subproblem of $\mathcal{L}$ with the largest upper bound F^r:

$P^r = (F^r; C^r)$;

$\ell_{i_\circ}^r \longleftarrow \max_{i=1,2,\ldots,n} \ell_i^r$;

$D^r \longleftarrow \left(\sum_{\substack{i=1 \\ i \neq i_\circ}}^{n} \left(\ell_i^r\right)^2 + \left(\frac{\ell_{i_\circ}^r}{p}\right)^2 \right)^{\frac{1}{2}}$;

Remove P^r from $\mathcal{L}$;

For j : 1 through p **do**

$A^{r_j} \longleftarrow \left(a_1^{r_j}, a_2^{r_j}, \ldots, a_{i_\circ-1}^{r_j}, a_{i_\circ}^{r_j} + \frac{j-1}{p}\ell_{i_\circ}^{r_j}, a_{i_\circ+1}^{r_j}, \ldots, a_n^{r_j}\right)$;

$L^{r_j} \longleftarrow \left(\ell_1^{r_j}, \ell_2^{r_j}, \ldots, \ell_{i_\circ-1}^{r_j}, \frac{\ell_{i_\circ}^{r_j}}{p}, \ell_{i_\circ+1}^{r_j}, \ldots, \ell_n^{r_j}\right)$;

$x^{r_j} \longleftarrow A^{r_j} + \frac{1}{2}L^{r_j}$;

$F^{r_j} \longleftarrow f(x^{r_j}) + \frac{1}{2}L D^{r_j}$;

If $f_{\text{opt}} < f(x^{r_j})$ **then**

$f_{\text{opt}} \longleftarrow f(x^{r_j})$;

$x_{\text{opt}} \longleftarrow x^{r_j}$

Endif;

Add the subproblem $P^{r_j} = (F^{r_j}\ ;\ C^{r_j})$ to $\mathcal{L}$

EndFor;

Discard from $\mathcal{L}$ all subproblems with an upper bound smaller than f_{opt}

EndWhile.

The last step, which consists in discarding subproblems from $\mathcal{L}$, is not necessary for the convergence of the algorithm. Its aim is only to improve the practical efficiency of the algorithm. Moreover, a *max-min heap* (see Atkinson et al. [1]) can also be used to improve the efficiency. It corresponds to the merging of a *max-heap* and a *min-heap* containing all the subproblems of $\mathcal{L}$ ranked, respectively, in decreasing order and in increasing order, by their upper bound.

2.2.3. The constrained case. Consider now the constrained optimization problem

$$\begin{cases} \text{maximize} & f(x), \\ \text{subject to :} & g_i(x) \leq 0 \quad i = 1, 2, \ldots, m, \\ & x \in C = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n], \end{cases}$$

where both the objective f and the constraint functions g_i, $(i = 1, 2, \ldots, m)$ satisfy a Lipschitz property with respective constants L and L_i, $(i = 1, 2, \ldots, m)$.

The algorithm of §2.2.2 can be used to handle constrained optimization with the following modifications and new tests.

Updating the incumbent value. The incumbent value f_{opt} should now be updated whenever a better value of the objective function has been found, but with the restriction that the corresponding point is feasible with respect to the constraints.

Checking redundant constraints. As the constraint functions are Lipschitz, piecewise constant upper bounding functions $\overline{G}_i^{r_j} = g_i(x^{r_j}) + \frac{1}{2}L_i D^{r_j}$ can be built in each cell C^{r_j}. Then the constraints g_i are such that $\overline{G}_i^{r_j} < 0$ are redundant and therefore can be omitted.

Infeasibility. Piecewise constant lower bounding functions $\underline{G}_i^{r_j} = g_i(x^{r_j}) - \frac{1}{2}L_i D^{r_j}$ can also be similarly built in each cell C^{r_j},. Then the subproblem can be discarded whenever it contains a constraint g_i such that $\underline{G}_i^{r_j} > 0$.

2.3. A decomposition-based algorithm (Dec). Consider again the problem

$$(\tilde{P}) \begin{cases} \text{maximize} & f(x, y), \\ \text{subject to :} & g_i(x, y) \leq 0, \qquad i = 1, 2, \ldots, m, \\ & x \in \mathbb{R}^n, \qquad y \in \mathbb{R}^{p-n}, \end{cases}$$

in which the functions are partially concave.

For any given value x^k, we define the restriction of problem $(\tilde{P})$ to variable y,

$$(\tilde{P}(x^k)) \begin{cases} \text{maximize} & f(x^k, y), \\ \text{subject to :} & g_i(x^k, y) \leq 0, \qquad i = 1, 2, \ldots, m, \\ & y \in \mathbb{R}^{p-n}, \end{cases}$$

and we denote by $\tilde{v}(x^k)$ its optimal value.

This last problem can be easily solved.

Apply the constrained Lipschitz optimization algorithm of §2.2.3 with objective function $\tilde{v}$. The evaluation of this new objective function at the successive points x^{r_j} is the result of a constrained concave maximization with respect to variable y. Efficient algorithms can then be applied; see, e.g., Avriel [2], Minoux [25], Wolfe [34]. The algorithm converges to the (global) maximum f^* as the sequence F^k_{opt} converges to f^*.

3. The statistical problem.

3.1. General theory of minimax estimation. Let us consider the classical statistical problem of making an inference about an unknown parameter λ, in particular, the estimation of λ, given an observation of $X \in \mathbf{X}$, where X is a random variable whose distribution depends on the parameter $\lambda \in \Lambda$, the parameter space. A solution consists of a nonrandomized estimator or decision procedure δ, which is a measurable function from the sample space $\mathbf{X} \to \Lambda$. Let $\mathcal{A}$ denote the space of all possible estimates. It will henceforth be assumed that $\mathcal{A}$ is convex. A risk function $R(\delta, \lambda)$ characterizes the performance of a decision procedure δ for each value of the parameter λ. The risk function is usually defined in terms of an underlying loss function $L(\delta, \lambda)$. A loss function maps $\mathcal{A} \times \Lambda \to \mathbb{R}^+ \cup \{0\}$ (where $\mathbb{R}^+$ is the positive real line) and defines the cost of estimating δ when λ is the true value of the parameter. To be able to confine our attention to nonrandomized estimators, it will be assumed that the loss function is convex in δ and that $\mathcal{A}$ and Λ are also convex (see, e.g., Lehmann [22]). The loss function will usually be assumed to equal $(\delta - \lambda)^2$; that is, the quadratic loss function or a normalized version of this function. The risk of an estimator δ when λ is true, $R(\delta, \lambda)$, is then the average loss incurred from using δ; that is,

$$R(\delta, \lambda) = E_\lambda L(\delta(X), \lambda). \tag{1}$$

An estimator δ^* is minimax for the above problem, if

$$\sup_{\lambda \in \Lambda} R(\delta^*, \lambda) \leq \sup_{\lambda \in \Lambda} R(\delta, \lambda) \quad \text{for all } \delta \epsilon D. \tag{2}$$

Minimax problems are often solved by considering the corresponding Bayes problems. A distribution or prior probability measure π is specified on the parameter space Λ, and a measure of the performance of a procedure δ is given by its Bayes risk

$$\tilde{r}_\Lambda(\delta, \pi) = \int_\Lambda R(\delta, \lambda) \pi(d\lambda). \tag{3}$$

δ_π is called the Bayes procedure with respect to the prior probability measure π if δ_π minimizes the Bayes risk. The minimum Bayes risk r of a distribution or prior probability measure π on Λ is defined as

$$r_\Lambda(\pi) = \tilde{r}_\Lambda(\delta_\pi, \pi). \tag{4}$$

Henceforth, $r_\Lambda(\pi)$ will be called the Bayes risk of π. A distribution or prior probability measure π^* is "least favorable" if its Bayes risk is greater than or equal to that of any other distribution; that is,

$$r_\Lambda(\pi^*) \geq r_\Lambda(\pi) \quad \text{for any distribution } \pi \text{ on } \Lambda. \tag{5}$$

Subject to the decision problem satisfying sufficient regularity conditions, a least favorable prior distribution exists and the corresponding Bayes procedure is minimax (see Wald [33], Kempthorne [20], or Brown [5]).

3.2. The bounded Poisson and normal mean problems. One of the problems considered here is that of estimating the mean of a Poisson distribution under the additional assumption that the true mean lies in a bounded interval of the form $[0, m]$ for $m > 0$. The Bayes risk $r_\Lambda(\pi)$ for a probability measure π on $\Lambda = [0, m]$ will be denoted by $r_m(\pi)$.

Let X be a random Poisson variable with mean λ; that is, the probability that $X = x$ denoted by $P(X = x) = e^{-\lambda}(\lambda^x/x!)$ for $x = 0, 1, 2, \ldots$ and $= 0$ otherwise. The normalized quadratic loss function used is defined by

$$L(\delta, \lambda) = \frac{(\delta - \lambda)^2}{\lambda}. \tag{6}$$

For $R(\delta, 0)$ to be finite, it follows that a minimax estimator must satisfy

$$\delta(0) = 0. \tag{7}$$

Henceforth, we shall only consider estimators satisfying (7). The risk of such an estimator can be written as

$$E_\lambda L(\delta(X), \lambda) = \lambda e^{-\lambda} + \sum_{x=1}^{\infty} (\delta(x) - \lambda)^2 \frac{\lambda^{x-1}}{x!} e^{-\lambda}. \tag{8}$$

Now, considering the dual Bayes problem, it follows from Ghosh's results [13] that a least favorable prior on $[0, m] = \Lambda$ will put mass on at most a finite number of points.

More precisely, as shown by Johnstone and MacGibbon [18], the "least favorable" prior $\pi^*(d\lambda)$ is of the form $\sum_{i=1}^{k} a_i \epsilon_{\{b_i m\}}$ $(k < \infty)$, where the Dirac measure

$$\begin{aligned} &\left\{ \begin{array}{ll} \epsilon_{\{b_i m\}}(x) = 1 & \text{if } x = b_i, \\ \epsilon_{\{b_i m\}}(x) = 0 & \text{otherwise,} \end{array} \right\} \quad \text{for } 0 \leq b_1 < \cdots < b_k = 1, \\ &\sum_{i=1}^{k} a_i = 1 \quad \text{with } a_i \geq 0 \quad \text{for all } k. \end{aligned} \tag{9}$$

The Bayes risk of π^* is defined as

$$r_P(m) = r_m(\pi^*) = \sum_{i=1}^{k} a_i R(\delta_{\pi^*}, b_i m). \tag{10}$$

Thus the statistical problem is reduced to the following global optimization problem.

Maximize $r_m(\pi^*)$ where π^* consists of a sequence of points $b_i m$ with weights a_i. Since the global unconstrained minimax risk on $[0, +\infty[$ is one (see Lehmann [22]), then $r_m(\pi^*) \leq 1$ for each m.

The general mathematical problem can be expressed as follows:

$$\text{Maximize } \theta(a, b) = \sum_{i=1}^{k} a_i g_i(a, b)$$

subject to:

$$\begin{aligned} &\sum_{i=1}^{k} a_i = 1, \\ &0 \leq a_i \leq 1, \qquad i = 1, 2, \ldots, k, \\ &0 \leq \theta \leq 1, \\ &0 \leq b_i \leq 1, \qquad i = 1, 2, \ldots, k, \end{aligned}$$

where

$$g_i(a, b) = b_i m e^{-b_i m} + \sum_{x=1}^{\infty} (b_i m - \delta_\pi(x))^2 \frac{(b_i m)^{x-1}}{x!} e^{-b_i m}$$

for $i = 1, 2, \ldots, k$;

$$\delta_\pi(x) = \frac{\sum_{i=1}^{k} a_i (b_i m)^x e^{-b_i m}}{\sum_{i=1}^{k} a_i (b_i m)^{x-1} e^{-b_i m}}, \qquad x \neq 0,$$

and $\delta_\pi(0) = 0$, $a = (a_1, a_2, \ldots, a_k)$, $b = (b_1, b_2, \ldots, b_k)$.

As shown in Johnstone and MacGibbon [18], the ratio of nonlinear to linear risk for this problem is

$$\mu_P(m) = \frac{m}{(1+m) r_P(m)}, \tag{11}$$

where $r_P(m)$ is the optimum value of $r_m(\pi^*)$ in the above optimization problem for fixed m and the Ibragimov–Hasminskii constant is

$$\mu_P^* = \sup_m \mu_P(m). \tag{12}$$

Since the optimization problem is nonconvex and nonlinear, global optimization techniques must be used to solve it exactly. However, due mainly to the difficulty of estimating the Lipschitz constants and the massive numerical calculations involved for this problem, global Lipschitz optimization algorithms seem to be practical only for a small number of variables. So as a first step, we will restrict ourselves to the solution of the problem when $k = 3$.

Johnstone and MacGibbon [18] gave upper bounds on $r_P(m)$; in particular, $r_P(m) \leq 1.249$ for all $m \geq 2.7$. The arguments they used to show that the two point prior is least favorable on $[0, m]$ for all $m \in (m_0, m_1]$ with $m_0 \simeq .57$ and $m_1 \simeq 1.27$ can be extended to show that on $[0, m]$ for all $m \in (m_1, m_2]$ with $m_2 \simeq 2.7$, the least favorable prior is supported on three points. On the interval $[0, m]$ with $m \in [0, 2.7]$, the minimax risk $r_P(m)$ will be calculated solving the following maximization problem.

For fixed m, find a_2, a_3, and b satisfying $0 \le a_2 \le 1$, $0 \le a_3 \le 1$, $0 \le a_2 + a_3 \le 1$, and $0 \le b \le 1$, such that $\pi^* = (1 - a_2 - a_3)\epsilon_{\{0\}} + a_2\epsilon_{\{bm\}} + a_3\epsilon_{\{m\}}$ is least favorable; that is, $r_m(\pi^*)$ is a global maximum over all prior probability measures π on $[0, m]$. (We can restrict our attention to measures of the form (9).) The problem can be expressed as follows.

Maximize $\theta(a_2, a_3, b) = (1 - a_2 - a_3)g_1(a_2, a_3, b) + a_2 g_2(a_2, a_3, b) + a_3 g_3(a_2, a_3, b)$, where

$$g_1(a_2, a_3 b) = \delta(1)^2 \quad \text{if } 1 - a_2 - a_3 > 0,$$
$$= 0 \quad \text{if not,}$$

$$g_2(a_2, a_3, b) = bm2^{-bm} + \sum_{x=1}^{\infty} \big(bm - \delta(x)\big)^2 \frac{(bm)^{x-1}}{x!} e^{-bm},$$

(13)
$$g_3(a_2, a_3, b) = me^{-m} + \sum_{x=1}^{\infty} \big(m - \delta(x)\big)^2 \frac{m^{x-1}}{x!} e^{-m},$$

$$\delta(x) = m \frac{a_2 b^x e^{-bm} + a_3 e^{-m}}{a_2 b^{x-1} e^{-bm} + a_3 e^{-m}} \quad \text{if } x \neq 0, 1,$$

$$\delta(1) = m \frac{a_2 be^{-bm} + a_3 e^{-m}}{a_2 e^{-bm} + a_3 e^{-m} + 1 - a_2 - a_3},$$

$$\delta(0) = 0,$$

subject to

$$0 \le a_2 \le 1,$$
$$0 \le a_3 \le 1,$$
$$0 \le b \le 1,$$
$$a_2 + a_3 \le 1.$$

This problem was solved using both the multivariate Lipschitz optimization algorithm (MLip) and the global optimization decomposition method (Dec) described in §2. This latter possibility arises from the fact that the objective function is concave with respect to the variables a_1, a_2, and a_3. This can be deduced from the strict concavity of r_m as a function of π (see Johnstone and MacGibbon [18]); that is

$$r_m(\alpha\pi + (1 - \alpha)\pi') > \alpha r_m(\pi) + (1 - \alpha) r_m(\pi') \quad \text{for all } \pi, \pi' \quad \text{and } 0 \le \alpha \le 1.$$

An analogous theory also holds for the normal bounded mean estimation problem on the interval $[-m, m]$ under quadratic loss, where the least favorable prior distribution π is now of the form

$$\sum_{i=1}^{k} a_i \varepsilon_{\{b_i m\}} + \sum_{i=1}^{k} a_i \varepsilon_{\{-b_i m\}} + \left(1 - 2\sum_{i=1}^{k} a_i\right) \varepsilon_{\{0\}},$$

where

(14)
$$0 \le b_i \le 1, \quad i = 1, 2, \ldots, k-1, \quad b_k = 1,$$
$$0 \le a_i \le 1, \quad i = 1, 2, \ldots, k,$$

and

$$0 \leq 1 - 2\sum_{i=1}^{k} a_i \leq 1.$$

The problem is to maximize the Bayes risk of π on $[-m, m]$ denoted by

$$\begin{aligned} r_m(\pi) &= \sum_{i=1}^{k} a_i R(\delta_\pi, b_i m) + \sum_{i=1}^{k} a_i R(\delta_\pi, -b_i m) + \left(1 - 2\sum_{i=1}^{k} a_i\right) R(\delta_\pi, 0) \\ &= 2\sum_{i=1}^{k} a_i R(\delta_\pi, b_i m) + \left(1 - 2\sum_{i=1}^{k} a_i\right) R(\delta_\pi, 0), \end{aligned}$$

where

$$R(\delta_\pi, \theta) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \big(\delta_\pi(x) - \theta\big)^2 e^{-1/2(x-\theta)^2}\, dx$$

and

$$\delta_\pi(x) = \frac{\displaystyle\sum_{i=1}^{k} a_i (b_i m) e^{-\frac{1}{2}(x-b_i m)^2} + \sum_{i=1}^{k} a_i (-b_i m) e^{-\frac{1}{2}(x+b_i m)^2}}{\left(1 - 2\displaystyle\sum_{i=1}^{k} a_i\right) e^{-\frac{1}{2}x^2} + \displaystyle\sum_{i=1}^{k} a_i e^{-\frac{1}{2}(x-b_i m)^2} + \sum_{i=1}^{k} a_i e^{-\frac{1}{2}(x+b_i m)^2}}. \tag{15}$$

If we let $r_N(m) = \max_\pi r_m(\pi)$ denote the nonlinear minimax risk, then both $r_N(m)$ and the ratio of linear to nonlinear risk, $\mu_N(m)$, as defined in the Introduction will be calculated. For this problem we shall take advantage of the calculations of Donoho, Liu, and MacGibbon [7]; i.e., that for $m \geq 2$, $\mu_N(m) \leq 1.242$ with four-digit accuracy and the fact that for $m \leq 2$ a three-point prior suffices (see Casella and Strawderman [6]). Thus our problem is reduced to the maximization problem for $k = 1$ on this interval. For this univariate problem, the fact that $r_m(\pi)$ is a concave function of a_1 follows from the concavity of r_m as a function of π (see Bickel [4]); thus Fibonacci's method can be used to solve it.

4. Numerical results. We now present the numerical results obtained when applying both algorithms (MLip) and (Dec) of §2 to minimax risk evaluation for a bounded Poisson mean. The algorithms were implemented in Fortran77 and run on a Sun Sparc 4/330-32 (16 mips). The results are first interpreted in terms of computational efficiency leading to a comparison of the two algorithms. Answers are then provided in terms of the evaluation of the minimax risk for a bounded Poisson mean.

4.1. Comparison between the two algorithms. The evaluation of the performance of both algorithms are based on the following parameters:

- iter for the number of function evaluations,
- cpu for the computation time in seconds,
- memo for the virtual memory size in Kbytes, and
- ε for the tolerance required on the value of the objective function.

The performances of both algorithms, in terms of computing time (cpu) and number of function evaluations (iter), for a tolerance of $\varepsilon = 10^{-2}$, are compared in Table 1.

As foreseen, the number of evaluations iter and the cpu processing time are closely related, which seems to indicate that iter is an accurate indicator of the algorithm performance. A single evaluation costs less in algorithm (MLip) ($\simeq$ 1/30 s) than in algorithm (Dec) ($\simeq$ 1/10 s), which

TABLE 1

Comparison of algorithm (MLip) *and algorithm* (Dec) *for the minimax risk* $r_P(m)$ *of a bounded Poisson mean on* $[0, m]$ *under normalized loss.*

Algorithm (MLip): with precision $\varepsilon = 10^{-2}$

m	iter	cpu	a_1	a_2	a_3	b	$r_P(m)$
1.0	506826	1999	.16790	.01193	.82017	.99807	.41454
1.1	473495	1869	.18336	.01193	.80471	.99807	.43151
1.2	457899	1825	.19497	.01193	.79310	.99810	.44658
1.3	457113	1832	.00161	.20529	.79310	.01387	.45977
1.4	404959	1625	.00354	.23236	.76410	.05254	.47190
1.5	320025	1302	.00161	.26330	.73509	.08348	.48336
1.6	251943	1029	.00161	.28650	.71189	.10668	.49424
1.7	199425	823	.00161	.30584	.69255	.12215	.50462
1.8	157098	651	.00159	.32131	.67710	.12988	.51450
1.9	131882	544	.00160	.34451	.65389	.14535	.52396
2.0	116641	489	.00160	.35998	.63842	.15308	.53298
3.0	146148	621	.05188	.42185	.52627	.23043	.60253
4.0	244964	1040	.09055	.46149	.44796	.27974	.65019

Algorithm (Dec): with precision $\varepsilon = 10^{-2}$

m	iter	cpu	a_1	a_2	a_3	b	$r_P(m)$
1.0	3505	3	.16906	.00000	.83094	.10281	.41454
1.1	4515	4	.18333	.00000	.81666	.06414	.43151
1.2	4567	4	.19331	.00000	.80669	.19562	.44659
1.3	8769	7	.19979	.00014	.80007	.04867	.45972
1.4	21969	17	.01640	.22832	.75528	.07187	.47178
1.5	21337	17	.00043	.25951	.74005	.07961	.48336
1.6	20619	17	.00000	.28659	.71341	.10281	.49426
1.7	17897	15	.00000	.30479	.69521	.11828	.50463
1.8	15853	13	.00000	.32503	.67497	.13375	.51453
1.9	15613	13	.00000	.34078	.65922	.14148	.52397
2.0	16299	13	.00000	.35605	.64394	.14922	.53299
3.0	12031	10	.05186	.42218	.52596	.23043	.60253
4.0	10863	9	.09144	.46112	.44744	.28070	.65019

is easily explained by the relative complexity of algorithm (Dec), but the convergence (iter) is faster in the case of algorithm (Dec). It follows that algorithm (Dec) finds the solution faster. Indeed the speedup (i.e., the ratio of the cpu needed by algorithm (MLip) to the cpu needed by algorithm (Dec) to obtain results with the same tolerance ε) varies from 650 for $m = 1.0$ to 50 for $m = 3.0$ with an average value of 200.

The second improvement due to the decomposition method used in algorithm (Dec), is the small amount of memory it needs, whatever tolerance is asked, (memo $\simeq$ 200 Kbytes); whereas in algorithm (MLip), the amount of memory increases drastically with the tolerance (memo $\simeq$ 400 Kbytes for $\varepsilon = 10^{-1}$, memo $\simeq$ 7000 Kbytes for $\varepsilon = 10^{-2}$).

4.2. Results of the minimax risk evaluation. The algorithm was run for several values of the parameter m. The variables a_2, a_3, b are bounded by $[\varepsilon, 1]$ where $\varepsilon = 10^{-2}$. When one of those variables is near to ε, we assume it is zero. This occurs for a_2 from $m = 1$ to $m = 1.27$, where it was already known theoretically that the two-point prior is sufficient (see Johnstone and MacGibbon [18]). It is confirmed by the algorithm that yields very small values of a_2. Hence the variable b is not really significant until $m \simeq 1.3$ or more, where the three-point prior clearly appears as a_2 starts to increase away from zero. At this stage of the study, we cannot be sure of the largest value of m for which the three-point prior suffices. Other computational methods suggest that this value is near 4.5 (see, e.g., Johnstone and MacGibbon [18]).

The infinite sum for the risk in (8) was limited to ten terms. This gave a bound of 10^{-9} on the remaining terms for the range of parameters considered. (See Appendix 1.) The method of calculating the Lipschitz constants is also given there.

The numerical results yielding a precision of $\varepsilon = 10^{-4}$ for the bounded Poisson mean problem using the decomposition method are presented in Table 2.

The numerical results in Table 2, combined with $\mu_P(m) \leq 1.249$ for $m \geq 2.7$, indicate a largest value $\mu_P(2.05) = 1.250826$. From (13) we deduce

$$\frac{d\mu_P(m)}{dr_P(m)} = \frac{-m}{(1+m)r_P^2(m)} \leq 0.$$

This implies a precision on μ_P,

$$\Delta\mu_P \simeq \frac{m}{(1+m)r_P^2(m)}\Delta r_P,$$

where $\Delta r_P = \varepsilon = 10^{-4}$ is the precision for r_P. For $m = 2.05$, this yields a value of $\Delta\mu_P \simeq 0.510^{-4}$. Finally, as μ_P is a decreasing function of r_P, our evaluation of the Ibragimov–Hasminskii constant μ_P^* for the Poisson problem on $[0, m]$, is

$$\underbrace{\mu_P(2.05) - \Delta\mu_P}_{=1.250776} \leq \mu_P^* \leq \underbrace{\mu_P(2.05)}_{=1.250826}. \tag{16}$$

The optimization problem for the bounded normal mean on $[-m, m]$ is one of concave maximization and hence the Fibonacci method was used.

The error analysis was carried out as follows. (See Appendix 2 for details.)

Setting $k = 1$ in (16) and (17) and letting $\alpha = 1 - 2a_1$, our aim is to maximize

$$r(\alpha) = (1-\alpha)R(\delta_\alpha(x), m) + \alpha R(\delta_\alpha(x), 0),$$

where

TABLE 2

Minimax risk ($r_P(m)$) *and ratio of linear to nonlinear minimax risk* ($\mu_P(m)$) *for the Poisson mean on* $[0, m]$ *under normalized loss with precision* $\varepsilon = 10^{-4}$.

Decomposition method: with precision $\varepsilon = 10^{-4}$								
m	iter	cpu	a_1	a_2	a_3	b	$r_P(m)$	$\mu_P(m)$
0.2	14014	9	.00000	.25000	.75000	.99976	.16374	1.01787
0.4	14798	11	.00000	.25000	.75000	.99976	.26812	1.06562
0.6	34898	26	.02411	.00000	.97589	.13375	.32965	1.13757
0.8	94362	72	.12086	.00000	.87914	.16082	.37525	1.18439
1.0	91895	70	.16909	.00000	.83091	.14535	.41454	1.20616
1.1	106574	82	.18336	.00000	.81664	.40059	.43151	1.21390
1.2	111454	85	.19332	.00000	.80668	.06027	.44659	1.22138
1.3	439674	335	.00000	.20793	.79270	.01387	.45978	1.22932
1.4	1670234	1271	.03320	.20632	.76047	.06819	.47183	1.23632
1.5	985418	759	.00000	.26201	.73799	.08039	.48337	1.24128
1.6	716662	546	.00000	.28603	.71397	.10239	.49426	1.24506
1.7	572398	440	.00000	.30777	.69223	.11943	.50464	1.24768
1.8	479898	368	.00000	.32728	.67272	.13272	.51454	1.24938
1.9	413114	319	.00000	.34470	.65529	.14318	.52399	1.25035
2.0	369834	286	.00000	.36054	.63946	.15188	.53300	1.25078
2.01	366131	284	.00000	.36179	.63820	.15236	.53388	1.250794
2.02	363967	283	.00000	.36324	.63677	.15308	.53475	1.250816
2.03	360987	280	.00000	.36468	.63532	.15381	.53563	1.250802
2.04	353707	279	.00000	.36631	.63369	.15478	.53649	1.250820
2.05	353175	275	.00000	.36473	.63527	.15496	.53735	1.250826
2.06	348887	279	.00000	.36984	.63016	.15707	.53821	1.250818
2.07	343111	275	.00000	.37031	.62969	.15659	.53906	1.250820
2.08	340619	270	.00000	.37172	.62827	.15731	.53991	1.250810
2.09	339463	267	.00000	.37295	.62705	.15780	.54076	1.250787
2.1	334555	265	.00000	.37426	.62574	.15840	.54160	1.25077
2.2	310811	250	.00000	.38482	.61518	.16402	.54978	1.25050
2.3	260371	212	.00000	.39653	.60347	.16873	.55757	1.25001
2.4	307003	246	.00000	.40708	.59292	.17278	.56495	1.24946
2.5	330391	264	.00000	.41659	.58341	.17632	.57195	1.24886
2.6	415323	333	.00000	.42578	.57422	.17946	.57856	1.24831

$$\delta_\alpha(x) = \frac{(1-\alpha)m\tanh(mx)}{(1-\alpha)+\alpha e^{\frac{m^2}{2}}\cosh(mx)},$$

$$R(\delta_\alpha(x), m) = \frac{1-\alpha}{\sqrt{2\pi}}\int_{-\infty}^{\infty} f(x)dx = I_\alpha,$$

$$R(\delta_\alpha(x), 0) = \frac{1-\alpha}{\sqrt{2\pi}}\int_{-\infty}^{\infty} g(x)dx = J_\alpha,$$

with

$$f(x) = [\delta_\alpha(x) - m]^2 e^{-\frac{1}{2}(x-m)^2} \text{ and } g(x) = [\delta_\alpha(x)]^2 e^{-\frac{1}{2}x^2}.$$

The errors due to the approximation of I_α by $\tilde{I}_\alpha = (1-\alpha/\sqrt{2\pi})\int_{-M}^{M} f(x)dx$ and J_α by $\tilde{J}_\alpha = (1-\alpha/\sqrt{2\pi})\int_{-M}^{M} g(x)dx$ can be bounded by

$$9m^2(1-\alpha)e^{-\frac{(M-m)^2}{4}} + \alpha m^2 e^{-M^2/4}. \tag{17}$$

Integrating $\tilde{I}_\alpha$ and $\tilde{J}_\alpha$ by the trapezoid rule with n subdivisions, the errors can be expressed as

$$\frac{M^3}{3n^2}\sup_{|x|\le M}|f''(x)|\frac{(1-\alpha)}{\sqrt{2\pi}} + \frac{M^3}{3n^2}\sup_{|x|\le M}|g''(x)|\frac{\alpha}{\sqrt{2\pi}}. \tag{18}$$

With $M = 10$, the first error given by (17) $\simeq 2 \times 10^{-10}$. With $n = 100,000$, the second error given by (18) yielded a maximum value of 2×10^{-4} for the values of m of interest. The error arising from the use of the Fibonacci method is of order 10^{-10}.

Table 3, combined with the upper bound of $\mu_N(m) \le 1.242$ for $m \ge 2$ obtained by Donoho, Liu, and MacGibbon [7], shows a highest value $\mu_N(1.6) = 1.246609$. A reasoning similar to that for μ_P^* yields an Ibragimov–Hasminskii constant bounded by

$$\underbrace{\mu_N(1.6) - \Delta\mu_N}_{=1.246509} \le \mu_N^* \le \underbrace{\mu_N(1.6)}_{=1.246609}. \tag{19}$$

This agrees with the previous upper bound of Donoho, Liu, and MacGibbon [7] for μ_N^* of 1.2497 and the approximation of Feldman and Brown [10] of 1.2465.

Clearly, (16) and (19) lead us to conclude that the Ibragimov–Hasminskii constants for the normal and the Poisson problems are different.

5. Conclusion. The experimental results of the previous section show that minimax risk evaluation problems on bounded parameter spaces of small size can be solved optimally using multivariate global optimization techniques, and that this, combined with upper bounds for larger values of m, suffices for the calculation of Ibragimov–Hasminskii constants. However, one should seek heuristics to solve problems for larger values of m.

The success of the decomposition method in improving the efficiency of global optimization algorithms leads us to believe that this technique could be applied to many other optimization problems in statistics. In particular, it could be extremely useful in maximum likelihood estimation in multiparameter families where the problem could be decomposed into two parts: for some parameters, standard convex optimization methods would suffice, while for others, a global optimization method would be necessary.

TABLE 3

Minimax risk ($r_N(m)$) *and ratio of linear to nonlinear minimax risk* ($\mu_N(m)$) *for a bounded normal mean on* $[-m, m]$ *under quadratic loss.*

Fibonacci method: with 100, 000 iterations					
m	ε	cpu	α	$r_N(m)$	$\mu_N(m))$
1.0	10^{-5}	1802	.000000	.449599	1.112100
1.1	4×10^{-5}	1803	.049675	.471455	1.161323
1.2	8×10^{-5}	1804	.143335	.492132	1.199199
1.3	10^{-4}	1805	.214465	.513398	1.223714
1.4	10^{-4}	1806	.269197	.534880	1.237963
1.5	10^{-4}	1806	.311714	.556172	1.244776
1.55	2×10^{-4}	1818	.329346	.566614	1.246173
1.56	2×10^{-4}	1835	.332621	.568679	1.246329
1.57	2×10^{-4}	1827	.335818	.570735	1.246449
1.58	2×10^{-4}	1839	.338938	.572782	1.246534
1.59	2×10^{-4}	1831	.341983	.574819	1.246586
1.60	2×10^{-4}	1808	.344955	.576847	1.246609
1.61	2×10^{-4}	1807	.347857	.578863	1.246602
1.62	2×10^{-4}	1807	.350689	.580869	1.246569
1.63	2×10^{-4}	1807	.353453	.582863	1.246511
1.64	2×10^{-4}	1807	.356152	.584846	1.246429
1.65	2×10^{-4}	1807	.358786	.586816	1.246327
1.7	2×10^{-4}	1836	.371047	.596466	1.245553
1.8	2×10^{-4}	1873	.391568	.614601	1.243328
1.9	2×10^{-4}	1916	.407716	.630838	1.241333
2.0	2×10^{-4}	1830	.420415	.644801	1.240693

In the particular problem of bounded parameter minimax risk evaluation, we have been able to conclude here that the Ibragimov–Hasminskii constants are different for the normal problem under quadratic loss and the Poisson problem under normalized quadratic loss. This exact minimax result should be contrasted with the asymptotic minimax result obtained by Johnstone and MacGibbon [19], where by means of the polydisc transform, an equivalence was obtained between the problems of Poisson estimation on bounded $2p$-dimensional domains under information normalized quadratic loss and $2p$-dimensional normal estimation under quadratic loss. More work is indicated both in the calculation of Ibragimov–Hasminskii constants for different bounded parameter estimation problems and in asymptotic minimax theory to truly understand the relationship between these problems.

Appendix 1. Specification of the Lipschitz constant for the Poisson estimation problem. After some calculations, the objective function θ used in problem (15) can be reformulated as follows:

$$\theta(a_2, a_3, b) = (a_2 b e^{-bm} + a_3 e^{-m}) m(1 - \delta(1)) + m^2(a_2 b^2 e^{-bm} + a_3 e^{-m}) + \sum_{x=2}^{\infty} \underbrace{\frac{a_2 a_3 (1-b)^2}{a_3 + a_2 b^{x-1} e^{(1-b)m}} \frac{b^{x-1} m^{x+1}}{x!} e^{-bm}}_{\Phi(a_2, a_3, b)}.$$

Considering a rectangular domain of $\mathbb{R}^3$ defined as $[\underline{a}_2, \overline{a}_2] \times [\underline{a}_3, \overline{a}_3] \times [\underline{b}, \overline{b}]$, a Lipschitz constant L for the function θ can easily be deduced from the variations of θ. Some analytical computations provide the following bounds on the partial derivatives of θ:

$$\left|\frac{\partial \theta}{\partial a_2}\right| \leq m^2 \overline{b}^2 e^{-\overline{b}m} + m^2 \frac{(\overline{a}_2 \tilde{b} e^{-\tilde{b}m} + \overline{a}_3 e^{-m})}{(\tilde{a_1} + \overline{a}_2 e^{-\underline{b}m} + \overline{a}_3 e^{-m})^2} M_1 + \sum_{x=2}^{\infty} \frac{\partial \Phi}{\partial a_2}(\underline{a}_2, \overline{a}_3, \tilde{b}),$$

$$\left|\frac{\partial \theta}{\partial a_3}\right| \leq m^2 e^{-m} + m^2 \frac{(\overline{a}_2 \tilde{b} e^{-\tilde{b}m} + \overline{a}_3 e^{-m})}{(\tilde{a_1} + \overline{a}_2 e^{-\underline{b}m} + \overline{a}_3 e^{-m})^2} M_2 + \sum_{x=2}^{\infty} \frac{\partial \Phi}{\partial a_2}(\overline{a}_2, \underline{a}_3, \tilde{b}),$$

$$\left|\frac{\partial \theta}{\partial b}\right| \leq m \overline{a}_2 e^{-\underline{b}m} sup\{(1 - bm + 2bm^2 - b^2 m^3)\} + m^2 \frac{(\overline{a}_2 \tilde{b} e^{-\tilde{b}m} + \overline{a}_3 e^{-m})}{(\tilde{a_1} + \overline{a}_2 e^{-\underline{b}m} + \overline{a}_3 e^{-m})^2} M_3 + \sum_{x=2}^{\infty} \frac{\overline{a}_2 \overline{a}_3 \overline{b}^{x-2} (1 - \underline{b}) m^{x+1} e^{-\underline{b}m}}{(\underline{a}_3 + \underline{a}_2 \tilde{b}^{x-1} e^{(1-\tilde{b})m}) x!} M_4,$$

where

$$\begin{aligned}
\tilde{b} &= \min\{\underline{b}, \overline{b}, 1/m\},\\
\tilde{a_1} &= \min\{0, 1 - \overline{a}_2 - \overline{a}_3\},\\
M_1 &= \max\{\overline{a}_3 e^{-m}(1 - e^{-\overline{b}m}) + 2\tilde{b} e^{-\tilde{b}m}, \overline{a}_2 b e^{-\tilde{b}m}(1 - e^{-\overline{b}m})\\
&\qquad + 2\overline{a}_3 \tilde{b} e^{-\tilde{b}m}(1 - e^{-m}) - \underline{a}_3 e^{-m}(1 - e^{-\underline{b}m})\},\\
M_2 &= \max\{\overline{a}_2 \tilde{b} e^{-\tilde{b}m}(1 - e^{-m}) + 2e^{-m} - 2\underline{a}_2 e^{-m}(1 - e^{-\underline{b}m}) - \underline{a}_3 e^{-m}(1 - e^{-m}),\\
&\qquad 2\overline{a}_2 e^{-m}(1 - e^{-\overline{b}m}) + \overline{a}_3 e^{-m}(1 - e^{-m}) - 2e^{-m}\}\\
M_3 &= \max\{2(1 - \underline{a}_2 - \underline{a}_3 + \underline{a}_2 e^{-\underline{b}m} + \underline{a}_3 e^{-m}) + m(\overline{a}_2 \tilde{b} e^{-\tilde{b}m} + \overline{a}_3 e^{-m})\\
&\qquad - m\underline{b}(\tilde{a_1} + \overline{a}_2 e^{-\underline{b}m} + \overline{a}_3 e^{-m}), m\overline{b}(1 - \underline{a}_2 - \underline{a}_3 + \underline{a}_2 e^{-\underline{b}m} + \underline{a}_3 e^{-m})\\
&\qquad - 2(\tilde{a_1} + \overline{a}_2 e^{-\underline{b}m} + \overline{a}_3 e^{-m}) - m\underline{a}_3 e^{-m}\},\\
M_4 &\max\{\overline{a}_3(x+1)(1 - \underline{b}) - \underline{a}_3(m(1 - \overline{b})\underline{b} + \underline{b} + 1) - 2\underline{a}_2 \underline{b}^{x+1} e^{(1-\overline{b})m},\\
&\qquad \overline{a}_3(m(1 - \underline{b})\overline{b} + \overline{b} + 1) + 2\overline{a}_2 \overline{b}^{x+1} e^{(1-\underline{b})m} - \underline{a}_3(x+1)(1 - \overline{b})\}
\end{aligned}$$

Appendix 2. Normal bounded mean estimation. Considering the case $k = 1$, using (14) (i.e., $b_1 = 1$) and letting $\alpha = 1 - 2a_1$, the Bayes risk can be rewritten in the following way:

$$r_m(\pi) = \overbrace{\frac{(1-\alpha)}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} f(x)\, dx}^{I_\alpha} + \overbrace{\frac{\alpha}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} g(x)\, dx}^{J_\alpha},$$

where

$$f(x) = \left[\delta_\alpha(x) - m\right]^2 e^{-\frac{1}{2}(x-m)^2}, \text{ and let } g(x) = \left[\delta_\alpha(x)\right]^2 e^{-\frac{1}{2}x^2}.$$

Since π now depends only on α, we will henceforth denote the Bayes risk by $r(\alpha)$.

Step 1. *First approximation.* Approximate I_α by

$$\tilde{I}_\alpha = \frac{(1-\alpha)}{\sqrt{2\pi}} \int_{-M}^{M} f(x)\,dx$$

and J_α by

$$\tilde{J}_\alpha = \frac{\alpha}{\sqrt{2\pi}} \int_{-M}^{M} g(x)\,dx.$$

Let $\tilde{M}_I = |I_\alpha - \tilde{I}_\alpha|$ and $\tilde{M}_J = |J_\alpha - \tilde{J}_\alpha|$ be the errors due to this approximation.

We obtain the following bounds on these errors:

$$\tilde{M}_I \le 9(1-\alpha)m^2 e^{-\frac{(M-m)^2}{4}},$$
$$\tilde{M}_J \le \alpha m^2 e^{-\frac{M^2}{4}}.$$

Step 2. *Second approximation.* Approximate now $\tilde{I}_\alpha$ and $\tilde{J}_\alpha$, respectively, by $\hat{I}_\alpha$ and $\hat{J}_\alpha$, where

$$\hat{I}_\alpha = \frac{(1-\alpha)}{\sqrt{2\pi}} \frac{M}{n} \left[f(M) + f(-M) + 2\sum_{i=1}^{n-1} f\left(-M + \frac{2M}{n} i\right)\right],$$
$$\hat{J}_\alpha = \frac{(1-\alpha)}{\sqrt{2\pi}} \frac{M}{n} \left[g(M) + g(-M) + 2\sum_{i=1}^{n-1} g\left(-M + \frac{2M}{n} i\right)\right].$$

Let

$$\hat{M}_I = |\tilde{I}_\alpha - \hat{I}_\alpha| = \frac{M^3}{3n^2} \sup_{-M\le x\le M} |f''(x)| \times \frac{(1-\alpha)}{\sqrt{2\pi}},$$
$$\hat{M}_J = |\tilde{J}_\alpha - \hat{J}_\alpha| = \frac{M^3}{3n^2} \sup_{-M\le x\le M} |g''(x)| \times \frac{\alpha}{\sqrt{2\pi}},$$

be the errors due to this approximation. We then have

$$\sup_{-M\le x\le M} |f''(x)| \le \frac{m}{(A+B)^2} \left[\left[\max\left\{2Am, 2Am\frac{A+B\cosh(mM)}{A\cosh(mM)+B}\right\} + (M+m)(A+B)\right]^2 + 2m^2 A\sinh(mM)|B-A| + Am \max\left\{2Am, 2Am\frac{A+B\cosh(mM)}{A\cosh(mM)+B}\right\} + A + B\right],$$

$$\sup_{-M\le x\le M} |g''(x)| \le \frac{Am}{(A+B)^2} \left[\left[\max\left\{2m, 2m\frac{A+B\cosh(mM)}{A\cosh(mM)+B}\right\} + M\sinh^2(mM)\right]^2 + 2m^2\sinh(mM)\frac{|B-A|}{(A+B)} + m\cosh(mM)\max\left\{2m, 2m\frac{A+B\cosh(mM)}{A\cosh(mM)+B}\right\} + \sinh^2(mM)\right],$$

where $A = 1 - \alpha$ and $B = \alpha e^{m^2/2}$.

Step 3. Maximizing $\hat{r}_\alpha = \hat{I}_\alpha + \hat{J}_\alpha$ by the Fibonacci method and letting $r_N(m)$ denote the errorless optimum, that is, the errorless minimax risk at m, we have an approximation of the optimum with an error ε, i.e.,

$$|\max_{0\le\alpha\le1}(\hat{r}_\alpha) - r_N(m)| < \varepsilon.$$

Let

$$r_{\max} = \max_{0\le\alpha\le1}(\hat{r}_\alpha).$$

Step 4. The cumulative error can be bounded as follows; we have

$$|I_\alpha - \hat{I}_\alpha| = |I_\alpha - \tilde{I}_\alpha + \tilde{I}_\alpha - \hat{I}_\alpha| \le |I_\alpha - \tilde{I}_\alpha| + |\tilde{I}_\alpha - \hat{I}_\alpha| \le \tilde{M}_I + \hat{M}_I,$$
$$|J_\alpha - \hat{J}_\alpha| \le \tilde{M}_J + \hat{M}_J.$$

Since $r(\alpha) = I_\alpha + J_\alpha$, then $|r(\alpha) - \hat{r}(\alpha)| \le \tilde{M}_I + \tilde{M}_J + \hat{M}_I + \hat{M}_J$, where $\tilde{M}_I$, $\tilde{M}_J$, $\hat{M}_I$, and $\hat{M}_J$ are functions of α and the relation holds true for all α, particularly for the $\alpha_{\max}$ obtained by the Fibonacci method (with an ε_α error, i.e., $|\alpha_{\max} - \alpha^*| \le \varepsilon_\alpha$). Hence $|r(\alpha_{\max}) - r_{\max}| \le \tilde{M}_I(\alpha_{\max}) + \tilde{M}_J(\alpha_{\max}) + \hat{M}_I(\alpha_{\max}) + \hat{M}_J(\alpha_{\max})$, which implies that $|r(\alpha_{\max}) - r_{\max}| \le \tilde{M}_I(\alpha_{\max}) + \tilde{M}_J(\alpha_{\max}) + \hat{M}_I(\alpha_{\max}) + \hat{M}_J(\alpha_{\max}) + \varepsilon$.

Note. Details of the computations outlined in the Appendices can be found in E. Gourdin, B. Jaumard, and B. MacGibbon, *Global optimization decomposition methods for bounded parameter minimax risk evaluation*, Cahiers du GERAD G90-48 (ISSN 0711-2440), Montréal, 1990 (revised version 1992).

Acknowledgments. The authors would like to express their thanks to the referees for their useful comments and suggestions.

REFERENCES

[1] M. D. Atkinson, J.-R. Sack, N. Santoro, and T. Strothotte, *Min-Max Heaps and Generalized Priority Queues*, Comm. ACM, 29 (1986), pp. 997–1000.

[2] M. Avriel, *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[3] J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*, 2nd ed., Springer-Verlag, New York, 1985.

[4] P. Bickel, *Minimax estimation of the mean of a normal distribution when the parameter space is restricted*, Ann. Statist., 9 (1981), pp. 1301–1309.

[5] L. D. Brown, *Statistical Decision Theory*, mimeographed notes, Cornell University, Ithaca, NY, 1979.

[6] G. Casella and W. Strawderman, *Estimating a bounded normal mean*, Ann. Statist., 9 (1981), pp. 868–876.

[7] D. L. Donoho, R. C. Liu, and B. MacGibbon, *Minimax risk over hyperrectangles with implications*, Ann. Statist., 18 (1990), pp. 1416–1437.

[8] R. J. Duffin, E. L. Peterson, and C. Zener, *Geometric Programming Theory and Application*, Wiley, New York, 1967.

[9] J. G. Ecker, *Geometric Programming: Methods, Computations and Applications*, SIAM Rev., 22 (1980), pp. 338–362.

[10] I. Feldman and L. D. Brown, *The Minimax Risk for Estimating a Bounded Normal Mean*, submitted, 1989.

[11] T. S. Ferguson, *Mathematical Statistics, A Decision Theoretic Approach*, Academic Press, New York, 1967.

[12] P. Gill, W. Murray, M. Saunders, and M. Wright, *User's Guide for NPSOL (Version 4.0): A FORTRAN Package for Non-Linear Programming*, Tech. Report SOL 86–2, Systems Optimization Laboratory, Stanford Univ., Stanford, CA, 1986.

[13] M. N. Ghosh, *Uniform approximation of minimax point estimates*, Ann. Math. Statist., 35 (1964), pp. 1031–1047.

[14] E. GOURDIN, *Global Optimization Algorithms for the Construction of Least Favorable Priors and Minimax Estimations*, engineering undergraduate thesis, Institut d'Informatique d'Entreprise, Evry, France, 1989.

[15] P. HANSEN, B. JAUMARD, AND S-H. LU, *On the number of iterations of Piyavskii's global optimization algorithm*, Math, Oper. Res., 16 (1991), pp. 334–350.

[16] R. HORST AND H. TUY, *Global Optimization: Deterministic Approaches*, Springer-Verlag, Berlin, New York, 1990.

[17] I. A. IBRAGIMOV AND R. Z. HASMINSKII, *Non parametric estimation of the value of a linear functional in Gaussian white noise*, Theory Probab. Appl., 29 (1984), pp. 1–32.

[18] I. M. JOHNSTONE AND K. B. MACGIBBON, *Minimax estimation of a constrained Poisson vector*, Ann. Statist., 22 (1992), pp. 807–831.

[19] I. M. JOHNSTONE AND K. B. MACGIBBON, *Asymptotically minimax estimation of a constrained Poisson vector via polydisc transforms*, Annales de l'Institut Henri Poincaré, Série B, 29 (1993).

[20] P. J. KEMPTHORNE, *Numerical specification of discrete least favorable prior distributions*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 171–184.

[21] J. KIEFER, *Sequential Minimax Search for a Maximum*, Proc. Amer. Math. Soc., 4 (1953), pp. 502–506.

[22] E. L. LEHMANN, *Theory of Point Estimation*, Wiley, New York, 1983.

[23] C. C. MEEWELA AND D. Q. MAYNE, *An algorithm for global optimization of Lipschitz continuous functions*, J. Optim. Theory Appl., 57 (1988), pp. 307–322.

[24] ———, *Efficient domain partitioning algorithms for global optimization of rational and Lipschitz continuous functions*, J. Optim. Theory Appl., 61 (1989), pp. 247–270.

[25] M. MINOUX, *Programmation Mathématique, Théorie et Algorithmes*, Tome 1, Dunod, Paris, 1983.

[26] R. H. MLADINEO, *An algorithm for finding the global maximum of a multimodal, multivariate function*, Math. Programming, 34 (1986), pp. 188–200.

[27] M. S. PINSKER, *Optimal filtering of square integrable signals in Gaussian white noise*, Problemy Peredachi Informatsii, 16(2), pp. 52–68; Also, Problems Inform. Transmission, (1980), pp. 120–133.

[28] S. A. PIYAVSKII, *An algorithm for finding the absolute minimum of a function*, Theory Optimal Solutions, 2 (1967), Kiev, IK AN USSR, pp. 13–24. (In Russian.)

[29] ———, *An Algorithm for Finding the Absolute Extremum of a Function*, Zh. vȳchisl Mat. mat. Fiz., 12(4) (1972), pp. 888–896; also USSR Comput. Math. Math. Phys., 12 (1972), pp. 57–67.

[30] H. RATSCHEK AND J. ROKNE, *New Computer Methods for Global Optimization*, Wiley, New York, 1988.

[31] J. B. ROSEN, *Global minimization of a linearly constrained concave function by partition of feasible domain*, Math. Oper, Res., 8 (1983), pp. 215–230.

[32] B. O. SHUBERT, *A sequential method seeking the global maximum of a function*, SIAM J. Numer. Anal., 9 (1972), pp. 379–388.

[33] A. WALD, *Statistical Decision Functions*, John Wiley, New York, 1950.

[34] P. WOLFE, *Methods of Nonlinear Programming*, in Nonlinear Programming, J. Abadie, ed., North-Holland, Amsterdam, pp. 67–86, 1967.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 36–61, January 1994

003

SMALL-SAMPLE STATISTICAL CONDITION ESTIMATES FOR GENERAL MATRIX FUNCTIONS*

C. S. KENNEY† AND A. J. LAUB†

Abstract. A new condition estimation procedure for general matrix functions is presented that accurately gauges sensitivity by measuring the effect of random perturbations at the point of evaluation. In this procedure the number of extra function evaluations used to evaluate the condition estimate determines the order of the estimate. That is, the probability that the estimate is off by a given factor is inversely proportional to the factor raised to the order of the method. The "transpose-free" nature of this new method allows it to be applied to a broad range of problems in which the function maps between spaces of different dimensions. This is in sharp contrast to the more common power method condition estimation procedure that is limited, in the usual case where the Fréchet derivative is known only implicitly, to maps between spaces of equal dimension. A group of examples illustrates the flexibility of the new estimation procedure in handling a variety of problems and types of sensitivity estimates, such as mixed and componentwise condition estimates.

Key words. conditioning, matrix functions

AMS subject classifications. 65F35, 65F30, 15A12

1. Introduction.

If a scalar real-valued function f has a large derivative at $x \in \mathbb{R}$, then the relationship

$$f(x+\delta) = f(x) + f'(x)\delta + O(\delta^2)$$

shows that small perturbations in x can lead to relatively large changes in f. The idea that the magnitude of the derivative provides a measure of the sensitivity or conditioning of f carries over to higher-dimensional maps [5], [23], [30], [31], [34], [35], [40], [45]. For example, if f maps $\mathbb{R}^n$ into $\mathbb{R}$, the gradient of f at $x \in \mathbb{R}^n$ is the row vector $\nabla f(x) = (\partial f(x)/\partial x_1, \ldots, \partial f(x)/\partial x_n)$, and the Taylor expansion of f has the form

$$f(x+\delta z) = f(x) + \delta \nabla f(x) z + O(\delta^2), \tag{1}$$

where $\|z\| = 1$ with $\|\cdot\|$ denoting the vector 2-norm defined by $\|z\|^2 = \Sigma |z_i|^2$. For notational convenience in the sequel, we use v^T to denote the gradient

$$v^T \equiv \nabla f(x). \tag{2}$$

It can be seen from (1) that the norm of the gradient ($\|\nabla f(x)\| = \|v\|$) is an appropriate measure of the local sensitivity of f. Nonlocal sensitivity in the sense of a Lipschitz-type condition measure is much harder to deal with. Demmel [8] has shown that for some problems, local condition information gives upper and lower bounds on nonlocal sensitivity. However, the depth of this subject is beyond the scope of this paper, and we are concerned only with the issue of estimating local sensitivity. For simplicity of exposition, we also assume throughout this paper that all functions considered are at least twice continuously differentiable.

Somewhat surprisingly, rigorous probability arguments show that only a few function evaluations are needed to obtain estimates of the norm of v that are accurate and reliable in a sense described below. These results are then applied to the problem of condition estimation for more general matrix functions such as the matrix exponential map $X \mapsto e^X$ and the map $(A, F, G) \mapsto X$, where X satisfies an algebraic Riccati equation

*Received by the editors February 18, 1992; accepted for publication (in revised form) January 27, 1993. This research was supported in part by the National Science Foundation grant ECS-9120643, the Air Force Office of Scientific Research grant AFOSR-91-0240, and the Office of Naval Research grant N00014-92-J-1706.

†Department of Electrical and Computer Engineering, University of California, Santa Barbara, California 93106-9560 (laub@ece.ucsb.edu).

$$0 = G + A^T X + XA - XFX. \tag{3}$$

This is illustrated in §4 with many other examples. This section also compares the small-sample statistical method with the more traditional power method of condition estimation. Sections 2 and 3 develop the theoretical basis for the small-sample methods.

The central idea of this method is that the norm of the gradient can be estimated if we can afford another function evaluation beyond $f(x)$, say, at $x + \delta z$. If z has unit norm, then the Newton difference defined by

$$dz \equiv \frac{f(x + \delta z) - f(x)}{\delta} \tag{4}$$

satisfies

$$dz = v^T z + O(\delta). \tag{5}$$

If z is selected uniformly and randomly from the unit sphere S_{n-1} in n dimensions, then (see Theorem 2.1) the expected value of $|v^T z|$ is given by

$$E(|v^T z|) = \|v\| E_n, \tag{6}$$

where $E_1 = 1$, and for $n > 1$

$$E_n = \frac{1 \cdot 3 \cdot 5 \cdots (n-2)}{2 \cdot 4 \cdot 6 \cdots (n-1)} \quad \text{for } n \text{ odd}, \tag{7}$$

$$E_n = \frac{2}{\pi} \frac{2 \cdot 4 \cdot 6 \cdots (n-2)}{1 \cdot 3 \cdot 5 \cdots (n-1)} \quad \text{for } n \text{ even}. \tag{8}$$

We will use the standard symbol !! to indicate the "skip" factorial [1], [18], so that $7!! = 1 \cdot 3 \cdot 5 \cdot 7$ and $6!! = 2 \cdot 4 \cdot 6$ with $0!! = 1$. Thus $E_n = (n-2)!!/(n-1)!!$ when n is odd, etc. Since the expected value function E_n plays such a central role in statistical condition estimation, a short discussion of its evaluation for large values of n is given in the Appendix.

By (5)–(6), the condition estimator $|dz|/E_n$ has expected value equal to the true condition number $\|v\|$ plus a term of order δ. Typically, we can take δ sufficiently small (say, less than 10^{-12}), so that for the purposes of estimating $\|v\|$ the expected value of $|dz|/E_n$ is more than adequate. To avoid having to add terms of order δ to all of our results, we will now make the simplifying assumption that for any given vector $z \in S_{n-1}$ we are able to evaluate $v^T z$ with the understanding that in real life we are merely able to evaluate dz as in (4). This is reasonable when we remember that for most sensitivity estimates, we only need to know the true condition number to within a factor of 10 or so, and we can often tolerate errors in the estimate up to a factor of 100.

The important question then becomes, what is the probability that the estimator

$$\zeta \equiv |v^T z|/E_n$$

lies within a given factor w of $\|v\|$? In §2 we derive an exact formula (see Theorem 2.3) for this probability, and we show that for $w > 1$

$$\Pr\left(\frac{\|v\|}{w} \le \zeta \le w\|v\|\right) \ge 1 - \frac{2}{\pi w} + O\left(\frac{1}{w^2}\right). \tag{9}$$

Thus ζ is a linear or first-order condition estimate in the sense that the chance of a catastrophically low or high estimate is inversely proportional to the size of the error. For example,

$\Pr(\|v\|/100 \le \zeta \le 100\|v\|) \ge 0.9936$, so that the chance of being off by more than a factor of 100 is less than 1 in 100.

While this is good, there are some situations in which we need more reliability. One way to achieve this is to use more function evaluations to get different values $\zeta^{(1)}, \zeta^{(2)}, \ldots, \zeta^{(m)}$ corresponding to independently randomly generated vectors $z^{(1)}, z^{(2)}, \ldots, z^{(m)}$ in S_{n-1} and then to take the average

$$\zeta(m) \equiv \frac{\zeta^{(1)} + \cdots + \zeta^{(m)}}{m}. \tag{10}$$

(As a matter of notation, we use subscripts to denote particular entries in a vector and superscripts in parentheses to distinguish between different vectors.)

This is the standard Monte Carlo method [22], [44], [51] for finding the expected value, and we can show that for $w > 1$

$$\Pr\left(\frac{\|v\|}{w} \le \zeta(m) \le w\|v\|\right) \ge 1 - \frac{1}{m!}\left(\frac{2m}{\pi w}\right)^m + O\left(\frac{1}{w^{m+1}}\right) \tag{11}$$

with asymptotic equality as $w \to +\infty$ or $m \to +\infty$. Thus $\zeta(m)$ is an mth-order condition estimator. That is, the probability that the estimator is off by more than a factor of w is less than a constant times w^{-m}. For example, with $m = 3$, which corresponds to three extra function evaluations, we have $\Pr(\|v\|/100 \le \zeta(3) \le 100\|v\|) \ge 0.99999884$, so that the chance of being off by a factor of 100 or more is approximately one in a million. Hence very reliable condition estimates can be obtained with only a few extra function evaluations. Note that this is in contrast to most Monte Carlo applications that usually involve thousands of population samples to estimate the expected value. Thus the method given above could be described as a "small-sample Monte Carlo method." However, to avoid the usual large-sample connotations of this phrase, we use instead the alternative description "averaged small-sample statistical method."

Remark 1. From (4) and (5) we can get the true value of $\|v\|$ at a price of $m = n$ extra function evaluations by simply setting $z^{(i)} = e^{(i)}$ and actually forming the gradient vector. (Here $e^{(i)}$ denotes the unit vector with a one in the ith position.) In general, however, this is much too costly because n may be in the hundreds or thousands.

Although the method of averaging is entirely adequate, we can obtain sharper estimates that are more easily analyzed by exploiting the geometry of the situation. For example, suppose that we know $v^T z^{(1)}$ and $v^T z^{(2)}$. From this information we can find the projection of v onto the span of $z^{(1)}$ and $z^{(2)}$. If $z^{(1)}$ and $z^{(2)}$ are orthogonal vectors in S_{n-1}, then the norm of this projection is given by

$$\left(|v^T z^{(1)}|^2 + |v^T z^{(2)}|^2\right)^{1/2}.$$

If we further assume that $z^{(1)}$ and $z^{(2)}$ are chosen so that their span is uniformly and randomly selected from the space of all two-dimensional subspaces of $\mathbb{R}^n$, then the expected value of the norm of the projection is given by (see Theorem 3.1)

$$E\left(\sqrt{|v^T z^{(1)}|^2 + |v^T z^{(2)}|^2}\right) = \frac{E_n}{E_2}\|v\|, \tag{12}$$

where E_n and E_2 are defined by (7) and (8). (Selecting $z^{(1)}$ and $z^{(2)}$ so that the above assumptions are satisfied is not difficult; from the work in [43] and [47] we may select $z^{(1)}$ and $z^{(2)}$ randomly and uniformly from S_{n-1} and then find an orthonormal basis for their span by using, say, a Gram–Schmidt procedure or a QR decomposition [16]. A similar procedure can

be used to randomly generate higher-dimensional subspaces and is described in more detail in §4.)

We see from (12) that the condition estimator defined by

$$\nu \equiv \frac{E_2}{E_n}\sqrt{|v^T z^{(1)}|^2 + |v^T z^{(2)}|^2} \tag{13}$$

has expected value equal to $\|v\|$. It can be shown (see Theorem 3.3) that for $w > 1$,

$$\Pr\left(\frac{\|v\|}{w} \le \nu \le w\|v\|\right) = \left(1 - \left(\frac{E_n}{E_2 w}\right)^2\right)^{(n-2)/2} - \left(1 - \left(\frac{E_n w}{E_2}\right)^2\right)^{(n-2)/2} \tag{14}$$

$$\approx 1 - \frac{\pi}{4w^2}, \tag{15}$$

which is slightly better than the corresponding estimate (11) for $\zeta(2)$.

This estimation method, which we refer to as the "subspace statistical method," can easily be extended: let $z^{(1)}, \ldots, z^{(m)}$ be an orthonormal basis for a subspace of dimension m that has been randomly and uniformly selected from the space of all m-dimensional subspaces of $\mathbb{R}^n$. (As above, this can be done by choosing m random vectors and then orthogonalizing.) Then $\left(|v^T z^{(1)}|^2 + \cdots + |v^T z^{(m)}|^2\right)^{1/2}$ is the norm of the projection of v onto the span of $z^{(1)}, \ldots, z^{(m)}$ and the expected value of this norm is (see Theorem 3.1)

$$E\left(\sqrt{|v^T z^{(1)}|^2 + \cdots + |v^T z^{(m)}|^2}\right) = \frac{E_n}{E_m}\|v\|. \tag{16}$$

Thus the subspace condition estimator defined by

$$\nu = \nu(m) \equiv \frac{E_m}{E_n}\sqrt{|v^T z^{(1)}|^2 + \cdots + |v^T z^{(m)}|^2} \tag{17}$$

has expected value $\|v\|$. These condition estimators give better results than the averaged statistical estimators and are analytically very tractable. For example, the kth moment of $\nu(m)$ is given by the formula (see Theorem 3.1)

$$E(\nu^k(m)) = \frac{E_n E_{n+1} \cdots E_{n+k}}{E_m E_{m+1} \cdots E_{m+k}} \left(\frac{\|v\| E_m}{E_n}\right)^k. \tag{18}$$

Additionally, a simple recursion can be derived (see Theorem 3.3) that gives exact values for the probability that $\nu(m)$ lies within a factor of w of $\|v\|$. Let us denote this probability by $P_m(w)$:

$$P_m(w) \equiv \Pr\left(\frac{\|v\|}{w} \le \nu(m) \le w\|v\|\right). \tag{19}$$

Table 1 gives lower bounds (independent of n) for $P_m(w)$ for the first few values of m and for a range of values of w.

Section 2 develops the theory of the averaged estimator $\zeta(m)$. Then in §3 results are presented for the subspace estimator $\nu(m)$.

2. Averaged statistical condition estimates.

2.1. Notation. It is convenient to begin this section with a review of some definitions from probability theory [13]. Let ϕ be a random variable and let $d(s; \phi)$ denote the value at s of the probability density function associated with ϕ. That is, for all $t \in \mathbb{R}$,

TABLE 1
Lower bounds on the probability that $\nu(m)$ equals $\|v\|$ to within a factor of w.

w	Lower bound for $P_1(w)$	Lower bound for $P_2(w)$	Lower bound for $P_3(w)$	Lower bound for $P_4(w)$
3	0.7736	0.9156	0.9632	0.9831
5	0.8732	0.9691	0.9916	0.9976
10	0.9364	0.9922	0.9989	0.9998
10^2	0.9936	0.9999	$1-10^{-5}$	$1-10^{-7}$
10^3	0.9994	$1-10^{-6}$	$1-10^{-8}$	$1-10^{-11}$

$$\Pr(\phi \le t) = \int_{-\infty}^{t} d(s;\phi)ds. \tag{20}$$

Similarly, if $\phi_1, \ldots, \phi_m$ are random variables, let $d(s_1, \ldots, s_m; \phi_1, \ldots, \phi_m)$ denote the value at $(s_1, \ldots, s_m)$ of the joint probability density function associated with $\phi_1, \ldots, \phi_m$. That is, for all $(t_1, \ldots, t_m) \in \mathbb{R}^m$,

$$\Pr(\phi_1 \le t_1, \ldots, \phi_m \le t_m) = \int_{-\infty}^{t_1} \cdots \int_{-\infty}^{t_m} d(s_1, \ldots, s_m; \phi_1, \ldots, \phi_m)ds_m \cdots ds_1. \tag{21}$$

If g is a measurable function from $\mathbb{R}^m$ to $\mathbb{R}$, and A is a measurable set in $\mathbb{R}$, then the probability that the random variable $g(\phi_1, \ldots, \phi_m)$ takes on a value in A is given by

$$\Pr(g(\phi_1, \ldots, \phi_m) \in A) = \int_{g \in A} d(s_1, \ldots, s_m; \phi_1, \ldots, \phi_m)ds_m \cdots ds_1, \tag{22}$$

where the notation $g \in A$ denotes the set of points $(s_1, \ldots, s_m)$ such that $g(s_1, \ldots, s_m) \in A$. The expected value of $g(\phi_1, \ldots, \phi_m)$ is defined to be

$$E(g(\phi_1, \ldots, \phi_m)) = \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} g(s_1, \ldots, s_m)d(s_1, \ldots, s_m; \phi_1, \ldots, \phi_m)ds_m \cdots ds_1. \tag{23}$$

If $\phi_1, \phi_2, \ldots, \phi_m$ are independent, then

$$d(s_1, \ldots, s_m; \phi_1, \ldots, \phi_m) = d(s_1; \phi_1) \cdots d(s_m; \phi_m). \tag{24}$$

Because of the multiple integrals involved in higher dimensions, it is occasionally useful to switch from Cartesian coordinates to polar coordinates [11]

$$\begin{aligned} x_1 &= r\cos\theta_1, \\ x_2 &= r\sin\theta_1\cos\theta_2, \\ &\vdots \\ x_{m-1} &= r\sin\theta_1\sin\theta_2 \cdots \sin\theta_{m-2}\cos\theta_{m-1}, \\ x_m &= r\sin\theta_1\sin\theta_2 \cdots \sin\theta_{m-2}\sin\theta_{m-1}. \end{aligned}$$

The Jacobian of the transformation from $x_1, x_2, \ldots, x_m$ to $r, \theta_1, \theta_2, \ldots, \theta_{m-1}$ is given by

$$J = \left| \frac{\partial x_1, \ldots, x_m}{\partial r, \theta_1, \ldots, \theta_{m-1}} \right| = r^{m-1}\sin^{m-2}\theta_1 \sin^{m-3}\theta_2 \cdots \sin\theta_{m-2}.$$

2.2. Inner product distributions on S_{n-1}. Since inner products of the form $v^T z$ are invariant under an orthogonal rotation of v and z, we may select a coordinate system in which the vector v is just a scalar multiple of the vector $e^{(1)} \equiv (1, 0, \ldots, 0)^T$. This means that $|v^T z| = \|v\|\,|z_1|$, where $z = (z_1, \ldots, z_n)^T$. Because of this, our concern is with the distribution of $|z_1|$ when z is uniformly and randomly distributed over the unit sphere S_{n-1} in $\mathbb{R}^n$. For simplicity we assume that $n \ge 3$ throughout the rest of this paper.

The density function for z_1 is given by [42]

$$d(s_1; z_1) = c\left(1 - s_1^2\right)^{(n-3)/2} \quad \text{for } -1 \le s_1 \le 1, \tag{25}$$
$$= 0 \quad \text{otherwise,}$$

where c is a normalization constant given by

$$c = \left(\int_{-1}^{1} \left(1 - s_1^2\right)^{(n-3)/2} ds_1\right)^{-1}. \tag{26}$$

The density function for $|z_1|$ is thus seen to be

$$d(s_1; |z_1|) = 2c\left(1 - s_1^2\right)^{(n-3)/2} \quad \text{for } 0 \le s_1 \le 1, \tag{27}$$
$$= 0 \quad \text{otherwise.}$$

The probability that $|z_1|$ is less than or equal to a given nonnegative value ϵ is

$$\Pr(|z_1| \le \epsilon) = 2c \int_0^{\epsilon} \left(1 - s_1^2\right)^{(n-3)/2} ds_1, \tag{28}$$

and the expected value of $|z_1|$, which we denote by E_n, is

$$E_n \equiv E(|z_1|) = 2c \int_0^1 |s_1| \left(1 - s_1^2\right)^{(n-3)/2} ds_1 = \frac{2c}{n-1}. \tag{29}$$

In the following we use the symbol !! to denote the "skip" factorial defined in §1.

THEOREM 2.1. *The expected value function E_n is given by*

$$E_n = \frac{(n-2)!!}{(n-1)!!} \quad \textit{for } n \textit{ odd}, \tag{30}$$

$$E_n = \frac{2}{\pi} \frac{(n-2)!!}{(n-1)!!} \quad \textit{for } n \textit{ even}. \tag{31}$$

(*See also the Appendix.*)

Proof. Use the change of variables $s_1 = \cos\theta$ and the recursion formula [18, p. 369]

$$\int_0^{\pi/2} \sin^j \theta d\theta = \frac{j-1}{j} \int_0^{\pi/2} \sin^{j-2} \theta d\theta$$

in (29). □

To discuss the probability that $|z_1|$ is less than or equal to a given value ϵ, it is convenient to introduce the function $g_n = g_n(\epsilon)$ defined as follows: $g_n(\epsilon) = 0$ if $\epsilon \le 0$ and $g_n(\epsilon) = 1$ if $\epsilon \ge 1$. If n is odd and $0 < \epsilon < 1$, then

$$g_n(\epsilon) = \epsilon\left(1 + \frac{1}{2}(1-\epsilon^2) + \frac{1 \cdot 3}{2 \cdot 4}(1-\epsilon^2)^2 + \cdots + \frac{(n-4)!!}{(n-3)!!}(1-\epsilon^2)^{(n-3)/2}\right). \tag{32}$$

If n is even and $0 < \epsilon < 1$, then

$$g_n(\epsilon) = \frac{2}{\pi}\epsilon\left(\frac{\arcsin\epsilon}{\epsilon} + \left(1-\epsilon^2\right)^{1/2} + \frac{2}{3}\left(1-\epsilon^2\right)^{3/2} + \cdots + \frac{(n-4)!!}{(n-3)!!}\left(1-\epsilon^2\right)^{(n-3)/2}\right). \tag{33}$$

LEMMA 2.2. *For* $n \geq 3$,

$$\Pr(|z_1| \leq \epsilon) = g_n(\epsilon). \tag{34}$$

Proof. Let $\epsilon = \sin\delta$ and define $\Pr(|z_1| \leq \epsilon) \equiv Q_n(\delta)$. Direct evaluation shows that $Q_2(\delta) = \frac{2}{\pi}\arcsin\epsilon$ and $Q_3(\delta) = \epsilon$. Standard results from [18] show that

$$Q_{n+2}(\delta) = Q_n(\delta) + \frac{\sin(\delta)\cos^{n-1}(\delta)}{(n-1)\displaystyle\int_0^{\pi/2} \sin^{n-2}(\theta)d\theta}.$$

When combined with $\cos\delta = (1-\epsilon^2)^{1/2}$, the desired result is obtained by induction on n. □

Our principal concern is with the properties of

$$\zeta = \frac{|v^T z|}{E_n} \tag{35}$$

as a condition estimator, since the right-hand side of (35) can be evaluated numerically (see the Appendix).

THEOREM 2.3. *The expected value of* ζ *is* $\|v\|$, *and for* $w > 1$

$$\Pr\left(\frac{\|v\|}{w} \leq \zeta \leq w\|v\|\right) = g_n(wE_n) - g_n\left(\frac{E_n}{w}\right). \tag{36}$$

Proof. Using Lemma 2.2 and $|z_1| = |v^T z|/\|v\|$, we find that

$$\begin{aligned}\Pr\left(\frac{|v^T z|}{E_n} \leq w\|v\|\right) &= \Pr\left(\frac{|v^T z|}{\|v\|} \leq wE_n\right)\\ &= \Pr(|z_1| \leq wE_n)\\ &= g_n(wE_n)\,.\end{aligned}$$

Replacing w with $\frac{1}{w}$ gives $\Pr(|v^T z|/E_n \leq \|v\|/w) = g_n(E_n/w)$. Taken together these expressions give (36). The fact that $E(\zeta) = \|v\|$ follows from a similar argument or by noting that

$$E(\zeta) = E\left(\frac{|v^T z|}{E_n}\right) = \frac{E(|v^T z|)}{E_n} = \frac{E(\|v\|\,|z_1|)}{E(|z_1|)} = \|v\|. \qquad \square$$

Although (36) is exact, it is a little hard to work with. The next few results deal with the averaged variables $\zeta(m)$ and show that (36) can be replaced, for $m = 1$, by the simple bound

$$\Pr\left(\frac{\|v\|}{w} \leq \zeta \leq w\|v\|\right) \geq 1 - \frac{2}{\pi w} - \frac{4}{w}e^{-w^2/4} \quad \text{for } n \geq 4. \tag{37}$$

As in the discussion in the Introduction, if we can afford to make m extra function evaluations, then one way of increasing the reliability and accuracy of our condition estimates is to use the averaged estimator

$$\zeta(m) = \frac{|v^T z^{(1)}| + \cdots + |v^T z^{(m)}|}{mE_n}, \tag{38}$$

where $z^{(1)}, \ldots, z^{(m)}$ are independently, randomly, and uniformly chosen vectors from the unit sphere in n dimensions.

2.3. Underestimation. The following theorem bounds the probability that $\zeta(m)$ underestimates $\|v\|$ by a factor $w > 1$.

THEOREM 2.4. *Let $\zeta(m)$ be given by* (38). *Then*

$$\Pr\left(\zeta(m) \le \frac{\|v\|}{w}\right) < \left(\frac{2m}{\pi w}\right)^m \frac{1}{m!}. \tag{39}$$

Proof. Let $z_1^{(i)}$ denote the first component of $z^{(i)}$ for $1 \le i \le m$. The independence assumption together with (27) implies that the joint density function for $|z_1^{(1)}|, \ldots, |z_1^{(m)}|$ satisfies

$$d\left(s_1, \ldots, s_m; |z_1^{(1)}|, \ldots, |z_1^{(m)}|\right) = d\left(s_1; |z_1^{(1)}|\right) \cdots d\left(s_m; |z_1^{(m)}|\right) \tag{40}$$

$$= (2c)^m \left(1 - s_1^2\right)^{(n-3)/2} \cdots \left(1 - s_m^2\right)^{(n-3)/2} \tag{41}$$

$$\le (2c)^m \tag{42}$$

for $0 \le s_i \le 1$ and $1 \le i \le m$; otherwise the density is zero.

To establish (39), write

$$\begin{aligned}
\Pr\left(\zeta(m) \le \frac{\|v\|}{w}\right) &= \Pr\left(\frac{|v^T z^{(1)}| + \cdots + |v^T z^{(m)}|}{m E_n} \le \frac{\|v\|}{w}\right) \\
&= \Pr\left(\frac{|v^T z^{(1)}|}{\|v\|} + \cdots + \frac{|v^T z^{(m)}|}{\|v\|} \le \frac{m E_n}{w}\right) \\
&= \Pr\left(|z_1^{(1)}| + \cdots + |z_1^{(m)}| \le \frac{m E_n}{w}\right) \\
&= \int_T d(s_1, \ldots, s_m; |z_1^{(1)}|, \ldots, |z_1^{(m)}|)\, ds_m \cdots ds_1,
\end{aligned}$$

where T denotes the triangular wedge $\Sigma s_i \le m E_n / w$, $s_i \ge 0$. Standard calculus techniques suffice to show that this wedge has volume equal to $(m E_n / w)^m / m!$. Combining this with the bound on the density function shows that

$$\begin{aligned}
\Pr\left(\zeta(m) \le \frac{\|v\|}{w}\right) &\le \int_T (2c)^m \, ds_m \cdots ds_1 \\
&= \left(\frac{2cm E_n}{w}\right)^m \frac{1}{m!}.
\end{aligned}$$

However, $2cE_n = (n-1)E_n^2 < \frac{2}{\pi}$ by (29) and Lemma 6.1 in the Appendix. This completes the proof of (39). □

Remark 2. Slightly more analysis shows that the bound (39) is asymptotically an equality as $w \to +\infty$.

2.4. Overestimation. The probability that $\zeta(m)$ overestimates $\|v\|$ turns out to be rather insignificant. First, we need the following technical lemma.

LEMMA 2.5. *For $m \ge 1$ and $r_0 > \sqrt{2(m-1)}$*

$$\int_{r_0}^{+\infty} r^{m-1} e^{-r^2/2} dr \le 2 r_0^{m-2} e^{-r_0^2/2}. \tag{43}$$

Proof. Use the standard methods of [7]. □

THEOREM 2.6. *Let $z^{(1)}, \ldots, z^{(m)}$ be independently, randomly, and uniformly chosen from S_{n-1} and define $\zeta(m)$ by* (38). *Then for $n \geq 5$, $w \geq 4$, and $2 \leq m \leq n-3$, the probability that $\zeta(m)$ overestimates $\|v\|$ by a factor of w is bounded by*

$$\Pr(\zeta(m) \geq w\|v\|) < \frac{6}{(m-2)!!}\left(\frac{w\sqrt{m}}{2}\right)^{m-2} e^{-mw^2/8}. \tag{44}$$

Proof. As in the proof of Theorem 2.4,

$$\begin{aligned}\Pr(\zeta(m) \geq w\|v\|) &= \Pr\left(|z_1^{(1)}| + \cdots + |z_1^{(m)}| \geq mwE_n\right)\\ &= \int_O d(s_1, \ldots, s_m; |z_1^{(1)}|, \ldots, |z_1^{(m)}|)\, ds_m \cdots ds_1,\end{aligned}$$

where O denotes the outer triangular wedge $\Sigma s_i \geq mwE_n$, $s_i \geq 0$. However, the region O is contained in the outer annular wedge W defined by $\Sigma s_i^2 \geq mw^2E_n^2$ because

$$(mwE_n)^2 \leq \left(\sum_{i=1}^m s_i\right)^2 \leq m\sum_{i=1}^m s_i^2$$

by the Cauchy–Schwarz inequality.

We also need an upper bound on the density function; this can be obtained by using a clever trick from [42]. Since $1 - x \leq e^{-x}$ for all x, we have $(1 - s_i^2)^{(n-3)/2} \leq e^{-(n-3)s_i^2/2}$ and

$$d\left(s_1, \ldots, s_m; |z_1^{(1)}|, \ldots, |z_1^{(m)}|\right) \leq (2c)^m e^{-(n-3)/2\,(s_1^2+\cdots+s_m^2)}. \tag{45}$$

Altogether this gives

$$\begin{aligned}\Pr(\zeta(m) \geq w\|v\|) &\leq \int_W d(s_1, \ldots, s_m; |z_1^{(1)}|, \ldots, |z_1^{(m)}|)\, ds_m \cdots ds_1\\ &\leq \int_W (2c)^m e^{-(n-3)\,(s_1^2+\cdots+s_m^2)/2}\, ds_m \cdots ds_1\\ &= \mu_m (2c)^m \int_{\sqrt{m}wE_n}^{+\infty} \rho^{m-1} e^{-(n-3)\rho^2/2} d\rho,\end{aligned} \tag{46}$$

where the last equality was obtained by switching to polar coordinates in m dimensions with $\rho^2 = s_1^2 + \cdots + s_m^2$. The term μ_m is given by

$$\mu_m \equiv \int_0^{\pi/2} \cdots \int_0^{\pi/2} \sin^{m-2}(\theta_1) \cdots \sin(\theta_{m-2}) d\theta_{m-1} \cdots d\theta_1,$$

and satisfies $\mu_m \leq (\frac{\pi}{2})^{m/2}/(m-2)!!$. Now for $w \geq 4$ and $n \geq 5$, we make the change of variables $r = \sqrt{n-3}\,\rho$ above, and then we apply Lemma 2.5 with $r_0 \equiv \sqrt{m}w/2 < \sqrt{m}wE_n\sqrt{n-3}$ to get

$$\Pr(\zeta(m) \geq w\|v\|) < \frac{2}{(m-2)!!}\left(\frac{2c^2\pi}{n-3}\right)^{m/2}\left(\frac{w\sqrt{m}}{2}\right)^{m-2} e^{-mw^2/8}.$$

Finally, by (29) and Lemma 6.1 in the Appendix,

$$
\begin{aligned}
2\left(\frac{2c^2\pi}{n-3}\right)^{m/2} &< 2\left(\frac{n-1}{n-3}\right)^{m/2} \\
&= 2\left(1+\frac{2}{n-3}\right)^{m/2} \\
&< 2e^{m/(n-3)} \\
&< 6,
\end{aligned}
\tag{47}
$$

since $m \leq n-3$ by assumption. □

Remark 3. The restrictions $w \geq 4$, $n \geq 5$, and $m \geq 2$ in Theorem 2.6 were selected to reduce the complexity of the proof. Similar results can be obtained for weaker hypotheses.

3. Subspace statistical condition estimates. Let O_n denote the set of orthogonal matrices of order n. If U is randomly and uniformly selected from O_n [43], then $Ue^{(1)}$ is randomly and uniformly distributed over the unit sphere S_{n-1} in n dimensions, where $e^{(1)} = (1, 0, \ldots, 0)^T$. In fact, for any fixed unit vector w, the product Uw is randomly and uniformly distributed over S_{n-1}. Similarly, by using the methods of [28], the vectors $Ue^{(1)}, \ldots, Ue^{(m)}$ form an orthonormal basis for a subspace of dimension m which is randomly and uniformly distributed over the set of all subspaces of dimension m in $\mathbb{R}^n$.

Let $z^{(i)} = Ue^{(i)}$. Then $v^T z^{(i)} = v^T U e^{(i)} = (U^T v)^T e^{(i)} = \|v\|\, w^T e^{(i)}$, where $w = U^T v/\|v\|$. The importance of this is that the joint density function of the random variables $w_i \equiv w^T e^{(i)}$ is known [42]. Specifically, for $n \geq 3$ and $1 \leq m < n$,

$$
d(s_1, \ldots, s_m; w_1, \ldots, w_m) = c_1 \left(1 - r^2\right)^{(n-m-2)/2} \quad \text{for } 0 \leq r \leq 1,
\tag{48}
$$

where $r^2 = s_1^2 + \cdots + s_m^2$ and c_1 is a normalization constant. For r not in the interval [0, 1], the density is zero. Since

$$
\sqrt{(v^T z^{(i)})^2 + \cdots + (v^T z^{(m)})^2} = \|v\| \sqrt{(w^T e^{(1)})^2 + \cdots + (w^T e^{(m)})^2},
\tag{49}
$$

we need to consider the random variable ϕ defined by

$$
\phi = \phi(m) \equiv \sqrt{(w^T e^{(1)})^2 + \cdots + (w^T e^{(m)})^2}\,.
\tag{50}
$$

THEOREM 3.1. *The expected value of ϕ is given by*

$$
E(\phi(m)) = \frac{E_n}{E_m},
\tag{51}
$$

where E_n is defined by (7)–(8). *More generally, the kth moment of ϕ is given by*

$$
E(\phi^k(m)) = \frac{E_n\, E_{n+1} \cdots E_{n+k-1}}{E_m\, E_{m+1} \cdots E_{m+k-1}}.
\tag{52}
$$

Proof. After changing to polar coordinates in m dimensions and using (48), we find that

$$
E(\phi^k(m)) = \frac{I(k, n, m)}{I(0, n, m)},
\tag{53}
$$

where

$$I(i,n,m) \equiv \int_0^1 \left(1-r^2\right)^{(n-m-2)/2} r^{m+i-1}\, dr. \tag{54}$$

Equations (51) and (52) can then be proved by using (53) and induction, but the details are omitted for the sake of brevity. □

Remark 4. From (52), the variance of $\phi(m)$ is given by

$$\begin{aligned}\mathrm{Var}(\phi(m)) &= E\left(\phi^2(m)\right) - E^2(\phi(m)) \\ &= \frac{E_n\, E_{n+1}}{E_m\, E_{m+1}} - \frac{E_n^2}{E_m^2} \\ &= \frac{m}{n} - \frac{E_n^2}{E_m^2},\end{aligned}$$

because $E_k\, E_{k+1} = \frac{2}{\pi k}$. Since the variance is nonnegative, this gives $m E_m^2 \geq n E_n^2$. Thus, for $m \leq n$, we have $m E_m^2 \geq n E_n^2 \geq m E_n^2$, which gives $E_m \geq E_n$, i.e., E_n is decreasing with n. This is used in the proof of Lemma 6.1 in the Appendix. In fact, from the equality, $E_{n+2} = \left(\frac{n}{n+1}\right) E_n$, it is easily seen by induction that $E_n < 1/\sqrt{n}$.

Theorem 3.1 shows that the subspace estimator ν defined by

$$\nu = \nu(m) \equiv \frac{E_m}{E_n} \sqrt{|v^T z^{(1)}|^2 + \cdots + |v^T z^{(m)}|^2} \tag{55}$$

has expected value $\|v\|$. The significance of this is that ν can be calculated in m function evaluations as in the discussion in the Introduction (see also §4 for more details). Theorem 3.1 also shows that the kth moment of $\nu(m)$ is given by

$$E(\nu^k(m)) = \frac{E_n E_{n+1} \cdots E_{n+k-1}}{E_m E_{m+1} \cdots E_{m+k-1}} \left(\frac{\|v\| E_m}{E_n}\right)^k. \tag{56}$$

Using $E_k E_{k+1} = \frac{2}{\pi k}$ and Theorem 3.1, the variance and skewness of $\nu(m)$ can be found:

$$\begin{aligned}\mathrm{Var}(\nu(m)) &\equiv E((\nu(m) - E(\nu(m)))^2) \\ &= \left(\frac{m E_m^2}{n E_n^2} - 1\right) \|v\|^2, \\ \mathrm{Skew}(\nu(m)) &\equiv E((\nu(m) - E(\nu(m)))^3) \\ &= \left(\left(\frac{m+1}{n+1} - \frac{3m}{n}\right) \frac{E_m^2}{E_n^2} + 2\right) \|v\|^3.\end{aligned}$$

Note that skewness is defined in many statistics texts as the centered third moment divided by the variance, but, for convenience here, we shall refer to the centered third moment by this appellation.

We can now also determine the variance and skewness of the averaged statistical estimator $\zeta(m)$ because $\zeta(1) = \nu(1)$ and

$$\operatorname{Var}(\zeta(m)) = \frac{1}{m}\operatorname{Var}(\zeta(1))$$

$$= \frac{1}{m}\left(\frac{1}{nE_n^2} - 1\right)\|v\|^2,$$

$$\operatorname{Skew}(\zeta(m)) = \frac{1}{m^2}\operatorname{Skew}(\zeta(1))$$

$$= \frac{1}{m^2}\left(\left(\frac{2}{n+1} - \frac{3}{n}\right)\frac{1}{E_n^2} + 2\right)\|v\|^3.$$

For example, the first example in §4 deals with the sensitivity of the roots of the polynomial $p(t) = (t-1)\cdots(t-(n-1))$ with respect to variation in the n coefficients of p. A comparison of the variance and skewness of the averaged and subspace statistical estimators for this problem with $n = 9$ is given in Table 2, which shows that both the variance and the skewness of the subspace estimator are smaller in absolute value than for the averaged estimator.

TABLE 2
Variance and skewness comparison of the two estimators for $n = 9$.

m	$\operatorname{Var}(\zeta(m))$	$\operatorname{Var}(\nu(m))$	$\operatorname{Skew}(\zeta(m))$	$\operatorname{Skew}(\nu(m))$
1	$0.4861\,\|v\|^2$	$0.4861\,\|v\|^2$	$0.2167\,\|v\|^3$	$0.2167\,\|v\|^3$
2	$0.2430\,\|v\|^2$	$0.2046\,\|v\|^2$	$0.0542\,\|v\|^3$	$0.0125\,\|v\|^3$
3	$0.1620\,\|v\|^2$	$0.1146\,\|v\|^2$	$0.0241\,\|v\|^3$	$-0.006\,\|v\|^3$

To develop an expression for the probability that $\nu(m)$ lies within a factor w of $\|v\|$, we need some preliminary results. Define

$$g_{n,m}(\epsilon) \equiv \Pr(\phi(m) \le \epsilon)\,. \tag{57}$$

As in the proof of Theorem 3.1, $\phi(1) = |z_1|$ so that $g_{n,1} = g_n$, where g_n is defined by (32) and (33).

Using polar coordinates and (48) shows that for $0 \le \epsilon \le 1$,

$$g_{n,m}(\epsilon) = \frac{T_{n,m}(\epsilon)}{T_{n,m}(1)}, \tag{58}$$

where

$$T_{n,m}(\epsilon) \equiv \int_0^\epsilon \left(1 - r^2\right)^{(n-m-2)/2} r^{m-1}\,dr. \tag{59}$$

LEMMA 3.2. *For $0 \le \epsilon \le 1$ and $m = 2$, we have*

$$g_{n,2}(\epsilon) = 1 - \left(1 - \epsilon^2\right)^{(n-2)/2}\,. \tag{60}$$

For $m > 2$, the value of $g_{n,m}(\epsilon)$ can be found by using the values of $g_{n,1}(\epsilon)$ and $g_{n,2}(\epsilon)$ with the recursion

$$g_{n,m}(\epsilon) = g_{n,m-2}(\epsilon) - \frac{(n-2)!!\,\epsilon^{m-2}}{(m-2)!!\,(n-m)!!}\left(1-\epsilon^2\right)^{(n-m)/2} \quad \textit{for } m \textit{ even}, \tag{61}$$

$$= g_{n,m-2}(\epsilon) - \frac{(n-1)!!\,E_n\,\epsilon^{m-2}}{(m-2)!!\,(n-m)!!}\left(1-\epsilon^2\right)^{(n-m)/2} \quad \textit{for } m \textit{ odd}. \tag{62}$$

Proof. Use integration by parts with $v = -\left(1-r^2\right)^{(n-m)/2}/(n-m)$ and $u = r^{m-2}$ in the definition of $T_{n,m}$. For $m = 2$ this gives (60). For $m > 2$, we get

$$T_{n,m}(\epsilon) = \frac{m-2}{n-m} T_{n,m-2}(\epsilon) - \frac{\epsilon^{m-2}}{n-m}\left(1-\epsilon^2\right)^{(n-m)/2}, \tag{63}$$

which, when coupled with the definition of E_n, yields (61) and (62). □

THEOREM 3.3. *For $w > 1$ and $\nu(m)$ defined by* (55),

$$\Pr\left(\frac{\|v\|}{w} \le \nu(m) \le w\|v\|\right) = g_{n,m}\left(\frac{wE_n}{E_m}\right) - g_{n,m}\left(\frac{E_n}{E_m\, w}\right). \tag{64}$$

Proof. From the above, $\nu(m) = \|v\|\phi(m)E_m/E_n$, so

$$\begin{aligned}\Pr(\nu(m) \le w\|v\|) &= \Pr\left(\|v\|\phi(m)\frac{E_m}{E_n} \le w\|v\|\right)\\ &= \Pr\left(\phi(m) \le \frac{wE_n}{E_m}\right)\\ &= g_{n,m}\left(\frac{wE_n}{E_m}\right),\end{aligned}$$

by (57) with $\epsilon = wE_n/E_m$. Replacing w by $\frac{1}{w}$ gives $\Pr(\nu(m) \le \|v\|/w) = g_{n,m}(E_n/(E_m w))$. Taken together these expressions give (64). □

As an example, for $m = 2$ with $w > 1$ and $wE_n/E_2 \le 1$,

$$\Pr\left(\frac{\|v\|}{w} \le \nu(2) \le w\|v\|\right) = \left(1-\left(\frac{E_n}{E_2 w}\right)^2\right)^{(n-2)/2} - \left(1-\left(\frac{wE_n}{E_2}\right)^2\right)^{(n-2)/2}. \tag{65}$$

Using the above and some lengthy but standard expansions, we find

$$\Pr\left(\frac{\|v\|}{w} \le \nu(1) \le w\|v\|\right) \approx 1 - \frac{2}{\pi w}, \tag{66}$$

$$\Pr\left(\frac{\|v\|}{w} \le \nu(2) \le w\|v\|\right) \approx 1 - \frac{\pi}{4w^2}, \tag{67}$$

$$\Pr\left(\frac{\|v\|}{w} \le \nu(3) \le w\|v\|\right) \approx 1 - \frac{32}{3\pi^2 w^3}, \tag{68}$$

$$\Pr\left(\frac{\|v\|}{w} \le \nu(4) \le w\|v\|\right) \approx 1 - \frac{81\pi^2}{512w^4}. \tag{69}$$

These estimates are generally very accurate for $w \ge 10$. To handle the case of smaller values of w, we use the fact that the probability that $\nu(m)$ is within a factor of $\|v\|$ decreases toward a nonzero constant as n increases to infinity [20]. These lower bounds can be calculated by applying the Richardson extrapolation [3], [29] to the exact probability expressions given by Theorem 3.3 above for large values of n. For example, in calculating the lower bounds given in Table 1, we used $n = 10000$ and $n = 20000$.

Before turning to some numerical examples, we list here for reference the key estimators defined above:

ζ averaged statistical estimate,

$\zeta(m)$ averaged statistical estimate with m samples,

ν subspace statistical estimate,

$\nu(m)$ subspace statistical estimate with m samples.

4. Numerical examples.

4.1. Generating random vectors. Let $\tilde{z}_1, \ldots, \tilde{z}_n$ be normally distributed independent random variables with mean zero and variance one. Then the vector $z = \tilde{z}/\|\tilde{z}\|$ is uniformly and randomly distributed over S_{n-1} [47] where $\tilde{z} = (\tilde{z}_1, \ldots, \tilde{z}_n)^T$. Because of this, the sum of the squares of the first m components of z, which is equal to the subspace statistical variable ϕ^2 as in §3, will have a Beta distribution with $a = \frac{1}{2}$ and $b = \frac{n}{2}$ (see §26.5 in [1]). That is, for $0 \le x \le 1$,

$$\Pr(\phi^2 \le x) = \frac{B(a, b, x)}{B(a, b, 1)}, \tag{70}$$

where

$$B(a, b, x) = \int_0^x t^{a-1} (1-t)^{b-1}\, dt. \tag{71}$$

This means that the theory of §3 could have been developed by using

$$\begin{aligned} \Pr(\phi^2 \le x) &= \Pr(\phi \le \sqrt{x}) \\ &= \frac{B(a, b, \sqrt{x})}{B(a, b, 1)}. \end{aligned}$$

As in the discussion at the beginning of §3, to get a basis for a subspace of dimension m that is selected uniformly and randomly from the set of all subspaces of dimension m in $\mathbb{R}^n$, it is sufficient to generate m independent vectors $z^{(1)}, \ldots, z^{(m)}$ uniformly and randomly on S_{n-1} as above and then find an orthonormal basis for these vectors by using, say, a Gram–Schmidt procedure or a QR decomposition [16].

Remark 5. It is worth mentioning that independent normally distributed random variables can be generated by the following procedure: let r_1 and r_2 be independent random variables with uniform distribution in (0,1) and define $s = \sqrt{2\ln(1/r_1)}$ and $\theta = 2\pi r_2$. Then $\tilde{z}_1 = s\cos(\theta)$ and $\tilde{z}_2 = s\sin(\theta)$ are independent normally distributed random variables with mean zero and variance one.

4.2. Scalar examples. *Example* 1. Suppose that p is a polynomial of degree $n-1$: $p(t) = p_0 + p_1 t + \cdots + p_{n-1}t^{n-1}$, with roots $t_1, \ldots, t_{n-1}$. If p is perturbed to the form $p_\epsilon \equiv p + \epsilon q$ where $q(t) = q_0 + q_1 t + \cdots + q_{n-1}t^{n-1}$, then the roots become functions of ϵ. From the theory of algebraic functions [17], a root of multiplicity k at $\epsilon = 0$ gives rise to k roots that are analytic functions of $\epsilon^{1/k}$ in a neighborhood of $\epsilon = 0$. Thus, an isolated root, say t_1, will be differentiable at $\epsilon = 0$, and we may differentiate the relation

$$0 = p(t_1(\epsilon)) + \epsilon q(t_1(\epsilon))$$

to get

$$\frac{dt_1(0)}{d\epsilon} = -\frac{q(t_1(0))}{p'(t_1(0))} = \nabla t_1 z,$$

where $z \equiv (q_0, q_1, \ldots, q_{n-1})^T$ and

$$\nabla t_1 = \left(\frac{\partial t_1}{\partial q_i}\right) = \left(-\frac{1}{p'(t_1(0))}, -\frac{t_1(0)}{p'(t_1(0))}, \ldots, -\frac{t_1^{n-1}(0)}{p'(t_1(0))}\right).$$

See [48] for more details. We may apply the statistical method to this problem by considering the n polynomial coefficients as the input variables $x = (p_0, \ldots, p_{n-1})$ and the scalar function $f = f(x)$ to be the value of the root t_1. The sensitivity of the root is then the norm of the gradient of f at x: $v \equiv \nabla t_1^T$. The above equation shows that the exact value of $\|v\|$ is given by

$$\|v\| = \frac{1}{|p'(t_1(0))|}\sqrt{1 + t_1^2(0) + \cdots + t_1^{2(n-1)}(0)}.$$

For example, consider the polynomial

$$\begin{aligned} p(t) &= (t-1)(t-2)\cdots(t-8) \\ &= 40320 - 109584t + 118124t^2 - 67284t^3 + 22449t^4 - 4536t^5 + 546t^6 - 36t^7 + t^8, \end{aligned}$$

and suppose that we are interested in the sensitivity of the root $t = 6$. This root is moderately sensitive with $\|v\| \approx 7097.7$. Sample estimates of $\|v\|$ are given in Table 3 for three different runs (all results are rounded to integer values). Of course, three runs is too small a number to be statistically meaningful, but Table 3 does give some feel for how the estimates perform. From Table 2, we see that the variance of $\nu(m)$ is smaller than the variance of $\zeta(m)$ for $m = 2$ and $m = 3$; this is illustrated by the greater spread in the values of the averaged statistical estimator in Table 3.

TABLE 3
Comparison of the two estimators for Example 1.

m	Run number	$\zeta(m)$	$\nu(m)$	Exact value of $\|v\|$
1	1	20883	20883	7098
1	2	3372	3372	7098
1	3	2130	2130	7098
2	1	6673	10047	7098
2	2	7374	9743	7098
2	3	2588	10726	7098
3	1	3147	9642	7098
3	2	11632	10183	7098
3	3	1609	6398	7098

Example 2. This example has an unexpected twist to it since it focuses on a problem in which we want to know when the condition number is near zero. More specifically, a pair of matrices (A, B) has an uncontrollable eigenvalue at λ if λ is an eigenvalue of $A + BF$ for any compatibly dimensioned matrix F [50]. In general, however, it is not easy to test numerically for uncontrollability [12], [39]. Because of this, it is more common to use the distance to uncontrollability [33], which we denote by $\rho_c(A, B)$:

$$\rho_c(A, B) \equiv \min\{\, \|(\Delta A, \Delta B)\| \;:\; (A + \Delta A, B + \Delta B) \text{ is uncontrollable}\}. \tag{72}$$

It seems reasonable to expect that if (A, B) has an eigenvalue λ that is nearly uncontrollable, then the sensitivity of the map f defined by $x \mapsto \lambda(A+BF)$, where $x = \text{vec}(F)$, should be nearly zero. Here we use the symbol $\text{vec}(F)$ to denote the vector formed by stacking the columns of F [19]. In fact, if we assume that λ is an isolated eigenvalue of A, then by using standard perturbation arguments [49] it can be shown that the norm of the gradient of f gives an upper bound on $\rho_c(A, B)$. As an example, let A be a 10×10 matrix with main diagonal entries $a_{i,i} = i$ and ones on the first superdiagonal $a_{i,i+1} = 1$ and zeros elsewhere. If we let $B = \text{diag}\left(1, 10^{-1}, \ldots, 10^{-9}\right)$, then the feedback matrix F is 10×10 and the number of input parameters is $n = 100$. The matrix B for this problem offers decreasing control authority for the higher states; in fact, the norm of the gradient of the eigenvalue $\lambda = 10$ is 1.5×10^{-9}. That is, $\lambda = 10$ is nearly uncontrollable. Table 4 shows that this insensitivity is easily detected.

TABLE 4
Comparison of the two estimators for Example 2.

m	Run number	$\zeta(m)$	$\nu(m)$	Exact value of $\|v\|$
1	1	1.6×10^{-9}	1.6×10^{-9}	1.5×10^{-9}
1	2	4.4×10^{-10}	4.4×10^{-10}	1.5×10^{-9}
1	3	2.7×10^{-9}	2.7×10^{-9}	1.5×10^{-9}
2	1	2.6×10^{-9}	2.3×10^{-9}	1.5×10^{-9}
2	2	1.2×10^{-9}	2.8×10^{-9}	1.5×10^{-9}
2	3	7.8×10^{-9}	1.1×10^{-9}	1.5×10^{-9}
3	1	5.2×10^{-10}	1.3×10^{-9}	1.5×10^{-9}
3	2	6.7×10^{-10}	1.6×10^{-9}	1.5×10^{-9}
3	3	1.3×10^{-9}	3.1×10^{-9}	1.5×10^{-9}

4.3. Matrix examples. The preceding sensitivity theory of scalar functions can be extended to differentiable vector-valued functions. By using the vec operator [19], which maps matrices into vectors by stacking the columns, the set of vector-valued functions can be viewed as including the class of functions that map matrices into matrices. For example, the exponential map $X \mapsto e^X$ is equivalent to $x \mapsto f(x)$, where $x = \text{vec}(X)$ and $f(x) = \text{vec}(e^X)$.

If f is a twice continuously differentiable function that takes $\mathbb{R}^n$ into $\mathbb{R}^q$, then the Taylor expansion about $x \in \mathbb{R}^n$ has the form

$$f(x + \delta z) = f(x) + \delta L z + O(\delta^2), \tag{73}$$

where L is a matrix of size $q \times n$. The classical approach [40] to measuring the sensitivity of f at x is to use the norm of L to bound the perturbations in f:

$$\|f(x + \delta z) - f(x)\| \leq \delta \|L\| + O(\delta^2), \tag{74}$$

where we assume that z has unit norm.

For the matrix 2-norm, the norm of L can be estimated by using the power method to approximate the largest eigenvalue of $L^T L$. This method was used in [6] for linear systems and subsequently extended to a variety of other problems [4], [5], [15], [31], [34], [35]. Related estimation methods for other matrix norms can be found in [21], [26], and [27]. The idea behind these methods is that the matrix L is usually not known explicitly (or is too expensive to evaluate directly), but it is possible to evaluate $w = Lz$ for a given vector z either by solving an associated linear system or by using difference approximations. The same applies to L^T, although, in some cases such as problems associated with differential equations [31], the computation of the transpose vectors $L^T w$ is harder than the computation of Lz (see Example 4).

The transpose step of the power method also runs into difficulty when the function f maps between spaces of different dimensions, i.e., when L is not a square matrix. For example, if f maps $\mathbb{R}^n$ into $\mathbb{R}$, then L is $1 \times n$ and Lz can be evaluated by using a finite difference. However, the transpose step involves forming $L^T w$, as described below, and since L^T is $n \times 1$, this cannot be evaluated without knowing all the entries of L^T! Thus the power method is impractical for this problem because finding the entries of L (and hence L^T) via finite differences would require n extra function evaluations. In fact, it was this difficulty that originally inspired the research in this paper.

In general, if the Fréchet derivative of f is known only implicitly (i.e., if we can evaluate Lz but we do not know L explicitly) then the power method is limited to estimating the condition number of maps between spaces of the same dimension, although it is sometimes possible to get around this by working with restricted maps. For example, in considering the effect of perturbations in the coefficient matrices (A, F, G) for a solution X of an algebraic Riccati equation

$$0 = G + A^T X + XA - XFX,$$

Byers [5], and subsequently Kenney and Hewer [31], found it convenient to consider separately the three equal-dimensional maps $A \mapsto X$, $F \mapsto X$, and $G \mapsto X$, rather than to work with the complete map $(A, F, G) \mapsto X$. Power method estimates from the three equal-dimensional maps can then be combined to get upper and lower bounds on the condition of the complete map (see [31]).

The power method estimation of $\|L\|$ starts with an initial vector $z^{(1)}$ and then forms $z^{(2)} = L^T L z^{(1)}$ via two steps: $w^{(1)} = Lz^{(1)}$ and $z^{(2)} = L^T w^{(1)}$. The norm of L is then estimated by $\sqrt{\|z^{(2)}\|/\|z^{(1)}\|}$, and better estimates can be obtained by repeating the power iteration with $z^{(1)}$ replaced by $z^{(2)}$. The quality of the estimation is affected by the choice of the initial vector, and most schemes rely on either a "look-ahead" procedure for selecting $z^{(1)}$ (for example, [4], [6]) or use randomly generated initial vectors [31]. In general, the power method gives very good estimates of $\|L\|$ (usually within a factor of 10 or so) with only a few power iterations. Probabilistic results concerning the power method can be found in [10] and [36]. However, for brevity we omit a detailed discussion of these results, which are based on bounds derived from the Beta distribution.

One of the strengths of the power method is that for many problems the formation of $w = Lz$ is much cheaper than the initial evaluation of f. For example, in solving linear systems, the initial factorization needed to find $f(x)$ can be used in forming $w = Lz$ with considerable savings (see Example 3). Similarly, for any problem in which $f(x)$ is the solution to a problem that can be computed via a Newton method, the evaluation of $f(x + \delta z)$ is cheap in comparison to the effort needed to evaluate $f(x)$, because we can use $f(x)$ as a starting value in solving for $f(x + \delta z)$. For example, this is the case in solving for the roots of polynomials or more generalized problems such as solving an algebraic Riccati equation [32]. Since the matrix statistical methods described in this paper are also based on vectors of the form $w = Lz$, they inherit the efficiency of the power method estimators. Moreover, because the statistical methods do not require the domain and codomain (i.e., the space into which a function maps) spaces to be of equal dimension, they are more broadly applicable than the power method.

The 2-norm approach to conditioning is convenient because all of the sensitivity information about f at x is compressed into the scalar value $\|L\|$. However, we do pay a penalty for this compression since a large value of $\|L\|$ can disguise the fact that some of the entries of f may be insensitive to perturbations while others are not. For this reason, it makes sense to consider separately the sensitivity of each entry of f, especially since this entails no greater computational effort.

Close inspection of (73) reveals that the ith entry of f, which we denote by f_i, satisfies

$$f_i(x + \delta z) = f_i(x) + \delta v_i^T z + O(\delta^2), \tag{75}$$

where v_i^T is the ith row of L. In fact, the ith row of L is simply the gradient of the scalar function f_i. Because of this, we may apply the preceding results for scalar functions to the problem of estimating the norm of v_i^T for $1 \leq i \leq q$.

For example, to find the averaged statistical condition estimate for f at x, we need to evaluate f at $x + \delta z$, where z is selected uniformly and randomly from S_{n-1}. Next, form the Newton difference vector

$$w \equiv \frac{f(x + \delta z) - f(x)}{\delta} \tag{76}$$

for δ small and positive. Then the absolute value of the ith entry of w divided by E_n is the averaged statistical condition estimate for the gradient of f_i:

$$\zeta_i = \frac{|w_i|}{E_n}. \tag{77}$$

Repeating this procedure m times and then averaging the results gives the mth-order averaged statistical condition estimates.

Similarly, to form the subspace statistical estimates for $m \geq 1$, let $z^{(1)}, \ldots, z^{(m)}$ be orthonormal vectors chosen so that their span is uniformly and randomly selected from the space of all m-dimensional subspaces of $\mathbb{R}^n$, as described at the beginning of this section. Then evaluate $f(x + \delta z^{(i)})$ and form the Newton difference vectors

$$w^{(i)} \equiv \frac{f(x + \delta z^{(i)}) - f(x)}{\delta} \tag{78}$$

for $1 \leq i \leq m$. The subspace condition estimator for f_i is then given by

$$\nu_i(m) \equiv \frac{E_m}{E_n} \sqrt{\left(w_i^{(1)}\right)^2 + \cdots + \left(w_i^{(m)}\right)^2}. \tag{79}$$

Remark 6. If it is easier to form Lz directly rather than the Newton differences, then (76) should be replaced by $w = Lz$ and (78) should be replaced by $w^{(i)} = Lz^{(i)}$. This is illustrated in the following example.

Example 3. Consider the problem of solving a linear system of the form $Af = x$, where A and x are known and we want to find the vector $f = f(x)$. Since $f(x) = A^{-1}x$ and $f(x + \delta z) = f(x) + \delta A^{-1}z$, we see that $L = A^{-1}$ in (73). This means that the Newton difference approximation vectors $w^{(i)}$ in (76) and (78) can be found by solving

$$Aw^{(i)} = z^{(i)}. \tag{80}$$

The importance of this is that if the initial solution to $Af = x$ is carried out by using, say, an LU factorization, then the vector $w^{(i)}$ can be found in $O(n^2)$ flops. Since the LU factorization of A requires $O(n^3)$ flops, the condition estimation vectors $w^{(i)}$ can be computed very efficiently relative to the initial function evaluation. Similar remarks apply to the standard power method condition estimates that require the solution of $Aw^{(1)} = z^{(1)}$ followed by $A^T z^{(2)} = w^{(1)}$.

To illustrate this, let A be an Ostrowski matrix of order n [38]. That is, A is an upper triangular matrix with -1 on the main diagonal and all of the upper entries equal to 1. Even though all of the eigenvalues of A are equal to -1, this matrix is nearly singular if n is large,

with smallest singular value on the order of 2^{-n+1}. If we take $n = 30$, then the sensitivity of the first entry of the solution vector f is $\|\nabla f_1\| \approx 3.1 \times 10^8$. The sensitivity of the remaining entries decreases by a factor of about two per entry down to $\|\nabla f_{30}\| \approx 1.0$. Both the averaged and subspace statistical condition estimators perform well on this problem. For example, out of 100 runs for the subspace estimation method with $m = 2$, the greatest overestimation was 2.76 times the corresponding true condition number, and the worst underestimation was 0.057 for all 30 entries of the solution vector. For $m = 3$ the maximum overestimation out of 100 runs was a factor of 1.74 and the worst underestimation was a factor of 0.297.

Example 4. Although it simplifies the presentation of the underlying theory, it is not necessary to use the vec operator formulation to calculate the condition estimates. For example, consider the problem of estimating the sensitivity of the matrix exponential map $X \mapsto e^X$ where X is of order n_1. The number of input variables is then $n = n_1^2$.

The basic idea is to perturb the entries of X, find the exponential of the perturbed X, and then subtract the unperturbed exponential and divide by δE_n. The result is a matrix each of whose entries mirrors the sensitivity of the corresponding entry of e^X. For $m > 1$, this is repeated m times and the entries are combined to get the sensitivity estimate, either by averaging the absolute values of the entries in the case of the averaged statistical method, or by taking E_m times the square root of the sum of the squares of the entries in the case of the subspace statistical method.

More specifically, let $z^{(1)}, \ldots, z^{(m)}$ be orthonormal vectors of length $n = n_1^2$ such that their span is uniformly and randomly selected from the space of all m-dimensional subspaces of $\mathbb{R}^n$, as described at the beginning of this section. Now form the $n_1 \times n_1$ matrices $Z^{(i)} = \operatorname{unvec}(z^{(i)})$ by "unstacking" the vectors $z^{(i)}$. That is, $z^{(i)} = \operatorname{vec}(Z^{(i)})$. The matrix version $W^{(i)} = \operatorname{unvec}(w^{(i)})$ of the difference vectors $w^{(i)}$ in (78) is given by

$$W^{(i)} = (e^{X+\delta Z^{(i)}} - e^X)/\delta. \tag{81}$$

We then define the subspace condition estimators for each entry of e^X by

$$v_{ij}(m) \equiv \frac{E_m}{E_n} \sqrt{\left(W_{ij}^{(1)}\right)^2 + \cdots + \left(W_{ij}^{(m)}\right)^2}, \tag{82}$$

which is the matrix analog of (79). The averaged statistical estimators can be found by using the matrix analog of (77).

As a specific example, let

$$A = \begin{bmatrix} -131 & 19 & 18 \\ -390 & 56 & 54 \\ -387 & 57 & 52 \end{bmatrix},$$

which is Test Case 3 from [46]. This example showed that the exponential condition estimator in [46] can give conservative estimates that may be off by many orders of magnitude. For this problem, the estimator in [46] indicated that the computed exponential was accurate to at least four digits, whereas the actual number of accurate digits was 12. The conditioning of this problem was subsequently treated by a finite difference power method described in [34] that gave an accurate condition estimate at the cost of two extra exponential evaluations.

To see how well the statistical condition estimators work on this problem, let us first compute the exact answer for comparison. Let $|\nabla e^X|$ denote the matrix whose (i, j) entry is equal to the sensitivity of the (i, j) entry of e^X as measured by the norm of the gradient of the map $X \mapsto \left(e^X\right)_{ij}$. This matrix can be computed using finite differences

$$|\nabla e^X| = \begin{bmatrix} 39.9 & 8.4 & 5.2 \\ 146 & 30.8 & 18.9 \\ 136 & 28.7 & 17.5 \end{bmatrix}.$$

The matrix exponential is thus seen to be rather well conditioned for this example with the most sensitivity in the (2,1) position and the least sensitivity in the (1,3) position.

Using one extra function evaluation ($m = 1$) with the averaged statistical method gives the following estimate of $|\nabla e^X|$:

$$|\nabla e^X|_{\text{est}} = \begin{bmatrix} 25.0 & 4.7 & 3.7 \\ 96.7 & 18.4 & 14.2 \\ 79.8 & 14.9 & 12.0 \end{bmatrix}.$$

While this result is generally representative of the first-order estimates for this problem, it is probably better from a reliability standpoint to use more functions evaluations. For example, in 1000 runs using the subspace method with $m = 2$, we found that the ratio of estimated to true condition numbers ranged from a low of 0.011 to a high of 2.2. For $m = 3$, this range was 0.10 to 1.8.

The computational effort needed for the exponential condition estimates can actually be reduced by about half by using some results from [34]. In that work, it was shown that for a given initial matrix Z, the corresponding value of the matrix W in (81) is given by the integral representation of the exponential Fréchet derivative

$$W = \int_0^1 e^{(1-s)X} Z e^{sX} ds \; + \; O(\delta). \tag{83}$$

This integral can then be accurately approximated via the trapezoidal method by setting

$$\Omega_0 = \left(e^{X/2^p} Z + Z e^{X/2^p}\right)/2^{p+1} \tag{84}$$

and for $j = p, p-1, \ldots, 1$,

$$\Omega_{p+1-j} = \left(e^{X/2^j} \Omega_{p-j} + \Omega_{p-j} e^{X/2^j}\right). \tag{85}$$

Here p is the number of squarings used in the "scaling and squaring" method of evaluating e^X [46], and we assume that $\|e^{X/2^p} - I\| < \frac{1}{4}$ to ensure that Ω_p is equal to W to within a relative accuracy of 95% (see [34] for more details). In the scaling and squaring method, $e^{X/2^p}$ is approximated by using, say, a Padé approximation and the result is then squared p times to get e^X. This means that the matrices $e^{X/2^j}$ for $j = p, p-1, \ldots, 0$ are available sequentially, and the matrices Ω_{p+1-j} in (85) can be evaluated in step with the squaring. If we write

$$\Omega_{p+1-j} = \left(\left(e^{X/2^j} + \delta \Omega_{p-j}\right)^2 - e^{X/2^{(j-1)}} \right)/\delta \; + O(\delta), \tag{86}$$

then we can evaluate $\Omega_0, \ldots, \Omega_p$ in $p+1$ matrix multiplies. (The correspondence between (85) and (86) was discovered by noting that (85) is simply the Fréchet derivative of the matrix squaring map.) If we wish, we may evaluate as many matrices $W^{(i)}$ as we like in parallel with the squaring steps. Thus to get a subspace condition estimate of order m requires about $m(p+1)$ matrix multiplies. On the other hand, the effort to evaluate e^X is about $8 + p$ matrix multiplies. As noted in [34], a typical value of p is 6, in which case the subspace condition

estimate of order m requires about the same effort as $\frac{m}{2}$ function evaluations. Hence, the above procedure, which might be called a trapezoid-subspace method, is twice as efficient as the finite difference method described in [34]. The trapezoid-subspace method is also about twice as fast as the trapezoid-power method of [34] because of the sequential nature of the power method. The reason for this is that the power method iterates on the series in (85), and after the first pass through (85), which corresponds to the step $w = Lz$ described above in the vec formulation, the matrices $e^{X/2^j}$ must be regenerated on each pass through (85). Thus the trapezoid-power method uses $p+1$ matrix multiplies on the first pass through (85) and then $2p+1$ matrix multiplies on each subsequent pass. Here we assume that due to the variable nature of the number of squarings, a general purpose evaluator of e^X will not be set up to store all the matrices $e^{X/2^p}, e^{X/2^{p-1}}, \ldots, e^X$. If these matrices are stored, then the cost of the power method is the same as the subspace method.

We end this example by noting that Mathias [37] has shown that the trapezoidal approximation to the exponential Fréchet derivative can be extended to higher-order approximations, such as Simpson's rule, and that, except for the definition of Ω_0, the result is the same as (85). This means that these higher-order methods can also be combined with the subspace statistical method with the same efficiency as the trapezoid-subspace method.

Remark 7. So far we have been concerned with absolute rather than relative sensitivity, since the corresponding relative condition numbers can be obtained by dividing by $\|f(x)\|$ for overall relative sensitivity or by $|f_i(x)|$ for the relative sensitivity of the individual entries of f. However, there is some justification for considering other approaches. For example, suppose that we are interested in measuring the sensitivity of a function that takes two variables (x_1, x_2) into two values (f_1, f_2): $f(x_1, x_2) = (f_1(x_1, x_2), f_2(x_1, x_2))$, at the point $x_1^0 = 2$ and $x_2^0 = 100$. If we select a random perturbation on the unit circle of the form $(\cos\theta, \sin\theta)$, then the perturbed values $(x_1^0 + \cos\theta, x_2^0 + \sin\theta)$ can show up to 50% variation in the first component and only 1% variation in the second component. To correct this imbalance we can rescale the independent variables by defining new variables $y_1 = x_1/x_1^0$ and $y_2 = x_2/x_2^0$. We then work with the rescaled function $g(y) = f(x)$, i.e., $g_i(y_1, y_2) \equiv f_i(y_1 x_1^0, y_2 x_2^0)$ for $i = 1, 2$. Then $g(1, 1) = f(x_1^0, x_2^0)$ and random perturbations of y about $(1, 1)$ affect each component with the same level of variation. Using the chain rule we see that the Fréchet derivative of g at $(1, 1)$, which we denote by L_g, is equal to the Fréchet derivative L_f of f at (x_1^0, x_2^0) multiplied by the diagonal scaling matrix $D = \operatorname{diag}(x_1^0, x_2^0)$:

$$L_g = L_f D. \tag{87}$$

These ideas extend to higher dimensions and lead to what is usually called the mixed condition number [2], [9], [14], [25], [41]:

$$m(f, x^0) = \frac{\|L_f D\|}{\|f(x^0)\|}. \tag{88}$$

Similarly, if we wish to get relative sensitivity estimates for the individual entries of f, we can rescale g to form h where $h_i(y) = g_i(y)/f_i(x^0)$. This double scaling gives $h(1, 1, \ldots, 1) = (1, 1, \ldots, 1)$ and the Fréchet derivative of h at $(1, 1, \ldots, 1)$ is

$$L_h = F^{-1} L_f D, \tag{89}$$

where $F = \operatorname{diag}(f_1(x^0), f_2(x^0), \ldots)$. Taking the norm of L_h gives what is called the componentwise condition number of f at x^0:

$$c(f, x^0) = \|F^{-1} L_f D\|. \tag{90}$$

For a nice discussion of these concepts for linear systems see [24].

At least from the point of view of statistical condition estimates, we may work directly with the functions g or h using finite differences to approximate the absolute values of the entries of the Fréchet derivative matrices L_g or L_h, respectively. The only difference between this and the earlier statistical condition estimation is that we first scale the variables x to get g and scale both x and f to get h.

Example 5. Let $V(x)$ be the Vandermonde matrix associated with the vector x: $V_{ij} = x_j^{i-1}$. If we want to measure the mixed sensitivity at x^0, then we work with the scaled matrix function $G(y)$ where $G_{ij} = (y_j x_j^0)^{i-1}$ and measure the sensitivity of G about $y^0 = (1, 1, \ldots, 1)$. That is, we perturb y^0, evaluate G at the perturbed point, and then take the divided differences with the unperturbed matrix $G(y^0)$, as described above. This gives us a matrix whose entries are estimates of the corresponding 2-norms of the rows of L_g in (87). The results for the case $x^0 = (1, 2, 3, 4, 5, 6)$ are given in Table 5 which shows the lowest ratios of the estimated versus the exact mixed condition numbers taken over all of the entries of the matrix. (All of the high ratios were less than 3.0.) The low ratios for $\nu(1)$ reflect the fact that these ratios are taken over all 36 entries of G; in effect we are doing 36 scalar condition estimates. If this were a Bernoulli experiment with the number of trials t equal to 36 times the number of runs, say $t = 36r$, with a success being that the estimate is within a factor w of the true condition number, then by (66) the expected number of failures $E_f(t)$ for $\nu(1)$ in t trials would be approximately

$$E_f(t) = t\,\frac{2}{\pi w}. \tag{91}$$

For example, if we take $w = 3125 = 1/(3.2 \times 10^{-4})$ as in the second column of Table 5 with $r = 100$ runs, then the left-hand side of (91) gives the the expected number of failures as 0.73.

TABLE 5
Lowest ratios of estimated to exact mixed condition numbers.

Number of runs	Lowest ratios for $\nu(1)$	Lowest ratios for $\nu(2)$	Lowest ratios for $\nu(3)$
10	5.7×10^{-3}	0.0617	0.1699
100	3.2×10^{-4}	0.0209	0.0686
1000	6.8×10^{-5}	0.0097	0.0377

This heuristic argument points to an important conclusion: if we want the condition estimates of each entry of a matrix function to have a high probability of being near its respective true value, then we should use higher-order methods. The important question is then, how high an order do we need? While the Bernoulli trials reasoning used above has some drawbacks, a good argument can be given that it represents a worst-case scenario. If so, then the expected number of failures in r runs of a system with l function values is approximately bounded by $E_f(t, \nu(i))$, where $t = r \times l$ and

$$E_f(t, \nu(1)) \approx t\,\frac{2}{\pi w}, \tag{92}$$

$$E_f(t, \nu(2)) \approx t\,\frac{\pi}{4w^2}, \tag{93}$$

$$E_f(t, \nu(3)) \approx t\,\frac{32}{3\pi^2 w^3}, \tag{94}$$

$$E_f(t, \nu(4)) \approx t\,\frac{81\pi^2}{512w^4}. \tag{95}$$

More research is needed on this question, but for problems with a very large number of function values, we are usually interested in the sensitivity of some specific entries, in which case the above is not a problem, or in an overall estimate of conditioning, in which case an estimate of the norm of the Fréchet derivative is appropriate. In the latter case, the usual power method can be used, or the Frobenius norm of the Fréchet derivative can be estimated from the Frobenius norm of the individual sensitivity estimates [20]. This is illustrated in Table 6 for the mixed condition number estimates in the Vandermonde example. Here the norm estimates are remarkably accurate even for a large number of runs.

TABLE 6
Lowest ratios of estimated to exact Fréchet derivative Frobenius norm.

Number of runs	Lowest ratios for $\nu(1)$ Frob. norm est.	Lowest ratios for $\nu(2)$ Frob. norm est.	Lowest ratios for $\nu(3)$ Frob. norm est.
10	0.5560	0.6874	0.7448
100	0.3569	0.5235	0.6103
1000	0.2471	0.3838	0.4485

5. Conclusion. An analysis has been given of a new condition estimation method for differentiable matrix functions that is based on measuring the effect of random perturbations on the argument of the function. By using m such function evaluations, a condition estimate of order m is obtained, i.e., the probability that the estimate is off by a factor w is less than a constant times $1/w^m$. For example, using three function evaluations gives an estimate that has a chance of less than 1 in 7402 of being off by more than a factor of 20. Moreover, the "transpose-free" nature of this new method allows it to be applied to a broad range of problems in which the dimension of the domain and codomain spaces may be different. This is in sharp contrast to the more common power method condition estimation procedure, which is only applicable to maps between spaces of equal dimension. Illustrative examples have been given for a variety of problems including the classical problem of estimating the condition number of the matrix exponential map, as well as for problems involving mixed and componentwise condition estimation.

6. Appendix. This appendix gives a short discussion of the expected value function E_n and its evaluation for large n. For small n the value of E_n can be computed easily via (7) or (8). However, for larger values of n it is not convenient to use these formulas, and we would like to have simple approximations to E_n. There are at least two ways to obtain such approximations.

The first uses the relation

$$E_n\, E_{n+1} = \frac{2}{\pi n}, \tag{96}$$

which follows directly from (7) and (8). As noted previously (see the remark following Theorem 3.1), E_n decreases with n, i.e.,

$$E_n > E_{n+1}. \tag{97}$$

Together (96) and (97) imply the following.

LEMMA 6.1. *For E_n defined by* (7) *and* (8),

$$\sqrt{\frac{2}{\pi n}} < E_n < \sqrt{\frac{2}{\pi(n-1)}} \tag{98}$$

and

$$E_n = \sqrt{\frac{2}{\pi(n-\frac{1}{2})}} + O\left(n^{-3/2}\right). \tag{99}$$

Proof. By (96) and (97),

$$\frac{2}{\pi n} = E_n\, E_{n+1} < E_n^2.$$

Taking square roots of both sides gives

$$\sqrt{\frac{2}{\pi n}} < E_n.$$

Similarly, by replacing n with $n-1$ in (96), we find

$$\frac{2}{\pi(n-1)} = E_{n-1}\, E_n > E_n^2,$$

which completes the proof of (98). Standard arguments then give (99) from (98). □

The approximation

$$E_n \approx \sqrt{\frac{2}{\pi(n-\frac{1}{2})}} \tag{100}$$

is usually adequate for the purposes of condition estimation. Higher-order asymptotic approximations can be derived by using a multiplicative correction to the right-hand side of (100) of the form

$$\eta(n) = \left(\Sigma a_i n^{-i}\right)^{-1/2}.$$

The coefficients can then be chosen to satisfy the relationship (96) up to a given order of $\frac{1}{n}$ and to possibly improve the fit for small values of n as well. For example, a fourth-order approximation of this type is given by

$$E_n \approx \sqrt{\frac{2}{\pi(n-\frac{1}{2})}}\sqrt{\frac{184n^4}{1+23n+23n^2+184n^4}}, \tag{101}$$

which has a relative error of less than 10^{-5} for $n \geq 2$.

A second method of approximation involves writing E_n in terms of the gamma function [1], where $\Gamma(n) = (n-1)!$ for positive integers n

$$E_n = \frac{1}{2^{n-1}}\,\frac{\Gamma(n)}{\Gamma^2\left(\frac{n+1}{2}\right)}. \tag{102}$$

The right-hand side is then approximated by using

$$\Gamma(n) \approx e^{-n}n^n\sqrt{\frac{2\pi}{n}}\left(1+\frac{1}{12n}+\frac{1}{288n^2}-\frac{139}{51840n^3}-\frac{571}{2488320n^4}\right). \tag{103}$$

(This approximation, which is almost universally referred to as "Stirling's approximation," is apparently (see p. 2 in [7]) due to Laplace instead!) Using (103) in (102) gives an approximation to E_n with about the same accuracy as (101).

REFERENCES

[1] M. ABRAMOWITZ AND I. STEGUN, *Handbook of Mathematical Functions*, Dover Publications, New York, 1965.

[2] M. ARIOLI, J. DEMMEL, AND I. DUFF, *Solving sparse linear systems with sparse backward error*, SIAM J. Matrix Anal., 10 (1989), pp. 165–190.

[3] K. E. ATKINSON, *An Introduction to Numerical Analysis*, Wiley, New York, 1978.

[4] R. BYERS, *A LINPACK-style condition estimator for the equation* $AX - XB^T = C$, IEEE Trans. Auto. Control, AC-29 (1984), pp. 926–928.

[5] ———, *Numerical condition of the algebraic Riccati equation*, in Linear Algebra and Its Role in Systems Theory, Vol. 47, Contemp. Math., 1985, pp. 35–49.

[6] A. K. CLINE, C. B. MOLER, G. W. STEWART, AND J. H. WILKINSON, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368–375.

[7] E. COPSON, *Asymptotic Expansions*, Cambridge University Press, London, 1965.

[8] J. DEMMEL, *On condition numbers and the distance to the nearest ill-posed problem*, Numer. Math., 51 (1987), pp. 251–289.

[9] J. DEMMEL AND W. KAHAN, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 873–912.

[10] J. D. DIXON, *Estimating extremal eigenvalues and condition numbers of matrices*, SIAM J. Numer. Anal., 20 (1983), pp. 812–814.

[11] C. EDWARDS, *Advanced Calculus of Several Variables*, Academic Press, New York, 1973.

[12] R. EISING, *Between controllable and uncontrollable*, Systems Control Lett., 4 (1984), pp. 263–264.

[13] W. FELLER, *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd ed., Wiley, New York, 1968.

[14] I. GOHBERG AND I. KOLTRACHT, *On the inversion of Cauchy matrices*, in Signal Processing, Scattering and Operator Theory, Proc. Internat. Symposium MTNS-89, Vol. III, J. van Schuppen and A. Ran, eds., 1990, pp. 381–392.

[15] G. H. GOLUB, S. NASH, AND C. F. VAN LOAN, *A Hessenberg–Schur Method for the Problem* $AX + XB = C$, IEEE Trans. Auto. Control, AC-24 (1979), pp. 909–913.

[16] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983; 2nd ed., 1989.

[17] E. GOURSAT, *A Course in Mathematical Analysis*, Vol. II, Part 1, Dover Publications, New York, 1959.

[18] I. GRADSHTEYN AND I. RYZHIK, *Table of Integrals, Series, and Products*, 4th ed., Academic Press, New York, 1965.

[19] A. GRAHAM, *Kronecker Products and Matrix Calculus with Applications*, Wiley, New York, 1981.

[20] T. GUDMUNDSSON, C. KENNEY, AND A. J. LAUB, *Small-sample statistical estimates for matrix norms*, manuscript.

[21] W. HAGER, *Condition estimates*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 311–316.

[22] J. HAMMERSELY AND D. HANDSCOMB, *Monte Carlo Methods*, Methuen and Wiley, London, New York, 1964.

[23] G. A. HEWER AND C. KENNEY, *The sensitivity of the stable Lyapunov equation*, SIAM J. Control Optim., 26 (1988), pp. 321–344.

[24] D. J. HIGHAM AND N. J. HIGHAM, *Componentwise perturbation theory for linear systems with multiple right-hand sides*, Linear Algebra Appl., 174 (1992), pp. 111–129.

[25] N. HIGHAM, *Error analysis of the Bjork-Pereyra algorithms for solving Vandermonde systems*, Numer. Math., 50 (1987), pp. 613–632.

[26] ———, *Algorithm* 674: *FORTRAN codes for estimating the one-norm of a real or complex matrix, with application to condition estimation*, ACM Trans. Math. Software, 14 (1988), pp. 381–396.

[27] ———, *Experience with a matrix norm estimator*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 804–809.

[28] A. JAMES, *Normal multivariate analysis and the orthogonal group*, Ann. Math. Statist., 25 (1954), pp. 40–75.

[29] D. JOYCE, *Survey of extrapolation processes in numerical analysis*, SIAM Rev., 13 (1971), pp. 435–490.

[30] B. KÅGSTRÖM, *Bounds and perturbation bounds for the matrix exponential*, BIT, 17 (1977), pp. 39–57.

[31] C. KENNEY AND G. HEWER, *The sensitivity of the algebraic and differential Riccati equations*, SIAM J. Control Optim., 28 (1990), pp. 50–69.

[32] C. KENNEY AND A. J. LAUB, *Efficient componentwise condition estimates for Riccati problems*, manuscript.

[33] ———, *Controllability and stability radii for companion form systems*, Math. Control Signals Systems, 1 (1988), pp. 239–256.

[34] ———, *Condition estimates for matrix functions*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 191–209.

[35] ———, *Polar decomposition and matrix sign function condition estimates*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 488–504.

[36] J. KUCZYŃSKI AND H. WOŹNIAKOWSKI, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1094–1122.

[37] R. MATHIAS, *Evaluating the Fréchet derivative of the matrix exponential*, Numer. Math., 63 (1992), pp. 213–226.

[38] A. M. OSTROWSKI, *On the spectrum of a one-parametric family of matrices*, Journal für die reine und angewandte Mathematik, band 193, heft 3/4, (1954), pp. 143–160.

[39] C. PAIGE, *Properties of numerical algorithms relating to uncontrollability*, IEEE Trans. Auto. Control, AC-26 (1981), pp. 130–138.

[40] J. R. RICE, *A theory of condition*, SIAM J. Numer. Anal., 3 (1966), pp. 287–310.

[41] R. SKEEL, *Scaling for numerical stability in Gaussian elimination*, J. Assoc. Comput. Mach., 26 (1979), pp. 494–526.

[42] A. STAM, *Limit theorems for uniform distributions on spheres in high-dimensional Euclidean spaces*, J. Appl. Probab., 19 (1982), pp. 221–228.

[43] G. W. STEWART, *The efficient generation of random orthogonal matrices with an application to condition estimators*, SIAM J. Numer. Anal., 17 (1980), pp. 403–409.

[44] A. STROUD, *Approximate calculation of multiple integrals*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[45] C. F. VAN LOAN, *The sensitivity of the matrix exponential*, SIAM J. Numer. Anal., 14 (1977), pp. 971–981.

[46] R. WARD, *Numerical computation of the matrix exponential with accuracy estimate*, SIAM J. Numer. Anal., 14 (1977), pp. 600–610.

[47] G. WATSON, *Statistics on Spheres*, Wiley, New York, 1983; Vol. 6 in the Univ. of Arkansas Lecture Notes in the Mathematical Sciences.

[48] J. H. WILKINSON, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[49] ———, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

[50] W. M. WONHAM, *Linear Multivariable Control: A Geometric Approach*, 2nd ed., Springer-Verlag, New York, 1979.

[51] S. ZAREMBA, *The Mathematical Basis of Monte Carlo and quasi-Monte Carlo methods*, SIAM Rev., 10 (1968), pp. 303–314.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 62–76, January 1994

004

THE DAVIDSON METHOD*

M. CROUZEIX†, B. PHILIPPE‡, AND M. SADKANE§

Abstract. This paper deals with the Davidson method that computes a few of the extreme eigenvalues of a symmetric matrix and corresponding eigenvectors. A general convergence result for methods based on projection techniques is given and can be applied to the Lanczos method as well. The efficiency of the preconditioner involved in the method is discussed. Finally, by means of numerical experiments, the Lanczos and Davidson methods are compared and a procedure for a dynamic restarting process is described.

Key words. Davidson method, Lanczos method, Krylov space, preconditioner, eigenvalue, sparse matrices, eigenvectors

AMS subject classification. 65F15

1. Introduction.

To compute a few of the extreme eigenvalues and the corresponding eigenvectors of a large, sparse, and symmetric matrix, two classes of iterative methods are usually considered. Their common characteristic is to build a sequence of subspaces that contains, in the limit, the desired eigenvectors. The subspaces of the first class are of constant dimension; this class includes simultaneous iteration [1] and the trace minimization method [11]. In the second class of methods, the sequence is increasing, at least piecewise, since there often exists a restarting process that limits the dimension of the subspaces to a feasible size; the class includes the well-known Lanczos method which is based on Krylov subspaces [6]. This paper deals with another method of the same class, namely, the Davidson method.

Davidson published his algorithm in the quantum chemistry field [2] as an efficient way to compute the lowest energy levels and the corresponding wave functions of the Schrödinger operator. The original algorithm that computes the largest (or the smallest) eigenvalue of the matrix A can be expressed by the following algorithm where D stands for the diagonal of the matrix A.

Choose an initial unit vector v_1; $V_1 := [v_1]$;
for $k = 1, \ldots$ **do**
  Compute the interaction matrix $H_k := V_k^t A V_k$;
  Compute the largest (or the smallest) eigenpair (μ_k, y_k) of H_k;
  Compute the corresponding Ritz vector $x_k := V_k y_k$;
  Compute the residual $r_k := (\mu_k I - A)x_k$;
  if convergence **then** exit;
  Compute the new direction to be incorporated $t_{k+1} := (\mu_k I - D)^{-1} r_k$;
  Orthogonalize the system $[V_k; t_{k+1}]$ into V_{k+1};
end for

This algorithm looks like an algorithm of the Lanczos type with a diagonal preconditioning. When the dimension of the basis V_k becomes too large, the process restarts with the last Ritz vector as initial vector. In this paper we consider a more general method in the sense that

*Received by the editors November 19, 1990; accepted for publication (in revised form) February 16, 1993.

†Institut de Recherche en Informatique et Systèmes Aleatoires/University of Rennes, Campus de Beaulieu, 35042 Rennes Cedex, France (crouzeix@irisa.fr).

‡Institut de Recherche en Informatique et Systèmes Aleatoires-Institut National de Recherche en Informatique et en Automatique, Campus de Beaulieu, 35042 Rennes Cedex, France (philippe@irisa.fr).

§Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, 42 Av. G. Coriolis, 31057 Toulouse Cedex, France. Present address: Institut de Recherche en Informatique et Systèmes Aleatoires-Institut National de Recherche en Informatique et en Automatique, Campus de Beaulieu, 35042 Rennes Cedex, France (sadkane@irisa.fr).

- several eigenpairs are sought at the same time;
- several vectors are incorporated in the basis at every step, leading to a block implementation;
- a general preconditioner is considered.

The block adaptation is important with supercomputers since it allows parallelism and efficient use of local memory.

Before analyzing the Davidson method, we formulate, in §2, a general convergence result for methods based on projection techniques; it can be applied to the Lanczos process as well. Consequences for the Davidson method are described in §3. Section 4 is devoted to a discussion on selecting the preconditioner and on the class of matrices on which the algorithm is the method of choice. In §5, numerical experiments illustrate the study and an improvement for the restarting process is proposed.

Notations and general assumptions. $A = (a_{ij})_{1\le i,\ j\le n}$ is a symmetric matrix supposedly large and sparse; $\lambda_1 \ge \cdots \ge \lambda_n$ are its eigenvalues and $u_1, \ldots, u_n$ a corresponding set of eigenvectors such that $u_i^t u_j = \delta_{ij}$ (Kronecker's symbol) for $1 \le i,\ j \le n$. The goal consists in computing the l ($l \ll n$) largest (or smallest) eigenpairs of A.

Throughout this paper, the symbol $\| \ . \ \|$ denotes the Euclidean norm and MGS stands for the modified Gram–Schmidt procedure. The orthogonal complement of the subspace spanned by the vectors $x_1, \ldots, x_k$ is denoted by $\{x_1, \ldots, x_k\}^\perp$.

$\rho(x) = x^t Ax/\|x\|^2$ is the Rayleigh quotient of the vector $x \ne 0$ and $R(x_1, \ldots, x_k) = \max_{x \in \mathrm{Span}(x_1,\ldots,x_k)} \rho(x)$ is the maximum of the Rayleigh quotient over the space spanned by the vectors $x_1, \ldots, x_k$.

$\{\mathcal{V}_k\}$ is a sequence of subspaces of $\mathbf{R}^n$ of dimension $n_k \ge l$ and V_k is a matrix whose column set is an orthonormal basis of $\mathcal{V}_k$. The matrix $H_k = V_k^t A V_k$ is called the Rayleigh or interaction matrix; it is of order n_k and its l largest eigenvalues are $\lambda_{k,1} \ge \cdots \ge \lambda_{k,l}$ with the corresponding eigenvectors $y_{k,1}, \ldots, y_{k,l}$, which constitute an orthonormal set of vectors in $\mathbf{R}^{n_k}$. The corresponding Ritz vectors $x_{k,1}, \ldots, x_{k,l}$ are defined by $x_{k,i} = V_k y_{k,i}$ for $i = 1, \ldots, l$. The reals $\lambda_{k,1}, \ldots, \lambda_{k,l}$ are called the Ritz values of A over $\mathcal{V}_k$.

2. Proof of convergence.

THEOREM 2.1. *Under the assumption*

$$x_{k,i} \in \mathcal{V}_{k+1} \quad \text{for } i = 1, \ldots l \quad \text{and} \quad k \in \mathbf{N},$$

the sequences $\{\lambda_{k,i}\}_{k\in\mathbf{N}}$ *are nondecreasing and convergent.*

Moreover, if, in addition, (i) *for any* $i = 1, \ldots, l$ *the set of matrices* $\{C_{k,i}\}_{k\in\mathbf{N}}$ *satisfies the following assumption: there exist* $K_1,\ K_2 > 0$ *such that for any* $k \in \mathbf{N}$ *and for any vector* $v \in \mathcal{V}_k^\perp$: $K_1 \| v \|^2 \le v^t C_{k,i} v \le K_2 \| v \|^2$;

(ii) *for any* $i = 1, \ldots, l$ *and* $k \in \mathbf{N}$, *the vector* $(I - V_k V_k^t) C_{k,i} (A - \lambda_{k,i} I) x_{k,i}$ *belongs to* $\mathcal{V}_{k+1}$, *then the limit* $\mu_i = \lim_{k\to\infty} \lambda_{k,i}$ *is an eigenvalue of* A *and the accumulation points of* $\{x_{k,i}\}_{k\in\mathbf{N}}$ *are corresponding eigenvectors.*

Proof. The first statement is a straight application of the well-known Courant–Fischer theorem [6] that characterizes the eigenvalues of a symmetric operator.

Let us prove the second statement. Let $r_{k,i} = (\lambda_{k,i} I - A) x_{k,i}$ and $w_{k,i} = (I - V_k V_k^t) C_{k,i} r_{k,i}$. Since the Ritz vectors are unit vectors and since $r_{k,i} = -(I - V_k V_k^t) A x_{k,i}$, the residuals $r_{k,i}$ belong to $\mathcal{V}_k^\perp$ and are uniformly bounded by $\|A\|$; hence the vectors $w_{k,i}$ are uniformly bounded as well. Moreover, since

$$w_{k,i}^t A x_{k,i} = -r_{k,i}^t C_{k,i} r_{k,i}, \tag{1}$$

and since the matrix $C_{k,i}$ is assumed to be positive definite on $\mathcal{V}_k^\perp$, we may ensure that $w_{k,i} = 0$ if and only if $r_{k,i} = 0$.

When $w_{k,i} \neq 0$, let us denote $v_{k,i} = w_{k,i} / \| w_{k,i} \|$ and $\Pi_k = [x_{k,1}, \ldots, x_{k,i}, v_{k,i}]$. Π_k is an $n \times (i+1)$ matrix whose columns are orthonormal. Consequently, the matrix $\Pi_k \Pi_k^t$ corresponds to the orthogonal projection onto a subspace of $\mathcal{V}_{k+1}$.

The matrix $\mathcal{H}_{k,i} = \Pi_k^t A \Pi_k$ has the following pattern:

$$\begin{pmatrix} \mu_{k,1} & & & \alpha_{k,1} \\ & \ddots & & \vdots \\ & & \mu_{k,i} & \alpha_{k,i} \\ \alpha_{k,1} & \ldots & \alpha_{k,i} & \beta_k \end{pmatrix},$$

where $\alpha_{k,j} = x_{k,j}^t A v_{k,i}$ for $j = 1, \ldots, i$ and $\beta_k = v_{k,i}^t A v_{k,i}$.

Let $\lambda_{k,1} \geq \lambda_{k,2} \geq \cdots \geq \lambda_{k,i} \geq \lambda_{k,i+1}$ be the eigenvalues of $\mathcal{H}_{k,i}$. Cauchy's interlace theorem and the optimality of the Rayleigh–Ritz procedure [6] ensure that

$$\mu_{k,j} \leq \lambda_{k,j} \leq \mu_{k+1,j}, \qquad j = 1, \ldots, i\ .$$

The Frobenius norm of the matrix $\mathcal{H}_{k,i}$ is

$$\sum_{j=1}^{i} \lambda_{k,j}^2 + \lambda_{k,i+1}^2 = 2 \sum_{j=1}^{i} \alpha_{k,j}^2 + \beta_k^2 + \sum_{j=1}^{i} \mu_{k,j}^2;$$

therefore

$$2 \sum_{j=1}^{i} \alpha_{k,j}^2 = \sum_{j=1}^{i} (\lambda_{k,j} - \mu_{k,j})(\lambda_{k,j} + \mu_{k,j}) + (\lambda_{k,i+1} - \beta_k)(\lambda_{k,i+1} + \beta_k).$$

Evaluating the trace of the matrix $\mathcal{H}_{k,i}$ by $\sum_{j=1}^{i} \mu_{k,j} + \beta_k = \sum_{j=1}^{i+1} \lambda_{k,j}$, we obtain

$$\begin{aligned} 2 \sum_{j=1}^{i} \alpha_{k,j}^2 &= \sum_{j=1}^{i} (\lambda_{k,j} - \mu_{k,j})(\lambda_{k,j} + \mu_{k,j} - \lambda_{k,i+1} - \beta_k) \\ &\leq 4 \| A \| \sum_{j=1}^{i} (\lambda_{k,j} - \mu_{k,j}), \end{aligned}$$

which implies

$$\alpha_{k,p}^2 \leq 2 \| A \| \sum_{j=1}^{i} (\mu_{k+1,j} - \mu_{k,j}) \quad \text{for} \quad p = 1, \ldots, i.$$

This last bound proves that $\lim_{k \to \infty} \alpha_{k,p} = 0$ for $p = 1, \ldots, i$. Therefore, since from (1), we have the relation

$$r_{k,i}^t C_{k,i} r_{k,i} = - \| w_{k,i} \| \alpha_{k,i}$$

so that $\lim_{k \to \infty} r_{k,i}^t C_{k,i} r_{k,i} = 0$. From the assumption of uniform positive definiteness of $C_{k,i}$ over ${\mathcal{V}_k}^{\perp}$, and since $r_{k,i} \in {\mathcal{V}_k}^{\perp}$, we may conclude that $\lim_{k \to \infty} r_{k,i} = 0$.

Let x_i be an accumulation point of the sequence $\{x_{k,i}\}$; then $\| x_i \| = 1$. From the definition of $r_{k,i}$, we obtain by continuity that $\mu_i x_i - A x_i = 0$. □

A straightforward application of the theorem may be obtained for a well-known version of the Lanczos method, namely, the block version with restarting process as defined in [8]. From

an initial block S of l vectors that constitute an orthonormal set, the matrix V_k is recursively built in such a way that its columns form an orthonormal basis of the Krylov space which is spanned by the columns of $S, AS, \ldots, A^{k-1}S$; this is done while $kl \leq m$, where m is a fixed maximum dimension. The Rayleigh matrix H_k, which is built from V_k, is a block tridiagonal matrix. When kl is larger than m, the process restarts with a new block S that corresponds to the Ritz vectors found with the last matrix V_k. Then, we claim the following corollary.

COROLLARY 2.2. *The block version of the Lanczos algorithm, used with restarting, converges.*

Proof. The Lanczos method corresponds to the situation where $C_{k,i}$ is the identity matrix and where $\mathcal{V}_k$ is the Krylov subspace generated from the block V_1. Therefore Theorem 2.1 may be applied. □

3. The generalized Davidson method.

3.1. Algorithm. The following algorithm computes the l largest (or smallest) eigenpairs of the matrix A; m is a given integer that limits the dimension of the basis.

Choose an initial orthonormal matrix $V_1 := [v_1, \ldots, v_l] \in \mathbf{R}^{n \times l}$;

for $k = 1, \ldots$ **do**

1. Compute the matrix $W_k := AV_k$;
2. Compute the Rayleigh matrix $H_k := V_k^t W_k$;
3. Compute the l largest (or smallest) eigenpairs $(\lambda_{k,i}, y_{k,i})_{1 \leq i \leq l}$ of H_k;
4. Compute the Ritz vectors $x_{k,i} := V_k y_{k,i}$ for $i = 1, \ldots, l$;
5. Compute the residuals $r_{k,i} := \lambda_{k,i} x_{k,i} - W_k y_{k,i}$ for $i = 1, \ldots, l$;
   if convergence **then** exit;
6. Compute the new directions $t_{k,i} := C_{k,i} r_{k,i}$ for $i = 1, \ldots, l$;
7. **if** $\dim(V_k) \leq m - l$
   then $V_{k+1} := \mathrm{MGS}(V_k, t_{k,1}, \ldots, t_{k,l})$;
   else $V_{k+1} := \mathrm{MGS}(x_{k,1}, \ldots, x_{k,l}, t_{k,1}, \ldots, t_{k,l})$;
   end if

end for

Steps (1)–(5) correspond to the classical Rayleigh–Ritz procedure [6]. We point out that only the last columns of W_k and H_k have to be computed at iteration k. At each iteration, the vectors $t_{k,i}$ are incorporated into the previous subspace. Unlike the Lanczos method, the Rayleigh matrix is dense.

The block size can be greater than l, and this may give faster convergence [3]. Since orthogonalization is performed at every iteration, too large a dimension for the basis implies prohibitive complexity. This is the reason for setting a maximum size for the basis. In §6, a dynamic choice for the restart point is described based on an index of efficiency for the iteration.

The selection of efficient preconditioners $C_{k,i}$ is studied in §4. As remarked in Corollary 2.2, the method becomes equivalent to the Lanczos method when the matrices $C_{k,i}$ are proportional to the identity matrix I. However, since in the Davidson method it is necessary to compute the Ritz vectors explicitly at every iteration, this version of the Lanczos algorithm has a much more expensive complexity than the regular version.

In the classical Davidson method, the preconditioners are built from the diagonal D of the matrix A: $C_{k,i} = (\lambda_{k,i} I - D)^{-1}$, which exists when $\lambda_{k,i}$ is not a diagonal entry of A. This choice is efficient when D is a good approximation of the matrix A in the sense that the matrix of eigenvectors of A is close to the identity matrix. More general preconditioners $C_{k,i} = (\lambda_{k,i} I - M)^{-1}$ have already been studied [5]; as for any preconditioning process,

the tradeoff consists in finding a matrix M that speeds up the convergence and keeps the complexity of the preconditioning step at a reasonable level. Finally, we mention that a block version of the Davidson method has already been introduced in [4].

Remark. It can be proved [10] that the accumulation points H of the sequence $\{H_k\}$ are of the form

$$H = \begin{pmatrix} \theta_1 & & & \mathbf{0} \\ & \ddots & & \\ & & \theta_l & \\ \mathbf{0} & & & E \end{pmatrix},$$

where $\theta_1 \geq \cdots \geq \theta_l$ are the l largest eigenvalues of H. Therefore, under the assumption that none of the θ_i, $i = 1, \ldots, l$ is an eigenvalue of the matrix E, the components of the corresponding eigenvectors of H are zero along the second block. As pointed out by Davidson [2], this fact can be used in practice to measure the convergence.

3.2. Convergence. In this section we assume that a diagonal preconditioner is used, i.e., $C_{k,i} = (\lambda_{k,i} - D)^{-1}$ for $i = 1, \ldots, l$, where D is the diagonal of A. We assume also that we require the largest eigenpair of A. The situation is analyzed in two different ways depending on the number of eigenpairs needed. The end of the section is devoted to an example of nonconvergence when the hypotheses of Theorem 2.1 are not satisfied.

3.2.1. Classical algorithm ($l = 1$). Theorem 2.1 ensures the convergence of the Davidson method when $(\lambda_{k,1} - D)^{-1}$ is positive definite. Since the sequence $\{\lambda_{k,1}\}$ is nondecreasing, it is sufficient to start with a vector v_1 such that $(\mu_{1,1} - D)^{-1}$ is positive definite. This can be ensured in the following way:

- Let i_o be the index of the largest diagonal entry of D. If the problem is not reducible into two smaller problems, there exists an index j_o such that $a_{i_o,j_o} \neq 0$.
- Let V_1 be the system $[e_{i_o}, e_{j_o}]$ built from the corresponding canonical vectors.

Since the matrix $H_1 = V_1^t A V_1$ is the matrix

$$\begin{pmatrix} a_{i_o,i_o} & a_{i_o,j_o} \\ a_{i_o,j_o} & a_{j_o,j_o} \end{pmatrix},$$

we have $\mu_{1,1} > a_{i_o,i_o} = \max_{1 \leq i \leq n} a_{i,i}$. In conclusion, the following bounds are obtained:

$$\|C_{k,1}\| \leq \frac{1}{\mu_{1,1} - a_{i_o,i_o}},$$
$$v^t C_{k,1} v \geq \alpha \|v\|^2 \text{ with } \alpha = \frac{1}{\max_{1 \leq i \leq n} (\mu_{1,1} - a_{i,i})}.$$

Hence Theorem 2.1 can be applied.

3.2.2. Block version ($l \neq 1$). The technique defined in the previous case can be used here to ensure that $(\lambda_{k,1} - D)^{-1}$ is positive definite; therefore the convergence is certain for the first eigenpair, but not for the others. However, it is possible to define another preconditioner $C_{k,i} = \operatorname{diag}(\mu_{k,i,1}, \ldots, \mu_{k,i,n})$ by $\mu_{k,i,j} = \min(|\lambda_{k,i} - a_{j,j}|^{-1}, M)$, where M is some large constant. With this preconditioning procedure, convergence is guaranteed for any initial system V_1.

Remark. The above choice of M is made only for the sake of the completion of the proof. In practice, the algorithm works regardless of this assumption.

3.2.3. Example of possible nonconvergence. The following example shows the importance of the assumption of positive definiteness for the preconditioning matrices. Let us assume that we look for the two largest eigenpairs ($l = 2$) of the matrix

$$A = \begin{pmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & -4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

and that the process is initialized by $V_1 = [v_1, v_2]$, where

$$v_1 = \left(\sqrt{\tfrac{7}{8}}, \ \sqrt{\tfrac{1}{8}}, \ 0, \ 0, \ 0 \right)^t,$$

$$v_2 = \left(0, \ 0, \ \sqrt{\tfrac{3(5-\sqrt{5})}{20}}, \ \tfrac{\sqrt{3\sqrt{5}-5}}{2}, \ \sqrt{\tfrac{3(5-2\sqrt{5})}{10}} \right)^t.$$

We use diagonal preconditioner, i.e., $C_{k,i} = (\lambda_{k,1} - D)^{-1}$ and restart if $m \geq 4$. A straightforward computation shows that $(\lambda_{k,1}, x_{k,1}) = (3, v_1)$ and $(\mu_{k,2}, x_{k,2}) = (\frac{1}{2}, v_2)$ for all k, although neither 3 nor $\frac{1}{2}$ are eigenvalues of A. Of course, it is clear that the assumption of positive definiteness of the preconditioning matrices is violated.

Remarks. 1. Convergence of the Davidson method is automatic if restarting is not used, because eventually the subspace V_k fills up. The significance of Theorem 2.1 is that it proves convergence even when restarting is used.

2. Even when the sequence $\{r_{k,i}\}$ of the residuals converges to zero, it is not clear that the limit $\mu_i = \lim_{k\to\infty} \lambda_{k,i}$ is the ith eigenvalue of A, since we may create situations where the subspaces $\mathcal{V}_k$ remain orthogonal to a required eigenvector. However, it can be proved [10] that this situation would be unstable and is unlikely to happen in finite arithmetic; it may only increase the number of iterations significantly.

4. Quality of the preconditioner. In this section, we restrict the study to the case $l = 1$. We assume also that $C_{k,i} = C(\lambda_{k,i})$, where $C(\mu)$ satisfies a Lipschitz condition in a neighbourhood of the first eigenvalue of A. This is the situation when $C(\mu) = (\mu I - M)^{-1}$ with M symmetric with eigenvalues smaller than the largest eigenvalue of A.

Since $l = 1$ we replace the index $(k, 1)$ by k in the algorithm. To simplify the notations, we denote by λ, λ', and $\lambda_{\min}$ the first, second, and last eigenvalue of A, respectively. Let x be the eigenvector corresponding to λ ($\lambda \geq \mu_k > \lambda'$ is assumed). Let θ_k be the angle $\angle(x, x_k)$. We may write $x_k = \alpha_k x + \beta_k y_k$, where $\alpha_k = \cos(\theta_k)$, $\beta_k = \sin(\theta_k)$, and where y_k is a unit vector orthogonal to x. The first lemma relates the convergence of the sequences $\{\mu_k\}$, $\{\theta_k\}$, and $\{\|r_k\|\}$.

LEMMA 4.1. *The following relations are true*:

$$\sqrt{\frac{\lambda - \mu_k}{\lambda - \lambda_{\min}}} \leq |\sin(\theta_k)| \leq \sqrt{\frac{\lambda - \mu_k}{\lambda - \lambda'}}, \tag{2}$$

$$\frac{2\,\|r_k\|}{\sqrt{5}(\lambda - \lambda_{\min})} \leq |\sin(\theta_k)| \leq \frac{\|r_k\|}{\mu_k - \lambda'}. \tag{3}$$

Proof. The proof is based on the sin Θ theorem [6]. □

Lemma 4.2 provides an estimate for the effect of the preconditioning process within one iteration. We may define a unit vector z_k such that the system (x, y_k, z_k) is orthonormal and such that $t_k = \gamma_k x + \delta_k y_k + \sigma_k z_k$ for some scalars γ_k, δ_k, and σ_k.

LEMMA 4.2. *The preconditioning process implies that*

$$t_k = \beta_k C(\lambda)(\lambda I - A) y_k + u_k \quad \textit{where} \quad \|u_k\| = O(\beta_k^2) \tag{4}$$

and

$$0 \le \lambda - \mu_{k+1} \le K_1 (\lambda - \mu_k), \tag{5}$$

$$|\sin \theta_{k+1}| \le K_2 |\sin \theta_k|, \tag{6}$$

where

$$K_1 = \frac{(\frac{\sigma_k}{\delta_k})^2}{(\alpha_k - \frac{\beta_k \gamma_k}{\delta_k})^2 + (\frac{\beta_k \sigma_k}{\delta_k})^2} \frac{\lambda - \lambda_{\min}}{\lambda - \lambda'}, \tag{7}$$

$$K_2 = \frac{\left|\frac{\sigma_k}{\delta_k}\right|}{\sqrt{(\alpha_k - \frac{\beta_k \gamma_k}{\delta_k})^2 + (\frac{\beta_k \sigma_k}{\delta_k})^2}} \frac{\lambda - \lambda_{\min}}{\lambda - \lambda'}. \tag{8}$$

Proof. Since $t_k = C(\mu_k)(\mu_k I - A)x_k$, we may write

$$t_k = \alpha_k(\mu_k - \lambda)C(\mu_k)x + \beta_k C(\mu_k)(\mu_k I - A)y_k,$$

and therefore (2) and the Lipschitz condition on $C(\lambda)$ imply (4).

By definition, $\mu_{k+1} \equiv \rho(x_{k+1}) = R(v_1, \ldots, v_k, t_k)$. Let us consider the vector $s_k = x_k - (\beta_k/\delta_k)t_k$, which belongs to the subspace spanned by V_{k+1}. From the optimality of the Rayleigh–Ritz procedure, we have the bounds

$$\rho(x_{k+1}) \ge R(x_k, t_k) \ge \rho(s_k). \tag{9}$$

Since

$$s_k = \left(\alpha_k - \frac{\beta_k \gamma_k}{\delta_k}\right) x - \frac{\beta_k \sigma_k}{\delta_k} z_k, \tag{10}$$

we obtain from (9)

$$\rho(x_{k+1}) \ge \frac{(\alpha_k - \frac{\beta_k \gamma_k}{\delta_k})^2 \lambda + (\frac{\beta_k \sigma_k}{\delta_k})^2 z_k^t A z_k}{(\alpha_k - \frac{\beta_k \gamma_k}{\delta_k})^2 + (\frac{\beta_k \sigma_k}{\delta_k})^2}$$

$$\ge \lambda - \frac{(\frac{\beta_k \sigma_k}{\delta_k})^2}{(\alpha_k - \frac{\beta_k \gamma_k}{\delta_k})^2 + (\frac{\beta_k \sigma_k}{\delta_k})^2} (\lambda - \lambda_{\min}),$$

which implies

$$\lambda - \mu_{k+1} \le \beta_k^2 K_1 (\lambda - \lambda').$$

Since $\rho(x_k) = \alpha_k^2 \lambda + \beta_k^2 y_k^t A y_k$, we also have

$$\rho(x_k) \le \lambda - \beta_k^2 (\lambda - \lambda').$$

The relation (5) with (7) is obtained from the last two bounds.

From (2) and (5), we obtain

$$\begin{aligned}\sin^2\theta_{k+1} &\leq \frac{\lambda-\mu_{k+1}}{\lambda-\lambda'}\\ &\leq K_1\frac{\lambda-\mu_k}{\lambda-\lambda'}\\ &\leq K_1\frac{\lambda-\lambda_{\min}}{\lambda-\lambda'}\sin^2\theta_k,\end{aligned}$$

which proves the relation (6) with (8). □

The best situation, which cannot be obtained in practice, would be to find a $C(\lambda)$ that admits x as an eigenvector and therefore $\{x\}^\perp$ as an invariant subspace. If we assume

$$\| (C(\lambda)(\lambda I - A) - I) |_{\{x\}^\perp} \| = \epsilon < 1,$$

then

$$\|t_k - \beta_k y_k\| = O(\beta_k(\beta_k + \epsilon)),$$

which implies

$$\begin{aligned}\gamma_k &= O(\beta_k(\beta_k+\epsilon)),\\ \sigma_k &= O(\beta_k(\beta_k+\epsilon)),\\ \delta_k &= \beta_k + O(\beta_k(\beta_k+\epsilon)),\end{aligned}$$

and therefore

$$\begin{aligned}\frac{\gamma_k}{\delta_k} &= O(\beta_k+\epsilon),\\ \frac{\sigma_k}{\delta_k} &= O(\beta_k+\epsilon).\end{aligned}$$

From (7) and (8), we obtain the estimate

$$K_1 = \left(\frac{\sigma_k}{\delta_k}\right)^2\left(\frac{\lambda-\lambda_{\min}}{\lambda-\lambda'}\right)(1+O(\beta_k(\beta_k+\epsilon)))\,,$$

$$K_2 = \left|\frac{\sigma_k}{\delta_k}\right|\left(\frac{\lambda-\lambda_{\min}}{\lambda-\lambda'}\right)(1+O(\beta_k(\beta_k+\epsilon)))\,.$$

Note that if $\epsilon = 0$, convergence is obtained after one step, since in this case $\sigma_k = 0$ and thus x belongs to the subspace spanned by (x_k, t_k).

The usual way to define the preconditioning matrix is to consider a matrix M that approximates A and hence the matrix $C(\lambda) = (\lambda I - M)^{-1}$. Let us consider two extreme situations: $M = I$ or $M = A$. In the former case, the method becomes equivalent to the Lanczos method as has been pointed out, while in the latter case, the method fails since $t_k = x_k$ and $w_k = (I - V_k V_k^T)t_k = 0$. Therefore M has to be an approximation of A, but with its largest eigenvalue smaller than λ to ensure the positive definiteness of the matrix $(\mu_k I - M)^{-1}$ as discussed in Theorem 2.1. With such a matrix we have

$$t_k = (\mu_k I - M)^{-1}(\mu_k I - A)x_k \tag{11}$$

$$= \alpha_k(\mu_k - \lambda)(\mu_k I - M)^{-1}x + \beta_k(\mu_k I - M)^{-1}(\mu_k I - A)y_k \tag{12}$$

and

$$x_k - t_k = \alpha_k \left(I - (\mu_k - \lambda)(\mu_k I - M)^{-1}\right) x + \beta_k \left(I - (\mu_k I - M)^{-1}(\mu_k I - A)\right) y_k. \tag{13}$$

Expressions (12) and (13) show that when the Ritz pair (μ_k, x_k) starts to approximate the solution (λ, x), then, provided $(\mu_k I - M)^{-1}$ is bounded and $\left(I - (\mu_k I - M)^{-1}(\mu_k I - A)\right)|_{\{x\}^\perp}$ is small in comparison with $\left(I - (\mu_k - \lambda)(\mu_k I - M)^{-1}\right) x$, the components of t_k in the direction of x are small, whereas the other components of t_k remain about the same as those of x_k. Therefore the vector $x_k - t_k$ has small components except those in the direction of x, and hence constitutes an improvement over x_k. We expect to have an efficient preconditioner when the angle $\angle(x, z_k)$ is smaller than the angle $\angle(x, x_k)$, where $z_k = x_k - t_k$. Table 2 illustrates this point in the context of Example 5.1.

To have an easy-to-invert matrix, the appropriate choice for M may be the main diagonal of A or its tridiagonal part, when A is strongly diagonally dominant in the sense that its eigenvectors are close to the vectors of the canonical basis.

5. Experimental results and implementation.

5.1. Efficiency of the preconditioner. The usual experience is to consider that the better the preconditioner approximates the matrix, the faster is the convergence. The diagonal preconditioner is the easiest to use, but often a larger part of the matrix, as, for example, the tridiagonal part brings a better efficiency. The following example illustrates an extreme case of the benefit that may be obtained from a good preconditioner.

Example 5.1. A is the matrix of order $n = 1000$ such that

$$a_{i,j} = \begin{cases} i & \text{if} \quad i = j, \\ 0.5 & \text{if} \quad j = i+1 \text{ or } j = i-1, \\ 0.5 & \text{if} \quad (i, j) \in \{(1, n), (n, 1)\}, \\ 0 & \text{otherwise.} \end{cases}$$

Table 1 displays the sequence of the residuals corresponding to the largest eigenvalue for the Lanczos and Davidson methods with diagonal and tridiagonal preconditioning, where multA is the number of matrix-vector multiplications. The algorithm is started as discussed in §3.2.1, which means here that $V_1 = [e_n, e_1]$ is used as starting vectors for the Davidson method, and $v_1 = V_1 y_1$ where y_1 is the eigenvector corresponding to the largest eigenvalue of the matrix $V_1^T A V_1$ is the starting vector for the Lanczos method.

Table 2 shows, as claimed in §4, that the efficient preconditioner approximates much better $z_k = x_k - t_k$ to x than does x_k. The vectors x_k, t_k, and x have been defined in (12) and (13) in §4.

Unfortunately, this rule of thumb may fail when the evaluation of the quality of a preconditioner is limited to only the consideration of the norm of its difference with the original matrix. It is well known that when two matrices are close, their spectrum are also close, but not necessarily their eigenvectors. Example 5.2 illustrates such a situation.

Example 5.2. A is the matrix of order $n = 1000$ such that

$$a_{i,j} = \begin{cases} 4 & \text{if} \quad i = j, \\ -1 & \text{if} \quad j = i+1 \text{ or } j = i-1, \\ -1 & \text{if} \quad j = i+2 \text{ or } j = i-2, \\ 0 & \text{otherwise.} \end{cases}$$

TABLE 1
Sequence of residuals depending on the preconditioner (Example 5.1).

multA	Lanczos	Davidson diagonal	Davidson tridiagonal
1	0.5000000e+00	0.5000000e+00	0.5000000e+00
2	0.2702456e+00	0.1913128e+00	0.2056694e+00
3	0.2697534e+00	0.4586425e−01	0.8539853e−04
4	0.6456297e−01	0.7378828e−02	0.3830080e−12
5	0.6436916e−01	0.8900376e−03	
6	0.1032884e−01	0.8615493e−04	
7	0.1028572e−01	0.6973576e−05	
8	0.1236296e−02	0.4852756e−06	
9	0.1229767e−02	0.2962304e−07	
10	0.1185097e−03	0.1610810e−08	
11	0.1177558e−03	0.7897330e−10	
12	0.9479829e−05	0.3541615e−11	

TABLE 2
Example of efficient preconditioner (Example 5.1).

iter	$\sin\angle(x_k, x)$ (tridiagonal)	$\sin\angle(z_k, x)$ (tridiagonal)
1	0.4202633e+00	0.4526842e+00
2	0.1176235e−01	0.7403736e−03
3	0.7440150e−06	0.2458322e−10
4	0.1116130e−12	0.2096840e−12

The diagonal preconditioner is not considered since, as already stated, a constant diagonal is equivalent to no preconditioning, and therefore the Lanczos and Davidson methods become equivalent. Figure 1 plots the behaviour of the residuals corresponding to the largest eigenvalue for the Lanczos and Davidson methods with tridiagonal preconditioning. Both methods have poor convergence. For the Lanczos method, this may be explained by the smallness of the gap ratio of λ_1 [6]: $\lambda_1 - \lambda_2/\lambda_2 - \lambda_n \approx 2.33\ 10^{-8}$, whereas for the Davidson method, the poor performance of the preconditioner may be explained by the near orthogonality of the eigenvectors corresponding to the largest eigenvalue of A and the largest eigenvalue of its tridiagonal part (angle $\approx \frac{\pi}{2}$).

Example 5.3. We consider the matrix $LAP30$ from the Harwell–Boeing test collection. This matrix is of order 900, symmetric with 4322 nonzero elements in its lower part. It is originated from a nine-point discretisation of the Laplacian on the unit square with Dirichlet boundary conditions. The eigenvalues as computed by EISPACK are $\lambda_1 = \lambda_2 = 11.9590598 \geq \lambda_3 = \lambda_4 = 11.9286959 \geq \lambda_5 = \lambda_6 = 11.8784356 \geq \cdots \geq \lambda_{900} = 0.0614628$.

The diagonal of this matrix is constant. So the Davidson method with diagonal preconditioning is equivalent to no preconditioning. We computed its four largest eigenpairs using the Davidson method with tridiagonal preconditioning. In this version of Davidson's algorithm, we decided to restart whenever the maximum size of the basis reaches 40. And once an eigenvector is converged, we put it at the beginning of the basis so that all vectors are orthogonalized against it and continue with a reduced block size. Since eigenvalues/eigenvectors seldom converge at the same time, this strategy can prevent harm and additional work in the most slowly converging eigenvectors. The stopping criterion was satisfied when the residual norm of the sought eigenpair is less than 10^{-7}.

We also computed the four largest eigenpairs of LAP30 by the Lanczos method. The version of the Lanczos method used here is the Lanczos algorithm with selective orthogonalization

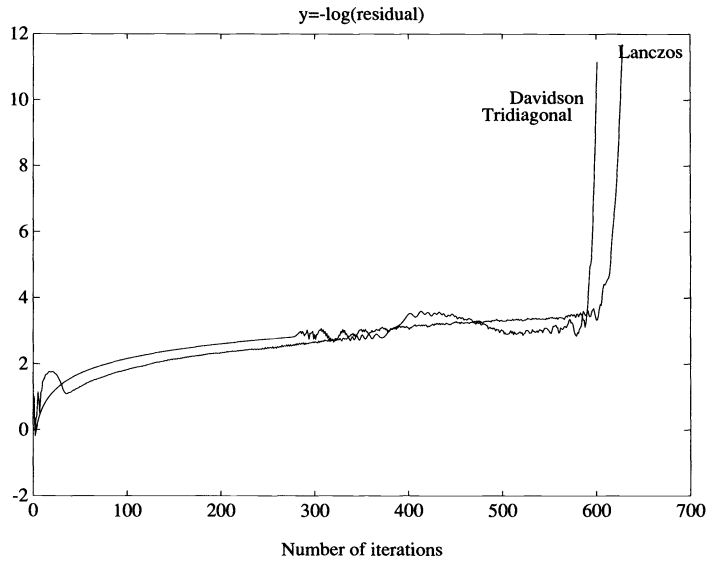


FIG. 1. *Example of a nonefficient preconditioner* (*Example* 5.2).

LASO2, developed by Parlett and Scott [8]. The blocksize, NBLOCK, for Lanczos was taken equal to four, and the number of decimal digits of accuracy desired in the eigenvalues, NFIG, was chosen equal to seven. The computations were carried out on one processor of a Cray 2 and the results are reported in Table 3. The error in eigenvalues and eigenvectors as given by Lanczos (LASO2) are also reported. Unfortunately, we do not have similar estimations for Davidson. But the residual norms and the execution times reveal that the Davidson method with tridiagonal preconditioning performs better on this example.

TABLE 3
Eigenpairs of the matrix LAP30 *by the Lanczos* (LASO2) *and Davidson methods.* (*Example* 5.3).

Method	Matrix–vector products	Time(sec)	Max residual norms	Max error in eigenvalues	Max error in eigenvectors
Lanczos	884	12.534	2.57 E−04	1.13 E−06	4.74 E−03
Davidson	607	11.487	9.36 E−08		

Example 5.4. In Table 4, we compare the Davidson method using diagonal preconditioning with the Lanczos method. (The version used is, again, the Lanczos algorithm with selective orthogonalization LASO2.) The matrix dealt with is of order 1000 and is generated randomly by setting its density of nonzero elements at 0.01. The nonzero off-diagonal entries are in the range $[-1, +1]$; the full diagonal entries are in the range [0, diagscal], where diagscal is a diagonal scaling factor to be varied. The four smallest eigenpairs are sought. The version of Davidson's algorithm uses the same technique concerning converged eigenvectors as described in Example 5.3. The stopping criterion was satisfied when the residual norm of the sought eigenpair is less than 10^{-7}. For the Lanczos algorithm, we chose the block size NBLOCK = 4, and the number of decimal digits of accuracy desired in the eigenvalues NFIG = 7. Experiments were run on a Cray 2. As expected the Davidson method becomes more efficient when the relative importance of the main diagonal increases. The efficiency of the Lanczos method is spectrum-dependent and is not sensitive to the dominance of the matrix.

TABLE 4
Davidson and Lanczos run time comparison (Example 5.4).

Diagonal factor	Time(sec)	
	Davidson	Lanczos
1	5.481	4.131
20	1.091	5.1531
40	0.844	11.574
60	0.657	9.959
80	0.537	16.062
100	0.400	11.171

5.2. Effect of the maximum size for the basis on the convergence. The easiest implementation for the restarting process consists in defining a fixed maximum size for the basis. The selection of an efficient value for m is difficult: too small a value increases the number of steps needed for convergence, whereas too large a value increases complexity and causes numerical problems.

Example 5.5 and Fig. 2 illustrate that the larger m is, the lower is the number of steps necessary to reach convergence.

Example 5.5. A is the matrix of order $n = 5000$ such that

$$a_{i,j} = \begin{cases} \text{if } i = j & \text{random in } [-10, +10], \\ \text{if } i \neq j & \begin{cases} \text{with probability } \alpha : & \text{random in } [-1, +1], \\ \text{ith probability } (1-\alpha) : & 0, \end{cases} \end{cases}$$

where $\alpha = 2 \times 10^{-3}$. There is an average of 11 nonzero entries per row. The eight largest eigenvalues ($l = 8$), that are sought, lie in the range [10.89, 11.57]. Convergence is obtained when the maximum of the L_1 norm of the residuals is smaller than $5. * 10^{-10}$. Experiments were run on an Alliant FX/80.

However, the value of m needs to be limited for three reasons.

1. The memory requirement is roughly proportional to nm, and this introduces a limit on m for large matrices.

2. The orthogonality of V_k is poorly maintained when the number of vectors in V_k is high (a loss of orthogonality plagues the convergence).

3. The complexity of the computation that is involved in one iteration increases with the number of vectors in V_k; therefore it may become too high compared to the benefit obtained from the decrease of the residual norms.

Actually, to be efficient, it is necessary to decide dynamically when to restart the process. The first reason implies a maximum for the size of the basis, but it can be more useful to restart before that limit. The second reason concerns a loss of orthogonality that is detected by an increasing sequence of the norms of the residuals; this may signal a necessary restarting. Let us now consider the detail of the computation involved in one iteration to define some index of efficiency that should indicate when it is worthwhile to restart.

The kth step since the last restart involves l multiplications by A and l applications of the preconditioning process that are of constant complexity. It involves also for the

computation of H_k:	kl^2n	flops;
diagonalization of H_k:	$O(k^3l^3)$	flops;
computation of the Ritz vectors:	kl^2n	flops;
computation of the residuals:	kl^2n	flops;
orthogonalization process:	$2kl^2n$	flops.

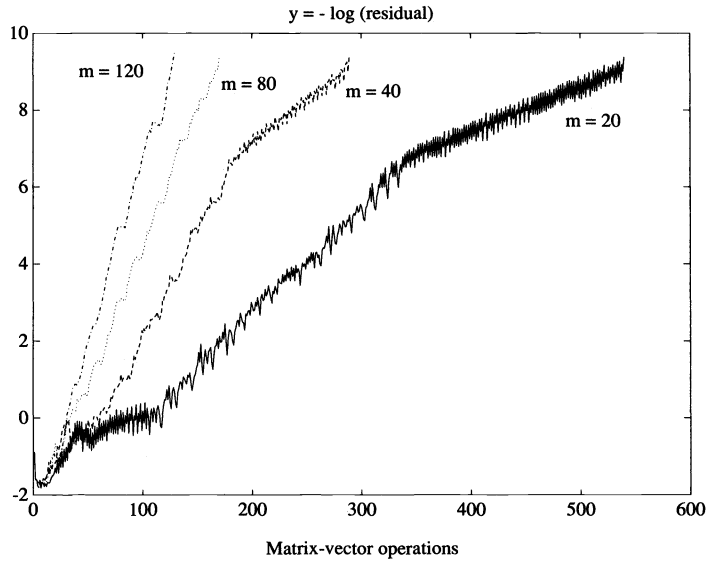


FIG. 2. *Influence of m on the convergence (Example* 5.5).

The diagonalization may be estimated as involving approximately $2k^3l^3$ flops. Let us denote by $C(k)$ the complexity involved at each iteration and by $\tau_k = \|R_k\|/\|R_{k-1}\|$ the local rate of convergence, where R_k stands for the matrix $[r_{k,1}, \ldots, r_{k,l}]$. The index of efficiency may be defined as

$$\mathcal{E}_k = \frac{1}{C_k \tau_k}.$$

By incorporating within the code a procedure that checks the variation of $\mathcal{E}_k$, the process can be restarted as soon as the index decreases significantly.

Example 5.6. The matrix under consideration is the same as in Example 5.5, where its eight largest eigenvalues and their corresponding eigenvectors are sought. Figure 3 plots the variation of the maximum of the residuals with respect to the number of iterations using dynamic restarting. We note in Fig. 3 that the convergence is reached within 140 iterations and 13 irregular restarts, while in Fig. 2 the same convergence with restarting at exactly every 20th step is reached within 500 iterations. This may be explained by the fact that dynamic restarting allows the algorithm to restart in due time, taking into account the quality of the basis for approximating the eigenvectors and the complexity involved during the iterations. The ratio $\tau_k = \|R_k\|/\|R_{k-1}\|$ also measures the changes in rate of convergence. In Table 5, the run with this dynamic restarting procedure is compared to the runs with a static restarting procedure for six values of the maximum block size (20, 40, 60, 80, 100, 120).

The efficiency of the dynamic restarting process is clearly seen in this example, since it corresponds to obtaining the optimum size of the basis automatically.

6. Conclusion. The Davidson method can be regarded as a preconditioned version of the Lanczos method. It appears to be the preferred method for some special classes of matrices, especially those where the matrix of eigenvectors is close to the identity. Although when used with a poor preconditioner it converges slowly, the Davidson method may overcome the Lanczos method tremendously.

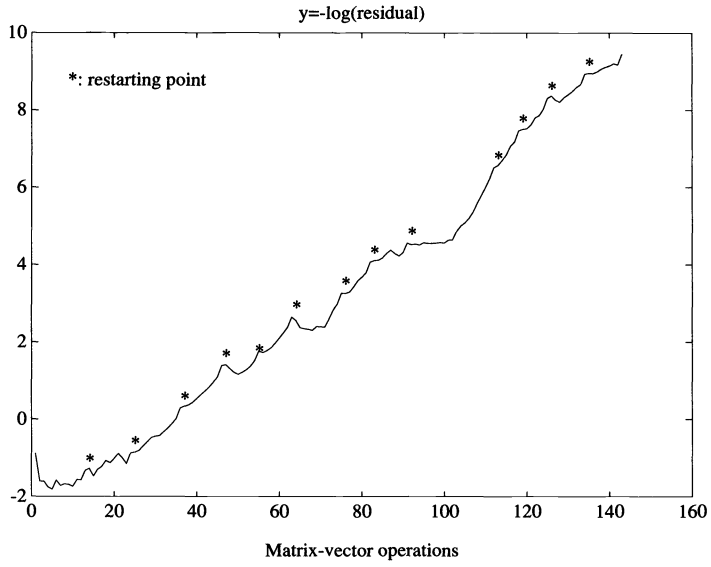


FIG. 3. *Dynamic restarting profile* (*Example* 5.6).

TABLE 5
Comparing static and dynamic restarting procedure (*Examples* 5.5 *and* 5.6).

Running times(s) with dynamic restarting	with fixed restarting	
	m	Times(s)
277.64	20	1015.94
	40	549.09
	60	334.07
	80	345.13
	100	277.83
	120	287.70

Acknowledgement. The authors would like to thank one of the referees for providing many instructive comments.

REFERENCES

[1] M. CLINT AND A. JENNINGS, *The evaluation of eigenvalues and eigenvectors of real symmetric matrices by simultaneous iteration*, Comput. J., 13 (1970), pp. 76–80.

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

[3] N. KOSUGI, *Modification of the Liu-Davidson method for obtaining one or simultaneously several eigensolutions of a large real-symmetric matrix*, Comput. Phys., 55 (1984), pp. 426–436.

[4] B. LIU, *The simultaneous expansion for the solution of several of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric eigenvectors of large real-symmetric matrices*, in Numerical Algorithms in Chemistry: Algebraic method, C. Moler and I. Shavitt, eds., LBL-8158 Lawrence Berkeley Lab., Univ. of California, 1978, pp. 49-53.

[5] R. B. MORGAN AND D. S. SCOTT, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 817–825.

[6] B. N. PARLETT, *The symmetric eigenvalue problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980.

[7] ———, *The software scene in the extraction of eigenvalues from sparse matrices*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 590–604.

[8] B. N. PARLETT AND D. S. SCOTT, *The Lanczos algorithm with selective orthogonalization*, Math. Comp., 33 (1979), pp. 217–238.

[9] B. PHILIPPE AND Y. SAAD, *Solving large sparse eigenvalue problems on supercomputers*, in Proc. Internat. Workshop on Parallel Algorithms and Architectures, Oct. 3–6, 1988, Bonas, France, North-Holland, Amsterdam, 1989.

[10] M. SADKANE, *Analyse Numérique de la Méthode de Davidson*, Ph.D thesis, Université de Rennes, France, June 1989.

[11] A. H. SAMEH AND J. A. WISNIEWSKI, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal., 19 (1982), pp. 1243–1259.

[12] D. B. SZYLD, *A Two-Level Iterative Method for Large Sparse Generalized Eigenvalue Calculations*, Ph.D thesis, Courant Institute, New York, Oct. 1983.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 77–88, January 1994

005

PRECONDITIONING CAPACITANCE MATRIX PROBLEMS IN DOMAIN IMBEDDING*

WLODEK PROSKUROWSKI† AND PANAYOT S. VASSILEVSKI‡

Abstract. Various preconditioners for the capacitance matrix problems in domain imbedding (DI) are studied. Such preconditioners can be viewed as inverses of the preconditioners for the Schur complement domain decomposition (DD) methods. Preconditioning techniques based on the hierarchical basis (HB), the square root of $-d^2/ds^2$, and on the vector probing for solving the capacitance matrix problems using preconditioned conjugate gradient iterations are analyzed and tested numerically.

Key words. capacitance matrix, domain imbedding, domain decomposition, hierarchical basis, vector-probing, second order elliptic equation, finite elements

AMS subject classifications. primary 65N30, secondary 65F10

1. Introduction.

We consider second order elliptic problems in polygonal planar domains Ω with Dirichlet boundary conditions. Polygonal domain Ω is imbedded in a rectangle R on which a fast separable elliptic solver can be used. Hence we assume that the coefficients of the elliptic operator allow separation of variables. We study various preconditioners for the capacitance matrix in domain imbedding (DI). The purpose of the DI technique is that problems on irregular regions can be solved by minor modification of the software available for standard problems (i.e., on rectangular domains) thus avoiding more complicated data structure needed for the irregular domains.

This technique was proposed by Buzbee, Dorr, George, and Golub [10] as a direct procedure and studied later as an iterative process by Proskurowski and Widlund [26], [27], O'Leary and Widlund [23], Astrakhantsev [1], [2], Dryja [14], Börgers and Widlund [7], Nepomnyaschikh [22], and others. In Börgers and Widlund [7] a detailed study of the derivation and comparison of various capacitance matrix techniques and their practical performance was presented.

Similar approaches, namely to avoid complicated data structure and to take advantage of already available software for problems on standard domains, have been used in deriving efficient preconditioning techniques for elliptic problems on grids with local refinement, cf. McCormick and Thomas [20], McCormick [21], Bramble, Ewing, Pasciak, and Schatz [8], Ewing, Lazarov, and Vassilevski [17].

The essence of the capacitance matrix technique is that the solution to the original problem can be obtained as a low order modification to the problem on the imbedding rectangle. Domain decomposition (DD) and DI are usually complementary. DI is most efficient when a domain is a small perturbation of a larger separable region, while DD is more appropriate when a domain is a union of several separable regions. In DD, one is willing to repetitively solve subdomain problems. Not so in DI because the subdomains may be irregular. Instead, one is willing to repetitively solve a larger regular problem.

One can observe a duality between the capacitance matrix technique in DI and the Schur complement methods in DD. More precisely, the preconditioners in the first method can be viewed as the inverses of the preconditioners in the second one. Thus the DI technique can be seen from a new perspective that comes from DD: using the well-developed theory of the

*Received by the editors June 30, 1992; accepted for publication (in revised form) February 11, 1993.

†Department of Mathematics, University of Southern California, DRB 155, 1042 W. 36th Place, Los Angeles, California 90089-1113 (proskuro@mathn.usc.edu).

‡Department of Mathematics, University of California at Los Angeles, 405 Hilgard Avenue, Los Angeles, California 90024-1555 (panayot@math.ucla.edu). This research was partially supported by National Science Foundation grant FDP NSF ASC 9003002.

preconditioners in DD we have immediate candidate preconditioners for the capacitance matrix problem that are straightforward to analyze. In the present paper we consider preconditioning techniques originally proposed for the DD Schur complement problems based on the square root of the one-dimensional Laplacian, on the hierarchical basis (HB), on the vector probing technique proposed by Keyes and Gropp [18] and Axelsson and Polman [3], as well as on the smoothing-correction technique proposed by Wittum [35], and apply them in the DI context.

The outline of the paper is as follows. In §2 we state the problem and derive the capacitance matrix equation following an earlier presentation by Dryja [14]. In §3 the preconditioners for the capacitance matrix are presented. Section 4 contains the numerical results for the model problem, as well as problems with anisotropic and discontinuous diffusion coefficients. Finally, in §5 some conclusions are drawn.

2. Problem formulation. Consider the following second order elliptic boundary value problem.

Given $f \in L^2(\Omega)$, find $u \in H_0^1(\Omega)$ such that

$$\mathcal{A}(u, \phi) = (f, \phi) \quad \text{for all } \phi \in H_0^1(\Omega),$$

where the open domain Ω is a planar polygon and

$$\mathcal{A}(u, \phi) = \int_\Omega \left\{ k_1(x) \frac{\partial u}{\partial x} \frac{\partial \phi}{\partial x} + k_2(y) \frac{\partial u}{\partial y} \frac{\partial \phi}{\partial y} + [\lambda_1(x) + \lambda_2(y)]\, u\phi \right\} dx\, dy. \tag{1}$$

The coefficients k_1, k_2 are assumed to be positive in Ω; and λ_1 and λ_2 are assumed to be nonnegative in Ω. Let R be a rectangle such that $R \supset \Omega$. In our model case shown in Fig. 1, Ω differs from the domain in which it is imbedded, $R = (0, 1)^2$, only along a connected segment. Generalization to more complicated domains Ω is straightforward, but its implementation may be more costly. The case of more general separable operators, nonsymmetric and indefinite, is treated in the forthcoming paper [25].

We assume that R can be triangulated on a right-angled triangular mesh $\mathcal{T}$ such that $\partial\Omega$ consists of edges of triangles from $\mathcal{T}$. This imposes some restriction on the polygonal form of Ω. A treatment of how to construct triangulations for the DI in a more general case was presented in Börgers and Widlund [7], see also Nepomnyaschikh [22]. Another possibility is to use patch local refinement in the neighborhood of the interface boundary across which Ω is imbedded in the rectangle R. Then one can use the efficient iterative elliptic solvers developed for problems on patched locally refined grids from McCormick and Thomas [20], McCormick [21], Bramble, Ewing, Pasciak, and Schatz [8], see also Ewing, Lazarov, and Vassilevski [17]. Note that these exploit fast elliptic solvers on the patches that are rectangles. However, at this point we shall not treat this subject more specifically.

So we assume that the boundary of Ω is aligned with the triangulation $\mathcal{T}$. Let W be the finite element space spanned by piecewise linear functions on $\mathcal{T}$, vanishing on ∂R and continuous in R. The subspace V of W consists of functions that have support in $\bar{\Omega}$.

Consider finally the bilinear form

$$\mathcal{B}(u, \phi) = \int_R \left\{ k_1(x) \frac{\partial u}{\partial x} \frac{\partial \phi}{\partial x} + k_2(y) \frac{\partial u}{\partial y} \frac{\partial \phi}{\partial y} + [\lambda_1(x) + \lambda_2(y)]\, u\phi \right\} dx\, dy.$$

We denote by A and B the stiffness matrices computed by the bilinear forms $\mathcal{A}(\cdot, \cdot)$ and $\mathcal{B}(\cdot, \cdot)$ using the finite element spaces V and W.

We note that problems with the matrix B can be solved very efficiently based on discrete variants of separation of variables (cf. Buzbee, Golub, and Nielson [9]), or on the odd-even

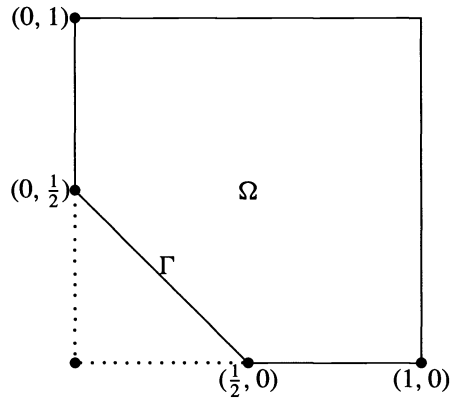


FIG. 1. *Polygon* Ω *imbedded in rectangle* $R = (0, 1)^2$.

cyclic reduction (cf. Swarztrauber [29]), or the generalized marching algorithm (cf. Bank and Rose [5] and Bank [6]). The cost is typically $O(N \log N)$ or $O(h^{-2} \log \frac{1}{h})$, where $N = O(h^{-2})$ is the number of nodes in $\mathcal{T}$ and h is the meshsize. A more refined implementation based on the fact that the right-hand sides of the problems to be solved on the rectangular imbedding domain are nonzero only on the interface boundary and based on the fact that the solution is also only needed on the interface will lead to a cost of $O(N) = O(h^{-2})$ operations, cf. Banegas [4], Kuznetsov [19], and Proskurowski [24].

The capacitance matrix method (CMM) relies on efficient solvers for problems with the matrix B. Next we derive the CMM in algebraic (i.e., matrix–vector) form following an earlier presentation of Dryja [14].

The original problem in Ω after discretization takes the form

$$A\mathbf{x}_1 = \mathbf{b}_1. \tag{2}$$

After imbedding Ω in R we consider the problem

$$B \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_0 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ \mathbf{b}_0 \end{bmatrix} \begin{matrix} \} \\ \} \\ \} \end{matrix} \begin{matrix} \Omega \equiv \Omega_1, \\ R \setminus \bar{\Omega} \equiv \Omega_2, \\ \Gamma, \end{matrix}$$

where $\Gamma = \partial\Omega \setminus \partial R$ is the interface boundary across which Ω is imbedded in R. Note that B admits the following 3×3 DD block form

$$B = \begin{bmatrix} B_{11} & 0 & B_{10} \\ 0 & B_{22} & B_{20} \\ B_{01} & B_{02} & B_{00} \end{bmatrix} \begin{matrix} \} \\ \} \\ \} \end{matrix} \begin{matrix} \Omega_1, \\ \Omega_2, \\ \Gamma. \end{matrix}$$

Finally, note that $B_{11} = A$. Consider now the following extended matrix

$$A^E = \begin{bmatrix} A & 0 & B_{10} \\ 0 & B_{22} & B_{20} \\ 0 & 0 & I \end{bmatrix} \begin{matrix} \} \\ \} \\ \} \end{matrix} \begin{matrix} \Omega_1, \\ \Omega_2, \\ \Gamma. \end{matrix}$$

The problem $A\mathbf{x}_1 = \mathbf{b}_1$ can be rewritten as

$$A^E \begin{bmatrix} \mathbf{x}_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ 0 \end{bmatrix}. \tag{3}$$

We seek $\mathbf{w}_0$ defined on Γ, such that

$$\mathbf{x}^E = B^{-1} \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ \mathbf{b}_0 \end{bmatrix} + B^{-1} \begin{bmatrix} 0 \\ 0 \\ \mathbf{w}_0 \end{bmatrix}$$

is the solution of (3). Then

$$\begin{aligned} 0 &= \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ 0 \end{bmatrix} - A^E \mathbf{x}^E \\ &= \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ 0 \end{bmatrix} - A^E \left\{ B^{-1} \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ \mathbf{b}_0 \end{bmatrix} + B^{-1} \begin{bmatrix} 0 \\ 0 \\ \mathbf{w}_0 \end{bmatrix} \right\}. \end{aligned} \tag{4}$$

Note now that

$$B = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ B_{01}B_{11}^{-1} & B_{02}B_{22}^{-1} & S \end{bmatrix} \begin{bmatrix} B_{11} & 0 & B_{10} \\ 0 & B_{22} & B_{20} \\ 0 & 0 & I \end{bmatrix} \begin{matrix} \} \ \Omega_1, \\ \} \ \Omega_2, \\ \} \ \Gamma, \end{matrix}$$

where

$$S = B_{00} - \sum_{i=1}^{2} B_{0i} B_{ii}^{-1} B_{i0}$$

is the so-called DD Schur complement of B. This shows that

$$B = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ B_{01}B_{11}^{-1} & B_{02}B_{22}^{-1} & S \end{bmatrix} A^E.$$

Hence

$$A^E B^{-1} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \star & \star & S^{-1} \end{bmatrix}, \tag{5}$$

where "$\star$," here and further, denotes submatrix or subvector that is of no immediate interest. From (4) and (5) we get

$$0 = \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ \star & \star & S^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ \mathbf{b}_0 + \mathbf{w}_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \star \end{bmatrix}.$$

Thus, by construction, we have that the first two equations of (4) are satisfied. Hence we need to choose $\mathbf{w}_0$ such that the last equation of (4) is zero; i.e.,

$$\left(B^{-1} \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ \mathbf{b}_0 \end{bmatrix} \right) \Bigg|_\Gamma + \left(B^{-1} \begin{bmatrix} 0 \\ 0 \\ \mathbf{w}_0 \end{bmatrix} \right) \Bigg|_\Gamma = 0.$$

Since

$$B^{-1} \begin{bmatrix} 0 \\ 0 \\ \mathbf{w}_0 \end{bmatrix} = \begin{bmatrix} \star \\ \star \\ S^{-1}\mathbf{w}_0 \end{bmatrix}, \tag{6}$$

we obtain the following equation for $\mathbf{w}_0$,

$$S^{-1}\mathbf{w}_0 = - \left(B^{-1} \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ \mathbf{b}_0 \end{bmatrix} \right) \Bigg|_\Gamma .$$

Let

$$C = S^{-1} = \left(B_{00} - \sum_{i=1}^{2} B_{0i} B_{ii}^{-1} B_{i0} \right)^{-1}. \tag{7}$$

C is called the capacitance matrix. Note that C is symmetric and positive definite and that the actions of C are readily available since for any given vector $\mathbf{v}_0$ on Γ,

$$C\mathbf{v}_0 = \left(B^{-1} \begin{bmatrix} 0 \\ 0 \\ \mathbf{v}_0 \end{bmatrix} \right) \Bigg|_\Gamma .$$

Note however that the actions of S on vectors are not generally readily available. This is because the computation of $S\mathbf{v}_0$ for a given vector $\mathbf{v}_0$ on Γ involves the action of $B_{11}^{-1} = A^{-1}$ when computing the product $B_{01}B_{11}^{-1}B_{10}\mathbf{v}_0$ (see (7)) and A is the stiffness matrix in the irregular, by assumption, original domain Ω.

We now present the algorithm for solving problem (2) based on the capacitance matrix method. For any given vector $\mathbf{z}$ we use the decomposition

$$\mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \mathbf{z}_0 \end{bmatrix} \begin{matrix} \} \ \Omega_1, \\ \} \ \Omega_2, \\ \} \ \Gamma. \end{matrix}$$

ALGORITHM CMM

(i) Solve the problem

$$B\mathbf{y} = \begin{bmatrix} \mathbf{b}_1 \\ 0 \\ \mathbf{b}_0 \end{bmatrix} \begin{matrix} \} \ \Omega_1, \\ \} \ \Omega_2, \\ \} \ \Gamma. \end{matrix}$$

(ii) Iterate for $\mathbf{w}_0$

$$C\mathbf{w}_0 = -\mathbf{y}_0.$$

(iii) Solve the problem

$$B\mathbf{v} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{w}_0 \end{bmatrix}.$$

(iv) The final solution of (2) is

$$\mathbf{x}_1 = \mathbf{y}_1 + \mathbf{v}_1.$$

3. Preconditioners for C. From now on we focus on step (ii) of Algorithm CMM solved by the preconditioned conjugate gradient (PCG) iterations and more precisely on finding a good preconditioner for the capacitance matrix C.

Recalling the relation (7) we see that any available preconditioner M_{DD} for S from the well-developed DD theory such that its action is readily computable (not necessarily the action of its inverse) will give a good preconditioner $M = M_{DD}^{-1}$ for $C = S^{-1}$.

We consider the following preconditioners.

1. *Dryja's preconditioner.* Let M be the matrix corresponding to the discretization of $-d^2/ds^2$, the second derivative along Γ. Then $M^{1/2}$ is readily available (based on the discrete FFT). In terms of preconditioning, $M^{-1/2}$ is a preconditioner that is spectrally equivalent to C. This is the case since $M^{1/2}$ is spectrally equivalent to $S = C^{-1}$. For details cf. Dryja [15], [16].

2. *Hierarchical Basis* (HB). Using the fact that (cf. Smith and Widlund [28] or Vassilevski [33])

$$\text{Cond}\left[J^T S J\right] = O(1 + \log^2 h),$$

where J is the transformation matrix from the HB to the nodal basis (on Γ only), we obtain that

$$\text{Cond}(J^{-1}S^{-1}J^{-T}) = \text{Cond}(J^{-1}CJ^{-T}) = O(1 + \log^2 h),$$

i.e., $J^{-1}CJ^{-T}$ is a much better conditioned matrix than C. Note that the actions of J^{-1} and J^{-T} are readily computable, cf., e.g., Yserentant [34].

3. *Vector probing* (A). If the coefficients of the bilinear form $\mathcal{A}(\cdot, \cdot)$ have strong variation, then, in practice, a choice of preconditioners based on algebraic vector-probing techniques might be preferable. For example, we can consider a symmetric tridiagonal matrix T defined on Γ by requiring for two vectors $\mathbf{v}_0^{(1)}$, $\mathbf{v}_0^{(2)}$ that

$$T\mathbf{v}_0^{(i)} = C\mathbf{v}_0^{(i)}, \quad i = 1, 2. \tag{8}$$

The choice of T to be tridiagonal is motivated by the fact that away from its main diagonal C has certain decay rate since S is approximately tridiagonal; for details cf. Keyes and Gropp [18], Axelsson and Polman [3], Chan and Mathew [11], and Chan, Mathew, and Shao [12]. For decay rates of inverses of band matrices, cf. Demko, Moss, and Smith [13] and Vassilevski [31]. The existence of such tridiagonal matrix T satisfying (8) is shown in Wittum [35] for a general choice of the probing vectors and in Axelsson and Polman [3] and Keyes and Gropp [18] for certain particular probing vectors. We consider a preconditioner T that corresponds to the following vectors proposed in Keyes and Gropp [18],

$$\mathbf{v}^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix}, \qquad \mathbf{v}^{(2)} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}.$$

4. *Vector probing* (B). We also choose the following preconditioner M based on the smoothing-correction technique proposed by Wittum [35] and defined by

$$M^{-1} = T_1 + T_2, \tag{9}$$

where the symmetric tridiagonal matrices T_s, $s = 1, 2$, are defined by their actions on two prescribed vectors. Namely, T_s is defined by the equations

$$\mathbf{v}_0^{(r)} = T_s C \mathbf{v}_0^{(r)}, \quad r = 1, 2,$$

for the following probing vectors:

$$\mathbf{v}_0^{(1)} = \left\{ \sin\left(\frac{\nu_s i\pi}{n_b + 1} \right) \right\}_{i=1}^{n_b},$$

$$\mathbf{v}_0^{(2)} = \left\{ \sin\left(\frac{(\nu_s + 1) i\pi}{n_b + 1} \right) \right\}_{i=1}^{n_b}.$$

Here n_b is the number of the interface nodes on Γ; ν_s for $s = 1, 2$ are properly chosen frequencies corresponding to the so-called smooth and rough components, cf. Wittum [35]. In Wittum [35] it has also been shown that there is a unique symmetric tridiagonal matrix T that has the same actions on two given vectors as the actions of another symmetric matrix on those vectors (the vectors should satisfy some linear independence property, however). We remark that preconditioner (9) is an additive version of the smoothing-correction scheme, which is a stationary iterative method as proposed in Wittum [35]. We modify it to be used as a preconditioner in the CG method.

4. Numerical experiments. We report some numerical experiments performed on a Sparc 2 station in double precision FORTRAN. A double precision version of the subroutines BLKTRI and SINT from the current version of [30] obtained via Netlib were used for solving separable problems in a rectangle and as a real FFT, respectively.

In all the experiments as the polygonal region Ω, we chose the unit square $R = (0, 1)^2$ with the corner $\{(x, y) : y < \frac{1}{2} - x\}$ cut off, and $\lambda_1 = 0$ and $\lambda_2 = 0$ in (1).

The exact solution was $u(x, y) = \sin x \sin y$ with the corresponding values for the boundary condition and the source function f. We remark that the exact solution u is not actually

in $H_0^1(\Omega)$, but this is not a problem since after the discretization the boundary conditions are eliminated in the right-hand side $\mathbf{b}_1$ of the linear algebraic problem (2). The stopping criterion for the PCG iterations to solve the capacitance matrix equation (step (ii) in Algorithm CMM) here, and elsewhere in the experiments, was when the energy norm of the preconditioned residuals decreased by 10^{-6} from its initial value (with the zero initial guess for the solution $\mathbf{w}_0$). We denote the number of interior meshpoints in each coordinate direction by n and the number of meshpoints on the separator Γ, equal to the dimension of the capacitance matrix, by n_b. By *iter* we denote the number of PCG iterations. Note that the dominant cost of the whole algorithm is the cost of the separable solver BLKTRI called (*iter* + 2) times while the cost of the preconditioners is negligible.

Recall that the preconditioners, described in detail in §3, are:

(1) square root of the finite difference analog of $-(d^2/ds^2)$ along Γ,

(2) hierarchical basis,

(3) vector probing (*A*), ([18]),

(4) vector probing (*B*), ([35] in the form (9)).

The first example corresponded to the model bilinear form (1) with the coefficients chosen as

(Problem 1) $$k_1(x) = e^x, \qquad k_2(y) = e^y.$$

TABLE 1
Problem 1 *and Preconditioner* 1.

n	$\|\text{error}\|_{\max}$	$\|\text{error}\|_2$	n_b	*iter*	CPU (in sec)
31	0.741D−05	0.413D−05	15	3	0.590
63	0.185D−05	0.102D−05	31	4	3.610
127	0.462D−06	0.253D−06	63	4	16.650
255	0.116D−06	0.628D−07	127	5	95.580

As the results in Table 1 indicate, the rate of convergence of the PCG iterations was essentially independent on the parameter of discretization $h = \frac{1}{n+1}$. The solution was clearly second order accurate, $O(h^2)$, here as well as in the remaining experiments; thus we do not report further on this item.

For comparison in Table 2 we include the results obtained without any preconditioner for the same problem and for the case with $k_1(x) = k_2(y) = 1$, i.e., for Δ (the Laplacian).

TABLE 2
Number of iterations for Problem 1 *and the Laplace operator* Δ *without preconditioning.*

n_b	Problem 1	Δ	$\Delta, u = random$
15	11	7	10
31	15	8	12
63	20	10	17
127	26	11	21

It appears that the convergence rate depends only weakly on n_b, particularly for the Laplacian and smooth solutions. It is known, e.g., Dryja [14], that in general $\text{Cond}(S) = O(\frac{1}{h})$, hence $\text{Cond}(C) = \text{Cond}(S^{-1}) = O(\frac{1}{h})$. This is very clearly seen for the example shown in the second and fourth columns of Table 2 (the number of iterations increased by about a factor of $\sqrt{2}$ as the number of meshpoints was doubled).

Next we tested anisotropic diffusion functions,

(Problem 2) $$k_1(x) = ce^x, \qquad k_2(y) = e^y,$$

where $c = 1, 10, 100$, and 1000 (because of symmetry results with $c := \frac{1}{c}$ were identical, here as well as in Tables 4, 5, and 6).

TABLE 3
Problem 2 and Preconditioner 1; number of iterations in case of anisotropy.

n_b	$c = 1$	$c = 10$	$c = 100$	$c = 1000$
15	3	6	10	14
31	4	7	11	18
63	4	7	12	20
127	5	7	12	22

Although the rate of convergence of the PCG iterations (see Table 3) slows down significantly with increased anisotropy, its behavior still suggests asymptotic independence for sufficiently high discretization parameter. (Note that $iter \leq n_b$.)

Next we proceed with the preconditioner based on a change to the hierarchical basis. The results (number of iterations) when we varied the anisotropy constant c are shown in Table 4.

TABLE 4
Problem 2 and Preconditioner 2; number of iterations in case of anisotropy.

n_b	$c = 1$	$c = 10$	$c = 100$	$c = 1000$
15	9	12	15	16
31	12	16	21	27
63	15	20	26	39
127	18	23	32	47

The results for problems with anisotropy with Preconditioner 3 based on the vector-probing technique defined by (8) are shown in Table 5. Similarly, the performance of Preconditioner 4 based on Wittum's smoothing-correction scheme defined by (9) is shown in Table 6. The specific frequencies ν_1, ν_2 which are used are also shown. These last two preconditioners showed very robust performance for problems with anisotropy. The preconditioner based on the two frequencies is not well understood although its performance is very satisfactory for properly chosen frequencies (so far, these were chosen experimentally). For more details cf. Wittum [35].

TABLE 5
Problem 2 and Preconditioner 3; number of iterations in case of anisotropy.

n_b	$c = 1$	$c = 10$	$c = 100$	$c = 1000$
15	14	13	11	11
31	18	17	14	13
63	20	20	18	16
127	25	24	22	21

TABLE 6
Problem 2 and Preconditioner 4; number of iterations in case of anisotropy.

n_b	$\nu_1,\ \nu_2$	$c = 1$	$c = 10$	$c = 100$	$c = 1000$
15	2, 4	9	7	6	4
31	3, 7	8	8	7	5
63	5, 15	8	9	9	5
127	10, 30	9	11	10	7

The final set of experiments (see Tables 7–10) was carried out for the discontinuous diffusion function in one direction:

(Problem 3)
$$k_1(x) = \begin{cases} ce^x & \text{for } x \geq 0.25, \\ e^x & \text{for } x < 0.25, \end{cases} \qquad k_2(y) = e^y.$$

Note that the line of discontinuity cuts the separator Γ. To ensure the continuity of the flux at the line of discontinuity of the diffusion function we chose the exact solution to be $u(x, y) = \sin(2\pi x)\sin(2\pi y)$. Note also that in the subdomain $\{(x, y)|x > 0.25\}$, Problem 3 is anisotropic like Problem 2.

TABLE 7
Problem 3 and Preconditioner 1; number of iterations in case of discontinuous coefficients.

n_b	$c = 1$	$c = 10$	$c = 100$	$c = 1000$	$c = 0.1$	$c = 0.01$	$c = 0.001$
15	4	10	12	15	10	12	12
31	4	10	14	20	11	18	20
63	4	10	15	21	12	20	33
127	5	10	15	23	12	21	36

Even in the case of discontinuous coefficients the rate of convergence of the PCG iterations is still essentially independent of the parameter of discretization, except for $c = 0.001$. At the same time, the convergence rate gradually deteriorates with the increased discontinuity.

TABLE 8
Problem 3 and Preconditioner 2; number of iterations in case of discontinuous coefficients.

n_b	$c = 1$	$c = 10$	$c = 100$	$c = 1000$	$c = 0.1$	$c = 0.01$	$c = 0.001$
15	9	13	15	15	13	13	14
31	12	17	25	27	17	23	25
63	15	20	31	42	21	30	42
127	18	23	36	56	24	36	57

For c=1, i.e., for continuous coefficients, one can observe the linear increase of the number of iterations with $\log(n_b + 1)$, the number of refinement levels as predicted by the theory. We saw that the HB preconditioner deteriorated with the anisotropy. This is also the case for the problem with discontinuous diffusion coefficients. Thus a proper stabilization of the HB-based preconditioner is worth providing. One possibility would be to use a hybrid multilevel iteration as proposed in [32].

The "$\star$" in Table 10 stands for nonpositive definiteness of Preconditioner 4. This is one of the difficulties with Wittum's smoothing-correction scheme when used as a preconditioner; namely, that the positive definiteness of the matrices constructed by probing is not a priori guaranteed.

TABLE 9
Problem 3 and Preconditioner 3; number of iterations in case of discontinuous coefficients.

n_b	$c = 1$	$c = 10$	$c = 100$	$c = 1000$	$c = 0.1$	$c = 0.01$	$c = 0.001$
15	14	13	15	15	14	14	15
31	19	19	23	25	22	21	21
63	23	28	29	42	32	30	31
127	28	42	32	57	49	46	46

TABLE 10
Problem 3 and Preconditioner 4; number of iterations in case of discontinuous coefficients.

n_b	$\nu_1,\ \nu_2$	$c = 1$	$c = 10$	$c = 100$	$c = 1000$	$c = .1$	$c = .01$	$c = .001$
15	2, 4	9	8	8	8	9	9	9
31	3, 7	8	9	9	9	10	11	12
63	5, 15	8	10	10	10	11	12	13
127	10, 30	9	11	11	11	12	15	$\star$

5. Conclusions. Preconditioner 1 is the best for operators with smooth coefficients ($c = 1$ in all tables), but its performance deteriorates gradually with increased anisotropy.

For Preconditioner 2 a similar but even faster deterioration (aggravated by its dependence on the size of the problem, n_b) is observed. This preconditioner needs stabilization of the growth of the condition number. One possible approach that needs further investigation is to use more complicated hybrid multilevel iterations as proposed in Vassilevski [32]. This would require additional solutions with a sequence of matrices $\{B^{(k)}\}$ obtained by discretizations of the separable elliptic problem in the rectangle R on a sequence of uniformly refined meshes.

In contrast, Preconditioners 3 and 4 based on vector probing exhibit a slightly improved performance with increased anisotropy. At the same time Preconditioner 3 shows deterioration in condition number with increasing n_b (for fixed coefficients in the operator) and thus cannot be recommended in this context. On the other hand, Preconditioner 4 shows great promise in such situations. Its properties must be better understood to allow for automatic optimization of the smoothing-correcting frequencies, and thus making its performance fully robust.

Acknowledgment. The authors thank the referees for the suggestions and remarks that led to an improved presentation of the paper.

REFERENCES

[1] G. P. ASTRAKHANTSEV, *Fictitious domain method for the second order elliptic equation with natural boundary conditions*, Zh. Vychisl. Mat. i Mat. Fiz., 18 (1978), pp. 119–125. (In Russian.)

[2] ———, *On numerical solution of mixed boundary value problems for second order elliptic equations in an arbitrary domain*, Zh. Vychisl. Mat. i Mat. Fiz., 25 (1985), pp. 200–209. (In Russian.)

[3] O. AXELSSON AND B. POLMAN, *A robust preconditioner based on algebraic substructuring and two-level grids*, Robust Multigrid Methods, W. Hackbusch, ed., Notes in Numer. Fluid Mech., 23, Vieweg, Braunschweig, 1989.

[4] A. BANEGAS, *Fast Poisson solvers for problems with sparsity*, Math. Comp., 32 (1978), pp. 441–446.

[5] R. E. BANK AND D. J. ROSE, *Marching algorithm for elliptic boundary value problems*, I: *The constant coefficient case*, SIAM J. Numer. Anal., 14 (1977), pp. 792–829.

[6] ———, *Marching algorithm for elliptic boundary value problems*, II: *The variable coefficient case*, SIAM J. Numer. Anal., 14 (1977), pp. 950–970.

[7] C. BÖRGERS AND O. B. WIDLUND, *On finite element domain imbedding methods*, SIAM J. Numer. Anal., 27 (1990), pp. 963–978.

[8] 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. Method Appl. Mech. Engrg., 67 (1988), pp. 140–159.

[9] B. L. BUZBEE, G. H. GOLUB, AND C. W. NIELSON, *On direct methods for solving Poisson equation*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[10] B. BUZBEE, F. DORR, J. GEORGE, AND G. GOLUB, *The direct solution of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal., 11 (1971), pp. 722–736.

[11] T. F. CHAN AND T. P. MATHEW, *The interface probing technique in domain decomposition*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 212–238.

[12] T. F. CHAN, T. P. MATHEW, AND J. SHAO, *Efficient variants of the vertex space domain decomposition algorithms*, Tech. Report CAM 92-07, University of California, Los Angeles, CA, 1992.

[13] S. DEMKO, W. F. MOSS, AND P. W. SMITH, *Decay rates of inverses of band matrices*, Math. Comp., 43 (1984), pp. 491–499.

[14] M. DRYJA, *A finite element-capacitance matrix method for the elliptic problem*, SIAM J. Numer. Anal., 20 (1983), pp. 671–680.

[15] ———, *Capacitance matrix method for Dirichlet problem on polygon region*, Numer. Math., 39 (1982), pp. 51–64.

[16] ———, *A finite element capacitance matrix method for elliptic problems on regions partitioned into substructures*, Numer. Math., 44 (1984), pp. 153–168.

[17] R. E. EWING, R. D. LAZAROV, AND P. S. VASSILEVSKI, *Local refinement techniques for elliptic problems on cell-centered grids*, II, *Optimal order two-grid iterative methods*, J. Numer. Linear Alg. Appl., to appear.

[18] D. E. KEYES AND W. D. GROPP, *A comparison of domain decomposition techniques for elliptic differential equations and their parallel implementation*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 166–202.

[19] Y. A. KUZNETSOV, *Block relaxation methods in subspaces, their optimization and application*, Soviet J. Numer. Anal. and Math. Modelling, 4 (1989), pp. 433–452.

[20] S. MCCORMICK AND J. THOMAS, *The fast adaptive composite grid* (FAC) *method for elliptic equations*, Math. Comp., 46 (1986), pp. 439–456.

[21] S. MCCORMICK, *Multilevel Adaptive Methods for Partial Differential Equations*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[22] S. V. NEPOMNYASCHIKH, *Method of splitting into subspaces for solving elliptic boundary value problems in complex form domain*, Soviet J. Numer. Anal. and Math. Modelling, 6 (1991), pp. 1–26.

[23] D. P. O'LEARY AND O. WIDLUND, *Capacitance matrix methods for the Helmholtz equation on general three-dimensional regions*, Math. Comp., 33 (1979), pp. 849–879.

[24] W. PROSKUROWSKI, *Numerical solution of Helmholtz equation by implicit capacitance matrix method*, ACM Trans. Math. Software, 5 (1979), pp. 36–49.

[25] W. PROSKUROWSKI AND P. VASSILEVSKI, *Preconditioning nonsymmetric and indefinite capacitance matrix problems in domain imbedding*, Tech. Report CAM 92-48, Univ. of California, Los Angeles, CA, 1992.

[26] W. PROSKUROWSKI AND O. WIDLUND, *On the numerical solution of Helmholtz equation by the capacitance matrix method*, Math. Comp., 30, (1976), pp. 433–468.

[27] ———, *A finite element-capacitance matrix method for the Neumann problem for Laplace equation*, SIAM Sci. Statist. Comput., 1 (1980), pp. 410–425.

[28] B. SMITH AND O. B. WIDLUND, *A domain decomposition algorithm using a hierarchical basis*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 1212–1226.

[29] P. N. SWARZTRAUBER, *The method of cyclic reduction, Fourier analysis and the* FACR *algorithm for the discrete solution of Poisson equation in a rectangle*, SIAM Rev., 19 (1977), pp. 490–501.

[30] P. N. SWARZTRAUBER AND R. A. SWEET, *Efficient FORTRAN subprograms for the solution of elliptic partial differential equations*, ACM Trans. Math. Software, 5 (1979), pp. 352–364.

[31] P. S. VASSILEVSKI, *On some ways of approximating inverses of banded matrices in connection with deriving preconditioners based on incomplete block-factorizations*, Computing, 43 (1990), pp. 277–296.

[32] ———, *Hybrid V-cycle algebraic multilevel preconditioners*, Math. Comp., 58 (1992), pp. 489–512.

[33] ———, *Multilevel preconditioners for elliptic problems by substructuring*, Appl. Math. Comput., 46 (1991), pp. 79–106.

[34] H. YSERENTANT, *On the multilevel splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

[35] G. WITTUM, *An* ILU*-based smoothing correction scheme*, Parallel Algorithms for PDEs, W. Hackbusch, ed., Notes in Numer. Fluid Mech., 24 Vieweg, Braunschweig, 1990.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 89–105, January 1994

006

A FAMILY OF QUASI-MINIMAL RESIDUAL METHODS FOR NONSYMMETRIC LINEAR SYSTEMS*

CHARLES H. TONG†

Abstract. The quasi-minimal residual (QMR) algorithm by Freund, Gutknecht, and Nachtigal [*Research Institute for Advanced Computer Science Tech. Report* 91.09, NASA Ames Research Center, Moffett Field, CA; *Numer. Math.*, 60 (1991), pp. 315–339], in addition to its ability to avoid breakdowns, also improves the irregular convergence behavior encountered by the biconjugate gradient (BiCG) method through the incorporation of quasi minimization. This paper studies this quasi-minimal residual approach applied, for simplicity, to the nonlook-ahead version of the QMR method. In particular, a family of quasi-minimal residual methods is defined where on both ends lie the original QMR method and a minimal residual method while lying in between are the variants derived here. These variants are shown to consume a little more computational work per iterations than the original QMR method, but generally give an even better convergence behavior as well as lower iteration counts. The same idea was also applied to a recently proposed transpose-free quasi-minimal residual method. In particular, a few quasi-minimal residual variants of the Van der Vorst BiCGSTAB method are derived. Numerical results are presented comparing the quasi-minimal residual and transpose-free quasi-minimal residual variants derived in this paper.

Key words. Lanczos method, quasi-minimal residual approach, nonsymmetric linear systems, transpose-free methods

AMS subject classifications. 65F10, 65N20

1. Introduction.

Conjugate gradient-like methods for nonsymmetric problems generally fall into three classes: orthogonalization methods, e.g., GMRES [14]; biorthogonalization methods, e.g., biconjugate gradient [6], [11]; and methods based on normal equation approach, e.g., conjugate gradient normal residual (CGNR) [10]. Due to its short recurrences, and thus lower storage requirements, the class of biorthogonalization methods has been studied extensively in recent years. The earliest algorithm in this class is the biconjugate gradient (BiCG) algorithm, which was first proposed by Lanczos [11], and then rediscovered by Fletcher [6]. Despite its advantage, BiCG also suffers from a few well-known drawbacks. These drawbacks are the need for adjoint matrix vector multiplication ($A^T y$), the possibility of breakdowns or near breakdowns, and its irregular convergence behavior. Subsequently, transpose-free methods such as conjugate gradient squared (CGS) [16], BiCGSTAB [18], etc., were developed to alleviate the need for adjoint matrix vector multiplies. To address the breakdown or near-breakdown problem, Freund, Gutknecht, and Nachtigal [8], [9] proposed the quasi-minimal residual (QMR) algorithm, which is based on a look-ahead Lanczos procedure to overcome curable breakdowns. The quasi-minimization step, in addition to curing one of the breakdown problems, also gives a smoother convergence curve. Such a quasi-minimization step can be applied to the transpose methods such as CGS and BiCGSTAB, giving rise to the recently developed algorithms such as Freund's transpose-free QMR algorithm [7], the transpose-free QMR method of Chan, De Pillis, and Van der Vorst [3], and QMRCGSTABx algorithms of Chan et al. [4].

While Faber and Manteuffel [5] proved that generally there exists no conjugate gradient-type scheme that incorporates both short recurrences and a minimization property, the quasi-minimal residual approach represents a step toward a compromise between these two conflicting but desirable features. This paper shows that a family of quasi-minimal residual methods (for simplicity, the nonlook-ahead version is used here) can be defined that is based on the use of different weight matrices (to be defined later) in the quasi-minimization step. We show that

*Received by the editors April 28, 1992; accepted for publication (in revised form) February 15, 1993.

†Center for Computational Engineering, Sandia National Laboratory, Livermore, California 94551-0969 (chtong@california.sandia.gov).

when this weight matrix is very sparse (i.e., diagonal), we have a simplified (nonlook-ahead) version of the Freund, Gutknecht, and Nachtigal quasi-minimal residual method. On the other hand, if we allow the weight matrix to be very dense (i.e., upper triangular), we have a minimal residual method. However, this minimal residual method suffers again from long recurrences. The variants in this paper are derived by allowing the weight matrices to be other than the two extremes mentioned above. In particular, we consider weight matrices that are block diagonal having upper triangular blocks. We show that these QMR (and BiCGSTAB-based transpose-free QMR) variants generally give smoother convergence curves and lower iteration counts with increasing block sizes at the expense of little extra computation per iteration.

The outline of this report is as follows. In §2, the basic ideas of the QMR approach are described and a few QMR variants are then derived. In §3, implementation details of the QMR variants as well as their computational complexity are given. In §4, numerical results are presented and discussed. In §5, we consider the transpose-free QMR algorithms and then derive a few quasi-minimal residual variants of the Van der Vorst BiCGSTAB method [18]. In §6, implementation details of the transpose-free QMR variants are given. In §7, numerical results are presented and discussed. Finally, in §8, some concluding remarks are made.

2. A family of quasi-minimal residual algorithms.

2.1. The quasi-minimal residual approach. We give a nonsingular and nonsymmetric $N \times N$ matrix A and two initial nonzero vectors $v_1 \in \Re^N$ and $w_1 \in \Re^N$ such that $w_1^T v_1 \neq 0$. The classical nonsymmetric Lanczos procedure, in exact arithmetic and in the absence of breakdowns, generates two sequences of vectors

$$V_n = \{v_1\ v_2\ \cdots\ v_n\} \quad \text{and} \quad W_n = \{w_1\ w_2\ \cdots\ w_n\}, \tag{2.1}$$

which satisfy the following two properties. (i) The two sequences span two Krylov subspaces. That is,

$$\text{span}\{v_1, v_2, \ldots, v_n\} = \text{span}\{v_1, Av_1, \ldots, A^{n-1}v_1\} = K_n(v_1, A), \tag{2.2}$$

$$\text{span}\{w_1, w_2, \ldots, w_n\} = \text{span}\{w_1, A^T w_1, \ldots, (A^T)^{n-1}w_1\} = K_n(w_1, A^T). \tag{2.3}$$

(ii) The two sequences satisfy a biorthogonality condition

$$w_i^T v_j = \delta_{ij}, \tag{2.4}$$

where δ_{ij} is the Kronecker symbol (that is, $\delta_{ij} = 1$, when $i = j$, and 0 otherwise).

The nonsymmetric Lanczos procedure can be written in matrix form as

$$AV_n = V_{n+1}H_n^e, \tag{2.5}$$

where H_n^e is a $(n+1) \times n$ tridiagonal matrix defined by parameters generated in the Lanczos process.

This nonsymmetric procedure can be used to find the solution of a linear system $Ax = b$. In particular, let $x_0 \in \Re^N$ be any initial guess. The initial residual vector is then $r_0 = b - Ax_0$. If we let $v_1 = r_0 / \| r_0 \|$ and w_1 be arbitrary with unit 2-norm (for example, $w_1 = v_1$), then the iterate x_n (solution at iteration n) is

$$x_n = x_0 + V_n z,\ z \in \Re^N. \tag{2.6}$$

Moreover, after some manipulations, the residual vector can be written as

$$r_n = V_{n+1}\Omega_{n+1}^{-1}(d_n - \Omega_{n+1}H_n^e z), \tag{2.7}$$

where Ω_{n+1} is an $(n+1) \times (n+1)$ diagonal weight matrix (following the Freund and Nachtigal notation [9]), $e_1^{(n+1)} = [1\ 0\ \cdots\ 0]^T \in \Re^{n+1}$, $\rho_0 = \| r_0 \|$, and $d_n = \omega_1 \rho_0 e_1^{(n+1)}$.

Instead of minimizing the 2-norm of the residual $\| r_n \|$, which takes prohibitively large number of arithmetic operations, the QMR approach minimizes only the Euclidean norm of the expression $(d_n - \Omega_{n+1} H_n^e z)$. The solution of this least squares problem can be computed by a standard QR decomposition of $\Omega_{n+1} H_n^e$ giving

$$\Omega_{n+1} H_n^e = Q_n^T \begin{bmatrix} R_n \\ 0 \end{bmatrix}, \tag{2.8}$$

where Q_n is an orthogonal $(n+1) \times (n+1)$ matrix, and R_n is a nonsingular upper triangular $n \times n$ matrix. This gives

$$\min_{z \in \Re^n} \| d_n - \Omega_{n+1} H_n^e z \|_2 = \min_{z \in \Re^n} \left\| \left(Q_n d_n - \begin{bmatrix} R_n \\ 0 \end{bmatrix} z \right) \right\|_2 . \tag{2.9}$$

If we let t_n be an $n \times 1$ vector consisting of the first n elements of the $(n+1) \times 1$ vector $Q_n d_n$, then z_n and the solution x_n are given by

$$z_n = R_n^{-1} t_n \quad \text{and} \quad x_n = x_0 + V_n z_n, \tag{2.10}$$

respectively.

2.2. Generalization of the QMR approach. In the simplified (nonlook-ahead) QMR algorithm, the weight matrix Ω_{n+1} was chosen to be a diagonal matrix. In general, Ω_{n+1} does not have to be diagonal. In fact, by allowing Ω_{n+1}^{-1} to be upper triangular, it is possible to construct an Ω_{n+1} such that $V_{n+1}\Omega_{n+1}^{-1}$ is orthonormal. This gives a minimal residual algorithm since

$$\begin{aligned} \min_{z \in \Re^n} \| r_n \|_2 &= \min_{z \in \Re^n} \| V_{n+1} \Omega_{n+1}^{-1} (\rho_0 \Omega_{n+1} e_1^{(n+1)} - \Omega_{n+1} H_n^e z) \|_2 \\ &= \min_{z \in \Re^n} \| \rho_0 \Omega_{n+1} e_1^{(n+1)} - \Omega_{n+1} H_n^e z \|_2 . \end{aligned} \tag{2.11}$$

However, since now the matrix $\Omega_{n+1} H_n^e$ is dense upper Hessenberg instead of tridiagonal, the computation of z becomes very expensive and takes up a prohibitively large amount of storage. This algorithm is in fact similar to GMRES except that it is susceptible to breakdowns.

Another alternative is to find a sparse Ω_{n+1} (other than diagonal) so that $V_{n+1}\Omega_{n+1}^{-1}$ becomes closer to orthonormal at the expense of requiring to store and use a few more of the previous direction vectors. A simple variant is to make Ω_{n+1}^{-1} block diagonal where each block is a 2×2 upper triangular submatrix. This can be accomplished by orthonormalizing consecutive pairs of v_i's, i.e., orthonormalize v_2 against v_1, v_4 against v_3, and so on. We call this new weight matrix $(B_{n+1}^{(2)})^{-1}$ and then we solve a different least squares problem

$$\| d_n - (B_{n+1}^{(2)})^{-1} H_n^e z_n \|_2 = \min_{z \in \Re^n} \| d_n - (B_{n+1}^{(2)})^{-1} H_n^e z \|_2, \tag{2.12}$$

where d_n is the same as before ($(B_{n+1}^{(2)})^{-1} d_n = d_n$ since all except the first element of d_n is nonzero and $B_{n+1}^{(2)}(1,1) = 1$). As B_{n+1} is block diagonal with a block size of 2, $(B_{n+1}^{(2)})^{-1} H_n^e$ will have one more superdiagonal than H_n^e. Thus only one additional direction vector needs to be kept for computing the solution update compared to the QMR method. Also, as will be shown in later sections, this algorithm takes only one more inner product and one more daxpy

($y = ax + b$, where x, y, b are vectors and a is a scalar) operation per iteration compared to the QMR method. We call this QMR variant BQMR(2) in subsequent sections.

We can further generalize this approach by allowing the weight matrix to be a block diagonal matrix of block size 3, 4, and so on. In this paper, we implement up to a block size of 3 (which we call BQMR(3) and the weight matrix for this case is $(B_{n+1}^{(3)})^{-1}$). It becomes obvious that when the block size is $n+1$, $V_{n+1}B_{n+1}^{(n+1)}$ has orthonormal columns, and we have a minimal residual algorithm similar to GMRES. When the block size is 1, we have the simplified (based on the classical Lanczos procedure) QMR algorithm (to be consistent in the use of notations, the simplified QMR algorithm will also be called BQMR(1) in subsequent sections).

2.3. A residual bound for the BQMR(k) algorithms. Freund and Nachtigal [9] have given a residual bound for the QMR algorithm based on the the following theorem [9].

THEOREM 2.1. *Suppose that the $m \times m$ matrix H_m generated by m steps of the* QMR *algorithm is diagonalizable (m denotes the termination index of the look-ahead Lanczos, for details refer to* [9]*), and set*

$$H = \Omega_{m-1} H_m \Omega_{m-1}^{-1}. \tag{2.13}$$

Then, for $n = 1, 2, \ldots, m-1$, the residual vectors of the QMR *algorithm satisfy*

$$\| r_n \| \leq \| r_0 \| \kappa(H)\sqrt{n+1}\, \vartheta_n \max_{j=1,\ldots,n+1} (\omega_1/\omega_j), \tag{2.14}$$

where

$$\kappa(H) = \min_{X: X^{-1}HX \text{diagonal}} \| X \| \cdot \| X^{-1} \|, \tag{2.15}$$

and

$$\vartheta_n = \min_{P \in \Pi_n : P(0)=1} \max_{\lambda \in \lambda(A)} |P(\lambda)|. \tag{2.16}$$

This convergence theorem is contingent on the use of a look-ahead Lanczos procedure. The formulation of BQMR(k) in the last section, however, is based on the classical Lanczos procedure. Thus this theorem does not hold for BQMR(k), in general, because of the possibility of breakdowns. Nevertheless, suppose that for a given test problem, BQMR(k) does not encounter breakdowns of the type $w_n^T v_n = 0$, $v_n \neq 0$, $w_n \neq 0$. Then we can derive a similar convergence bound for the BQMR(k) algorithms. Since this bound is rather restricted or not rigorous, we will state it as a remark instead of a theorem.

Remark. Suppose an $N \times N$ linear system is given and in exact arithmetic, the BQMR(k) for some $k \leq N$ terminates at iteration $m \leq N$ with $\| v_{m+1} \| = 0$ or $\| w_{m+1} \| = 0$ (note that this index m is different from that defined in [9]), and no breakdown of the type $v_n^T w_n = 0$, $v_n \neq 0$, $w_n \neq 0$ occurs at iterations 1 through m. Furthermore, suppose the $m \times m$ matrix H_m generated by m steps of the BQMR(k) algorithm is diagonalizable, and set

$$H = (B_{m-1}^{(k)})^{-1} H_m B_{m-1}^{(k)}. \tag{2.17}$$

Then, for $n = 1, 2, \ldots, m-1$, the residual vectors of the BQMR(k) algorithm satisfy

$$\| r_n \| \leq \| r_0 \| \kappa(H)\sqrt{\left\lceil \frac{n+1}{k} \right\rceil}\, \vartheta_n, \tag{2.18}$$

where

$$\vartheta_n = \min_{P\in\Pi_n: P(0)=1} \max_{\lambda\in\lambda(A)} |P(\lambda)|. \tag{2.19}$$

Proof. The proof of this remark is similar to that of the Freund and Nachtigal theorem stated above. In this theorem, the factor $\sqrt{n+1}$ arises from bounding $\| V_{n+1} \|$. A tighter bound (by a factor of $\approx \sqrt{k}$) can be prescribed by realizing that $V_{n+1}B_{n+1}^{(k)}$ is k-orthonormal (columns 1 through k are orthonormal, columns k+1 through 2k are orthonormal, etc.). Moreover, since H_n^e is tridiagonal and $B_{n+1}^{(k)}$ is block diagonal with upper triangular blocks and nonzero diagonal elements, $(B_{n+1}^{(k)})^{-1}H_n^eB_{n+1}^{(k)}$ is upper Hessenberg with nonzero subdiagonal. Using this fact, together with the proof in [9], we obtain the same ϑ_n as in [9], and thus we arrive at the above remark. □

The above remark is given to shed some light on the convergence behavior of these QMR variants. This remark basically implies that the larger the block size (k) used, the tighter are the residual bounds if no breakdown occurs. In the limit when $k = n+1$ (or BQMR(∞)) and in absence of breakdowns, the residual bound is the same as the GMRES residual bound.

3. Implementation and complexity of the BQMR(k) algorithms.

3.1. Implementation details.

3.1.1. The BQMR(2) algorithm. Considering the BQMR(2) algorithm, solving the least squares problem

$$\| d_n - (B_{n+1}^{(2)})^{-1}H_n^e z_n \|_2 = \min_{z\in\Re^n} \| d_n - (B_{n+1}^{(2)})^{-1}H_n^e z \|_2 \tag{3.1}$$

basically involves three steps.

Step 1. Update $B_{n+1}^{(2)}$.

Step 2. Update the relatively sparse matrix $\tilde{H}_n^e = (B_{n+1}^{(2)})^{-1}H_n^e$.

Step 3. Perform QR decomposition on $\tilde{H}_n^e$ to get R_n, and apply the same Q_n to d_n to get $t_n = [\tau_1 \cdots \tau_n]^T$.

The solution can then be updated by

$$x_n = x_{n-1} + V_n R_n^{-1}[0 \cdots 0\ \tau_n]^T. \tag{3.2}$$

Since the update processes for the first and second step are different for odd and even iterations, we shall describe them separately.

At the odd iterations, $B_{n+1}^{(2)}$ and H_n^e are of the form

$$B_{n+1}^{(2)} = \begin{bmatrix} & & 0 \\ & B_n^{(2)} & \vdots \\ & & \delta_{n+1} \\ 0 & \cdots & \epsilon_{n+1} \end{bmatrix}, \qquad H_n^e = \begin{bmatrix} & & 0 \\ & H_{n-1}^e & \vdots \\ & & \beta_n \\ & & \alpha_n \\ 0 & \cdots\ 0 & \gamma_{n+1} \end{bmatrix}. \tag{3.3}$$

The new entries in $B_{n+1}^{(2)}$, namely, δ_{n+1} and ϵ_{n+1} can be computed by orthonormalizing v_{n+1} against v_n.

Let the nth column of H_n^e be denoted by $h_n = [0 \ \ldots\ \beta_n\ \alpha_n\ \gamma_{n+1}]^T$ (which has a vector of length $n+1$). Then, at the nth iteration, $\tilde{H}_n^e$ can be updated by performing a U-solve (solving a linear system where the matrix is upper triangular), that is, to find a vector u_n such that

$$B_{n+1}^{(2)}u_n = h_n, \tag{3.4}$$

where u_n will have the form $[0 \cdots 0\ \mu_4\ \mu_3\ \mu_2\ \mu_1]^T$.

At the even iterations, the derivation can be carried out similarly. Here no orthonormalization needs to be performed. Also $B_{n+1}^{(2)}$ is in a different form:

$$B_{n+1}^{(2)} = \begin{bmatrix} & & 0 \\ & B_n^{(2)} & \vdots \\ & & 0 \\ 0 & \cdots & 1 \end{bmatrix}. \tag{3.5}$$

Again, a U-solve is performed on $B_{n+1}^{(2)} u_n = h_n$, where now the fourth-to-the-last entry (μ_4 in the odd case) in u_n is 0.

Step 3 first applies the previous Givens rotations to u_n's so that

$$Q_{n-1} u_n = [0 \cdots 0\ v_4\ v_3\ v_2\ t_1\ \mu_1]^T. \tag{3.6}$$

Next a new Givens rotation is used to zero out the entry corresponding to μ_1 giving

$$Q_n u_n = [0 \cdots 0\ v_4\ v_3\ v_2\ v_1\ 0]^T, \tag{3.7}$$

which is the last column of R_n. The computation of $t_n = Q_n d_n$ can be performed the same way as in BQMR(1).

Finally, the expression $V_n R_n^{-1}[0 \cdots 0\ \tau_n]^T$ needs to be evaluated. This can be achieved by defining a sequence of vectors y_i where

$$Y_n = [y_1\ y_2\ \cdots\ y_n] = V_n R_n^{-1}, \tag{3.8}$$

and the solution can be updated by

$$x_n = x_{n-1} + \tau_n * y_n. \tag{3.9}$$

It is straightforward to show that

$$y_n = (v_n - v_2 y_{n-1} - v_3 y_{n-2} - v_4 y_{n-3})/v_1. \tag{3.10}$$

In [9], it was shown that the BQMR(1) residual can be updated by

$$r_n^{\mathrm{BQMR(1)}} = s_{n+1}^2 r_{n-1}^{\mathrm{BQMR(1)}} - c_{n+1} \tilde{\tau}_{n+1} v_{n+1}. \tag{3.11}$$

The BQMR(2) residual can be updated slightly differently by

$$r_n^{\mathrm{BQMR(2)}} = s_{n+1}^2 r_{n-1}^{\mathrm{BQMR(2)}} - c_{n+1} \tilde{\tau}_{n+1} \hat{v}_{n+1}, \tag{3.12}$$

where $\hat{v}_{n+1}$ is the $(n+1)$th column of $V_{n+1} B_{n+1}^{(2)}$.

The pseudocode for the BQMR(2) algorithm is presented in the following.

ALGORITHM BQMR(2)

(1) Initialization

(a) $r_0 = b - Ax_0$; $v_1 = r_0 / \| r_0 \|$; w_1 arbitrary

(b) $y_{-2} = y_{-1} = y_0 = v_{-1} = v_0 = w_{-1} = w_0 = 0$; $r_0^{\mathrm{BQMR(2)}} = r_0$

(c) $\tilde{\tau}_1 = \| r_0 \|$; $s_{-2} = s_{-1} = s_0 = 0$; $c_{-2} = c_{-1} = c_0 = -1$

(d) $\epsilon_0 = 1$; $\delta_0 = \beta_1 = \gamma_1 = 0$

(2) **For** $n = 1, 2, \ldots$ do

(a) $\alpha_n = w_n^T A v_n$
(b) $\tilde{v}_{n+1} = A v_n - \alpha_n v_n - \beta_n v_{n-1}$
(c) $\tilde{w}_{n+1} = A^T w_n - \alpha_n w_n - \gamma_n w_{n-1}$
(d) $\gamma_{n+1} = \| \tilde{v}_{n+1} \|$; $\beta_{n+1} = \tilde{w}_{n+1}^T \tilde{v}_{n+1} / \gamma_{n+1}$
(e) $v_{n+1} = \tilde{v}_{n+1} / \gamma_{n+1}$; $w_{n+1} = \tilde{w}_{n+1} / \beta_{n+1}$;

(3) U-solve to get the modified H matrix
(a) if n is odd
$\bar{v}_{n+1} = v_{n+1} - (v_{n+1}^T v_n) v_n$; $\hat{v}_{n+1} = \bar{v}_{n+1} / \| \bar{v}_{n+1} \|$
$\epsilon_{n+1} = 1 / \| \bar{v}_{n+1} \|$; $\delta_{n+1} = -(v_{n+1}^T v_n) * \epsilon_{n+1}$
$\mu_1 = \gamma_{n+1} / \epsilon_{n+1}$; $\mu_2 = \alpha_n - \delta_{n+1} * \mu_1$; $\mu_3 = \beta_n / \epsilon_{n-1}$; $\mu_4 = -\delta_{n-1} * \mu_3$
(b) else if n is even
$\mu_1 = \gamma_{n+1}$; $\mu_2 = \alpha_n / \epsilon_n$; $\mu_3 = \beta_n - \delta_n * \mu_2$; $\mu_4 = 0$
$\hat{v}_{n+1} = v_{n+1}$

(4) Perform quasi minimization
(a) $\nu_4 = s_{n-2} * \mu_4$; $t_3 = -c_{n-2} * \mu_4$
(b) $\nu_3 = -c_{n-1} * t_3 + s_{n-1} * \mu_3$; $t_2 = -s_{n-1} * t_3 - c_{n-1} * \mu_3$
(c) $\nu_2 = -c_n * t_2 + s_n * \mu_2$; $t_1 = -s_n * t_2 - c_n * \mu_2$
(d) if $(|\mu_1| > |t_1|)$ $t_a = -t_1 / \mu_1$; $s_{n+1} = 1/\sqrt{1 + t_a^2}$; $c_{n+1} = c_{n+1} * t_a$
(e) else $t_a = -\mu_1 / t_1$; $c_{n+1} = 1/\sqrt{1 + t_a^2}$; $s_{n+1} = c_{n+1} * t_a$
(f) $\nu_1 = -c_{n+1} * t_1 + s_{n+1} * \mu_1$; $\tau = -c_{n+1} \tilde{\tau}$; $\tilde{\tau} = -s_{n+1} \tilde{\tau}$
(g) $y_n = (v_n - \nu_2 y_{n-1} - \nu_3 y_{n-2} - \nu_4 y_{n-3}) / \nu_1$

(5) Update solution and residual
(a) $x_n = x_{n-1} + \tau y_n$
(b) $r_n^{\mathrm{BQMR(2)}} = s_{n+1}^2 r_{n-1}^{\mathrm{BQMR(2)}} - c_{n+1} \tilde{\tau} \hat{v}_{n+1}$

(6) **End for**

3.1.2. The BQMR(3) algorithm. The BQMR(3) algorithm can be derived in a similar way. The pseudocode for the BQMR(3) algorithm is presented here.

ALGORITHM BQMR(3)
(1) Initialization
(a) $r_0 = b - A x_0$; $v_1 = r_0 / \| r_0 \|$; w_1 arbitrary
(b) $y_{-3} = y_{-2} = y_{-1} = y_0 = v_{-1} = v_0 = w_{-1} = w_0 = 0$; $r_0^{\mathrm{BQMR(3)}} = r_0$
(c) $\tilde{\tau}_1 = \| r_0 \|$; $s_{-3} = s_{-2} = s_{-1} = s_0 = 0$
(d) $c_{-3} = c_{-2} = c_{-1} = c_0 = -1$; $\epsilon_0 = \epsilon_{-1} = 1$; $\delta_0 = \delta_{-1} = \eta = 0$

(2) **For** $n = 1, 2, \ldots$ do
same as step (2a)–(2e) in Algorithm BQMR(2)

(3) U-solve to get the modified H matrix
(a) if mod(n,3) = 1
$\bar{v}_{n+1} = v_{n+1} - (v_{n+1}^T v_n) v_n$; $\hat{v}_{n+1} = \bar{v}_{n+1} / \| \bar{v}_{n+1} \|$
$\epsilon_{n+1} = 1 / \| \bar{v}_{n+1} \|$; $\delta_{n+1} = -(v_{n+1}^T v_n) * \epsilon_{n+1}$
$\mu_1 = \gamma_{n+1} / \epsilon_{n+1}$; $\mu_2 = \alpha_n - \delta_{n+1} * \mu_1$; $\mu_3 = \beta_n / \epsilon_{n-1}$
$\mu_4 = -\delta_{n-1} * \mu_3 / \epsilon_{n-2}$; $\mu 5 = -\delta_{n-2} * \mu_4 - \eta * \mu_3$
(b) else if mod(n,3) = 2
$\bar{v}_{n+1} = v_{n+1} - (v_{n+1}^T v_{n-1}) v_{n-1} - (v_{n+1}^T \hat{v}_n) \hat{v}_n$; $\hat{v}_{n+1} = \bar{v}_{n+1} / \| \bar{v}_{n+1} \|$
$\epsilon_{n+1} = 1 / \| \bar{v}_{n+1} \|$; $\delta_{n+1} = -(v_{n+1}^T \hat{v}_n) * \epsilon_{n+1} * \epsilon_n$
$\eta = -\epsilon_{n+1} * v_{n+1}^T v_{n-1} - \epsilon_{n+1} * (v_{n+1}^T \hat{v}_n) * \delta_n$
$\mu_1 = \gamma_{n+1} / \epsilon_{n+1}$; $\mu_2 = (\alpha_n - \delta_{n+1} * \mu_1) / \epsilon_n$
$\mu_3 = \beta_n - \delta_n * \mu_2 - \eta * \mu_1$; $\mu_4 = \mu_5 = 0$

(c) else if mod(n,3) = 0

$\mu_1 = \gamma_{n+1}$; $\mu_2 = \alpha_n/\epsilon_n$; $\mu_3 = (\beta_n - \delta_n * \mu_2)/\epsilon_{n-1}$

$\mu_4 = -\delta_{n-1} * \mu_3 - \eta * \mu_2$; $\mu_5 = 0$; $\hat{v}_{n+1} = v_{n+1}$

(4) Perform quasi minimization

(a) $\nu_5 = s_{n-3} * \mu_5$; $t_4 = -c_{n-3} * \mu_5$

(b) $\nu_4 = -c_{n-2} * t_4 + s_{n-2} * \mu_4$; $t_3 = -s_{n-2} * t_4 - c_{n-2} * \mu_4$

(c) $\nu_3 = -c_{n-1} * t_3 + s_{n-1} * \mu_3$; $t_2 = -s_{n-1} * t_3 - c_{n-1} * \mu_3$

(c) $\nu_2 = -c_n * t_2 + s_n * \mu_2$; $t_1 = -s_n * t_2 - c_n * \mu_2$

(d) if $(|\mu_1| > |t_1|)$ $t_a = -t_1/\mu_1$; $s_{n+1} = 1/\sqrt{1+t_a^2}$; $c_{n+1} = c_{n+1} * t_a$

(e) else $t_a = -\mu_1/t_1$; $c_{n+1} = 1/\sqrt{1+t_a^2}$; $s_{n+1} = c_{n+1} * t_a$

(f) $\nu_1 = -c_{n+1} * t_1 + s_{n+1} * \mu_1$; $\tau = -c_{n+1}\tilde{\tau}$; $\tilde{\tau} = -s_{n+1}\tilde{\tau}$

(g) $y_n = (v_n - \nu_2 y_{n-1} - \nu_3 y_{n-2} - \nu_4 y_{n-3} - \nu_5 y_{n-4})/\nu_1$

(5) Update solution and residual

(a) $x_n = x_{n-1} + \tau y_n$

(b) $r_n^{\mathrm{BQMR(3)}} = s_{n+1}^2 r_{n-1}^{\mathrm{BQMR(3)}} - c_{n+1}\tilde{\tau}\hat{v}_{n+1}$

(6) **End for**

3.2. Complexity and storage analysis. Table 3.1 shows the operation counts required for the BQMR(1), BQMR(2), and BQMR(3) algorithms. Since the operation counts for the BQMR(2) algorithm are different between odd and even iterations, only their average is listed. The same is true for BQMR(3). BQMR(2) and BQMR(3) are only slightly more expensive than BQMR(1). When large block size k is used, the operation count for BQMR(k) per iteration grows with the square of the block size.

TABLE 3.1
Complexity and storage statistics per iteration.

Method	(1)	(2)	(3)	(4)	Total op. count	Storage
BQMR(1)	2	4	7	6	28N+2 (MVP + P)	12N+A
BQMR(2)	2	5	8	6	32N+2 (MVP + P)	13N+A
BQMR(3)	2	6	9	6	36N+2 (MVP + P)	14N+A

(1) Number of matrix vector multiplications

(2) Number of inner product calculations

(3) Number of daxpy operations

(4) Number of other (e.g., vector add) operations

P = preconditioning

MVP = matrix vector products.

4. Numerical experiments with BQMR(k) algorithms. The numerical examples used in this experiment range from a two-dimensional convection diffusion equation to the problems in the Harwell–Boeing collections [2]. cde31 and cde63 are the convection-diffusion equations

$$-\Delta u + \gamma\left(x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y}\right) + \beta u = f, \tag{4.1}$$

where $\gamma = 50$ and 100, $\beta = -25$ and -100 for $n = 31$ (i.e., 31×31 grid) and $n = 63$, respectively. The rest of the problems were obtained from the Harwell–Boeing collection.

Table 4.1 shows the iteration counts for the different algorithms without preconditioning and with right ILUT(0) preconditioning. (This ILUT(0) was originally proposed by Saad [13]. The version used in this experiment, however, was borrowed from the implementation by Freund and Nachtigal.) The result for GMRES (nonrestarted version) is also included for comparison purposes since, when breakdown does not occur, GMRES is mathematically equivalent to BQMR(∞). The stopping criterion used here is $\| r_k \| / \| r_0 \| < 10^{-8}$. Moreover, $x_0 = 0$ and $\tilde{r}_0 = r_0$.

TABLE 4.1
Iteration counts for different problems.

Problem	BQMR(1)		BQMR(2)		BQMR(3)		GMRES
	(U)	(P)	(U)	(P)	(U)	(P)	(P)
cde31	101	26	101	26	91	26	24
cde63	259	45	259	43	259	43	39
orsreg1	323	29	323	27	318	26	24
orsirr1	1026	21	1020	21	1016	20	19
orsirr2	721	22	720	22	718	21	19
sherman1	437	33	437	32	437	32	28
sherman3	(a)	82	(a)	82	(a)	82	76
sherman4	134	35	134	35	134	34	33
sherman5	(a)	24	(a)	23	(a)	23	22
saylr1	(a)	14	(a)	14	(a)	14	11
saylr3	435	33	435	32	435	32	28
saylr4	(a)	50	(a)	50	(a)	49	44

(U) No preconditioning

(P) ILUT(0) preconditioning used

(a) > 2000 iterations

From Table 4.1, it can also be observed that the number of iterations needed for the BQMR(k) algorithms in most cases is not much more than the GMRES algorithm. The number of iterations for BQMR(k) decreases typically when k increases from one to three. However, Table 4.1 does not reflect the total cost for each method on a given problem, since the different methods have unequal cost per iteration.

If the total operation count is the sole criterion for determining efficiency, then BQMR(1) is more efficient than BQMR(2) and BQMR(3) in most problems considered here. However, for some problems such as orsreg1, BQMR(2) with preconditioning is more efficient. An analysis is given as follows.

- orsreg1 has $N = 2205$ and number of nonzeros $NNZ = 14133$.
- A matrix vector multiply takes $NNZ/N \times 2 \approx 12.8N$ operations.
- There are three matrix vector multiply and two ILUT(0) preconditioning per iteration $\Rightarrow$ cost = $12.8N \times 5 = 64N$ operations.
- Total operation count per iteration for BQMR(1) = $64N + 28N = 92N$.
- Total operation count per iteration for BQMR(2) = $64N + 32N = 96N$.
- Total operation count for BQMR(1) = $92N * 29 = 2668N$ operations.

- Total operation count for BQMR(2) = $96N * 27 = 2592N$ operations.

Even though BQMR(1) seems to be more efficient in most problems, there are other factors that can affect performance. For example, it is well known that matrix vector multiplication and preconditioning are much more expensive than vector operations, especially on some parallel computers and on supercomputers without efficient gather/scatter hardware support. Thus, on these advanced computers, BQMR(2) and BQMR(3) are potentially more efficient, in addition to giving smoother convergence curves than BQMR(1).

The convergence history of a convection diffusion equation cde63 is also plotted as shown in Fig. 1. The plots basically show that BQMR(1) does help to smooth out the irregular convergence behavior of BCG; and BQMR(2) and BQMR(3) most of the time give an even smoother convergence behavior.

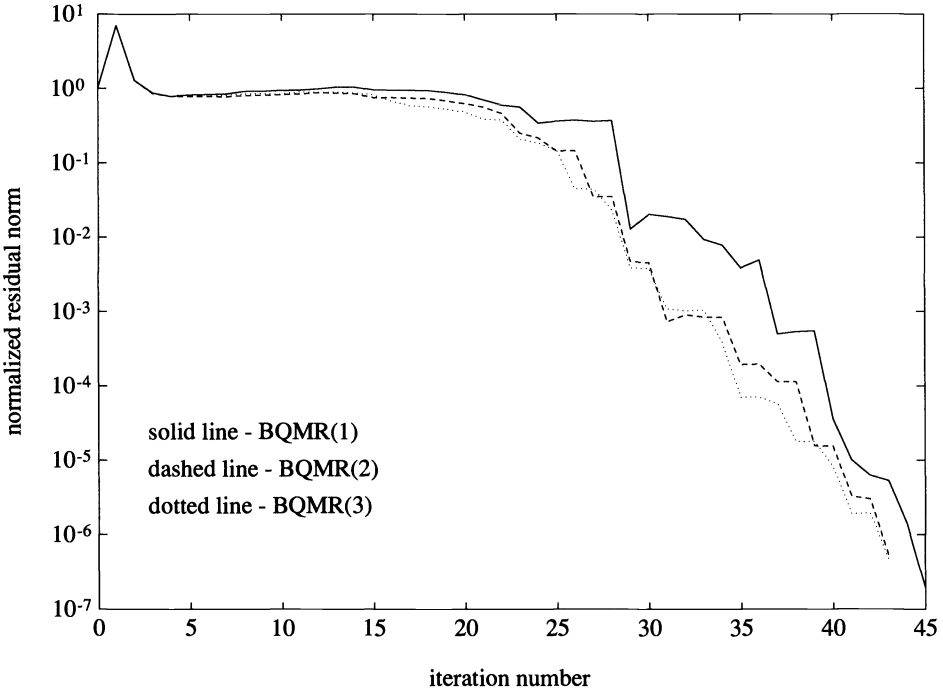


FIG. 1. *Convergence history for the* cde63 *problem* (ILUT(0) *preconditioned*).

5. Transpose-free quasi-minimal residual algorithms.

5.1. A transpose-free QMR algorithm based on BiCGSTAB. In many engineering applications the matrix vector product $A^T y$ is not easily computable. This, together with other efficiency issues, has motivated the transpose-free Lanczos-based algorithms. The first of these is CGS [16] that often gives extremely irregular convergence curves. Since Freund and Nachtigal [9] have shown that quasi minimization does help to give a smoother convergence behavior for BCG, quasi minimization has also been applied to transpose-free methods giving rise to the several transpose-free QMR methods such as Freund's transpose-free QMR method [7], the transpose-free implementation of the simplified QMR method of Chan, De Pillis, and Van der Vorst [3], and the transpose-free QMR method based on the BiCGSTAB algorithm of

Chan et al. [4] (QMRCGSTABx). Since this paper proposes variants of the QMRCGSTAB1 method, only this method will be described in detail.

QMRCGSTAB1 was developed in the same spirit as Freund developed his transpose-free QMR method from CGS. QMRCGSTAB1 was derived based on the matrix representation of Van der Vorst's BiCGSTAB algorithm [18]. In particular, the vectors generated by the BiCGSTAB process can be put into the following matrix form

$$AY_m = W_{m+1}L_m^e, \tag{5.1}$$

where $Y_m = [p_1\ s_1\ p_2\ s_2 \cdots]$, $W_m = [r_0\ s_1\ r_1\ s_2 \cdots]$, and p_i's, r_i's, and s_i's are defined in the BiCGSTAB algorithm [18]. Moreover, L_m^e is the product of an $(m+1) \times m$ bidiagonal matrix (which has unity diagonal and -1's on the subdiagonal) with an $m \times m$ diagonal matrix $(= \text{diag}[\alpha_1^{-1}\omega_1^{-1}\alpha_2^{-1}\omega_1^{-1} \cdots])$. The solution can be written in the form

$$x_m = x_0 + Y_m z \quad \text{for some } z \in \Re^m, \tag{5.2}$$

and the corresponding residual is

$$\begin{aligned} r_m &= r_0 - AY_m z \\ &= W_{m+1}(e_1^{(m+1)} - L_m^e z). \\ &= W_{m+1}\Omega_{m+1}^{-1}[\Omega_{m+1}(e_1^{(m+1)} - L_m^e z)]. \end{aligned} \tag{5.3}$$

The idea of QMRCGSTAB1 is to minimize the expression $\Omega_{m+1}(e_1^{(m+1)} - L_m^e z)$. Having the knowledge of the L_m^e matrix, quasi minimization can be carried out the same way it was done in deriving Freund's transpose-free QMR algorithm [7].

5.2. A family of transpose-free quasi-minimal residual algorithms. The generalization of the quasi-minimization step described in §2 can also be applied to the transpose-free Lanczos-based algorithms. In particular, we choose to start with the QMRCGSTAB1 algorithm [4]. (We can also start with the CGS algorithm giving rise to variants of Freund's transpose-free QMR algorithm.) In QMRCGSTAB1, Ω_{m+1} is a diagonal matrix. In general, Ω_{m+1} does not have to be a diagonal matrix. Again, if we let $\Omega_{m+1}^{-1} = B_{m+1}^{(k)}$ to be upper triangular such that $W_{m+1}B_{m+1}^{(k)}$ is orthonormal, we have a minimal residual algorithm. However, this is too expensive to compute. The strategy we propose in this paper is to determine a relatively sparse $B_{m+1}^{(k)}$ such that $W_{m+1}B_{m+1}^{(k)}$ is close to orthonormal. For example, we can orthonormalize s_1 against r_0. (These vectors are defined in [18], s_2 against r_1, and so on, so that $W_{m+1}B_{m+1}^{(k)}$ is pairwise orthonormal (having $\lceil (m+1)/2 \rceil$ subblocks where each subblock is orthonormal).) To accomplish this, $B_{m+1}^{(k)}$ is constructed to be block diagonal with block size 2 (each block is 2×2 upper triangular). We call this QMRCGSTAB(2). (To be consistent in naming the methods, QMRCGSTAB1 will subsequently be called QMRCGSTAB(1). We can further generalize this approach by generating orthonormal groups in W_{m+1} of size k. Again this can be accomplished by using a sparse $B_{m+1}^{(k)}$ weight matrix to be block diagonal with block size k. In this paper we also implement the case when $k = 4$ and we call this QMRCGSTAB(4).

6. Implementation and complexity of the transpose-free QMR algorithms.

6.1. Implementation details. Considering the QMRCGSTAB(k) algorithms, solving the least squares problem basically involves three steps.

Step 1. Update B_{m+1}^k.

Step 2. Update the relatively sparse matrix $\tilde{L}_m^e = (B_{m+1}^{(k)})^{-1}L_m^e$.

Step 3. Perform QR decomposition on $\tilde{L}_m^e$ to get R_m, and apply the same Q_m to $\rho_0 e_1^{(m+1)}$ to get $t_m = [\tau_1 \cdots \tau_m]^T$.
The solution can then be updated by

$$x_m = x_{m-1} + Y_m R_m^{-1}[0 \cdots 0\ \tau_m]^T. \tag{6.1}$$

The details of the QMRCGSTAB(k) algorithms for $k = 2, 4$ are described in the following sections.

6.1.1. Implementation of QMRCGSTAB(2) algorithms. The pseudocode for the QMRCGSTAB(2) algorithm is as follows.

ALGORITHM QMRCGSTAB(2)
(1) Initialization
(a) $r_0 = b - Ax_0$; $\tilde{r}_1$ arbitrary, $\tilde{r}_0 r_0 \neq 0$
(b) $p_0 = v_0 = d_0 = 0$; $\rho_0 = \alpha_0 = \omega_0 = 1$; $\tilde{\tau} = \tau = \| r_0 \|$
(c) $s_{-1} = s_0 = 0$; $c_{-1} = c_0 = -1$; $\epsilon_{11} = 1/\| r_0 \|$; $w_0 = \epsilon_{11} r_0$
(2) **For** $n = 1, 2, \ldots$ do
(a) $\rho_n = \tilde{r}_0^T r_{n-1}$; $\beta_n = (\rho_n \alpha_{n-1})/(\rho_{n-1}\omega_{n-1})$
(b) $p_n = r_{n-1} + \beta_n(p_{n-1} - \omega_{n-1} v_{n-1})$
(c) $v_n = Ap_n$; $\alpha_n = \rho_n/(\tilde{r}_0^T v_n)$; $y_n = r_n - \alpha_n v_n$
(3) U-solve to get the modified B matrix
$m = 2n - 1$; $\tilde{w}_m = y_n - (y_n^T w_{m-1}) w_{m-1}$; $\epsilon_{22} = 1/\| \tilde{w}_m \|$; $w_m = \epsilon_{22}\tilde{w}_m$
$\epsilon_{12} = -(y_n^T w_{m-1})\epsilon_{22}\epsilon_{11}$; $\mu_1 = -1/(\alpha_n \epsilon_{22})$; $\mu_2 = 1/[(\alpha_n - \epsilon_{12}\mu_1)\epsilon_{11}]$
$\nu_2 = s_{m-1}\mu_2$; $\delta_1 = -c_{m-1}\mu_2$
$\tilde{d}_m = p_n - \nu_2 d_{m-1}$
(4) Continue quasi minimization
if $(|\mu_1| > |\delta_1|)$ $\xi = -\delta_1/\mu_1$; $s_m = 1/\sqrt{1+\xi^2}$; $c_m = s_m \xi$
else $\xi = -\mu_1/\delta_1$; $c_m = 1/\sqrt{1+\xi^2}$; $s_m = c_m \xi$
$\nu_1 = -c_m\delta_1 + s_m\mu_1$; $\tau = -c_m\tilde{\tau}$; $\tilde{\tau} = -s_m\tilde{\tau}$; $d_m = \tilde{d}_m/\nu_1$
$x_m = x_{m-1} + \tau d_m$
(5) Continue with BiCGSTAB
$t_n = Ay_n$
$\omega_n = (y_n^T t_n)/(t_n^T t_n)$
$r_n = y_n - \omega_n t_n$
(6) second U-solve to get the modified B matrix
$m = 2n$; $\epsilon_{33} = 1/\| r_n \|$; $w_m = \epsilon_{33} r_n$
$\mu_1 = -1/(\omega_n \epsilon_{33})$; $\mu_2 = 1/(\omega_n \epsilon_{22})$
$\mu_3 = -(\epsilon_{12}\mu_2)/\epsilon_{11}$; $\nu_3 = s_{m-2}\mu_3$; $\delta_2 = -c_{m-2}\mu_3$; $\nu_2 = -c_{m-1}\delta_2 + s_{m-1}\mu_2$
$\epsilon_{11} = \epsilon_{33}$; $\delta_1 = -s_{m-1}\delta_2 - c_{m-1}\mu_2$, $\tilde{d}_m = y_n - \nu_3 d_{m-2} - \nu_2 d_{m-1}$
(7) Continue quasi minimization
if $(|\mu_1| > |\delta_1|)$ $\xi = -\delta_1/\mu_1$; $s_m = 1/\sqrt{1+\xi^2}$; $c_m = s_m \xi$
else $\xi = -\mu_1/\delta_1$; $c_m = 1/\sqrt{1+\xi^2}$; $s_m = c_m \xi$
$\nu_1 = -c_m\delta_1 + s_m\mu_1$; $\tau = -c_m\tilde{\tau}$; $\tilde{\tau} = -s_m\tilde{\tau}$; $d_m = \tilde{d}_m/\nu_1$
$x_m = x_{m-1} + \tau d_m$
(8) If x_m is accurate enough, stop
(9) **End for**

6.1.2. Implementation of QMRCGSTAB(4) algorithm. The pseudocode for the QMRCGSTAB(4) algorithm is as follows.

ALGORITHM QMRCGSTAB(4)

(1) Initialization

(a) $r_0 = b - Ax_0$; $\tilde{r}_1$ arbitrary, $\tilde{r}_0 r_0 \neq 0$

(b) $p_0 = v_0 = d_0 = 0$; $\rho_0 = \alpha_0 = \omega_0 = 1$; $\tilde{\tau} = \tau = \| r_0 \|$

(c) $s_{-3} = s_{-2} = s_{-1} = s_0 = 0$; $c_{-3} = c_{-2} = c_{-1} = c_0 = -1$

(d) $\epsilon_{11} = 1/\| r_0 \|$; $w_0 = \epsilon_{11} r_0$

(2) **For** $n = 1, 2, \ldots$ do

same as step (2a)–(2c) in Algorithm QMRCGSTAB(2)

(3) U-solve to get the modified B matrix

(a) if mod(n,2) = 1 ($m = 2n - 1$)

$\tilde{w}_m = y_n - (y_n^T w_{m-1}) w_{m-1}$; $\epsilon_{22} = 1/\| \tilde{w}_m \|$; $w_m = \epsilon_{22} \tilde{w}_m$

$\epsilon_{12} = -(y_n^T w_{m-1}) \epsilon_{22} \epsilon_{11}$; $\mu_1 = -1/(\alpha_n \epsilon_{22})$; $\mu_2 = 1/[(\alpha_n - \epsilon_{12}\mu_1)\epsilon_{11}]$

$\nu_2 = s_{m-1}\mu_2$; $\delta_1 = -c_{m-1}\mu_2$

$\tilde{d}_m = p_n - \nu_2 d_{m-1}$

(b) else if mod(n,2) = 0

$\tilde{w}_m = y_n - (y_n^T w_{m-1}) w_{m-1} - (y_n^T w_{m-2}) w_{m-2} - (y_n^T w_{m-3}) w_{m-3}$

$\epsilon_{44} = 1/\| \tilde{w}_m \|$; $w_m = \epsilon_{44} \tilde{w}_m$

$\epsilon_{14} = -\epsilon_{44}[(y_n^T w_{m-1})\epsilon_{11} + (y_n^T w_{m-2})\epsilon_{12} + (y_n^T w_{m-3})\epsilon_{13}]$

$\epsilon_{24} = -\epsilon_{44}[(y_n^T w_{m-2})\epsilon_{22} + (y_n^T w_{m-3})\epsilon_{33}]$

$\epsilon_{34} = -\epsilon_{44}(y_n^T w_{m-3})\epsilon_{33}$; $\mu_1 = -1/(\alpha_n \epsilon_{44})$; $\mu_2 = (1/\alpha_n - \epsilon_{34}\mu_1)/\epsilon_{33}$

$\mu_3 = -(\epsilon_{24}\mu_1 + \epsilon_{23}\mu_2)/\epsilon_{22}$; $\mu_4 = -(\epsilon_{14}\mu_1 + \epsilon_{13}\mu_2 + \epsilon_{12}\mu_3)/\epsilon_{11}$

$\nu_4 = -c_{m-3}\mu_4$; $\delta_3 = -c_{m-3}\mu_4$; $\nu_3 = -c_{m-2}\delta_3 + s_{m-2}\mu_3$

$\delta_2 = -s_{m-2}\delta_3 - c_{m-2}\mu_3$

$\nu_2 = -c_{m-1}\delta_2 + s_{m-1}\mu_2$; $\delta_1 = -s_{m-1}\delta_2 - c_{m-1}\mu_2$

$\tilde{d}_m = p_n - \nu_4 d_{m-3} - \nu_3 d_{m-2} - \nu_2 d_{m-1}$

(4) Continue quasi minimization–same as step (4) in QMRCGSTAB(2)

(5) Continue with BiCGSTAB–same as step (5) in QMRCGSTAB(2)

(6) second U-solve to get the modified B matrix

(a) if mod(n,2) = 1 ($m = 2n - 1$)

$\tilde{w}_m = r_n - (r_n^T w_{m-1}) w_{m-1} - (r_n^T w_{m-2}) w_{m-2}$

$\epsilon_{33} = 1/\| \tilde{w}_m \|$; $w_m = \epsilon_{33} \tilde{w}_m$

$\epsilon_{13} = -\epsilon_{33}[(r_n^T w_{m-1})\epsilon_{11} + (r_n^T w_{m-2})\epsilon_{12}]$; $\epsilon_{23} = -(r_n^T w_{m-2})\epsilon_{22}\epsilon_{33}$

$\mu_1 = -1/(\omega_n \epsilon_{33})$; $\mu_2 = (1/\omega_n - \epsilon_{23}\mu_1)/\epsilon_{22}$

$\mu_3 = -(\epsilon_{13}\mu_1 + \epsilon_{12}\mu_2)/\epsilon_{11}$; $\nu_3 = s_{m-2}\mu_3$; $\delta_2 = -c_{m-2}\mu_3$

$\nu_2 = -c_{m-1}\delta_2 + s_{m-1}\mu_2$; $\delta_1 = -s_{m-1}\delta_2 - c_{m-1}\mu_2$

$\tilde{d}_m = y_n - \nu_3 d_{m-2} - \nu_2 d_{m-1}$

(b) else if mod(n,2) = 0 ($m = 2n$)

$\epsilon_{55} = 1/\| r_n \|$; $w_m = \epsilon_{55} r_n$; $\mu_1 = -1/(\omega_n \epsilon_{55})$; $\mu_2 = 1/(\omega_n \epsilon_{44})$

$\mu_3 = -(\epsilon_{34}\mu_2)/\epsilon_{33}$; $\mu_4 = -(\epsilon_{24}\mu_2 + \epsilon_{23}\mu_3)/\epsilon_{22}$

$\mu_5 = -(\epsilon_{14}\mu_2 + \epsilon_{13}\mu_3 + \epsilon_{12}\mu_4)/\epsilon_{11}$; $\epsilon_{11} = \epsilon_{55}$

$\nu_5 = -c_{m-4}\mu_5$; $\delta_4 = -c_{m-4}\mu_5$

$\nu_4 = -c_{m-3}\delta_4 + s_{m-3}\mu_4$; $\delta_3 = -s_{m-3}\delta_4 - c_{m-3}\mu_4$

$\nu_3 = -c_{m-2}\delta_3 + s_{m-2}\mu_3$; $\delta_2 = -s_{m-2}\delta_3 - c_{m-2}\mu_3$

$\nu_2 = -c_{m-1}\delta_2 + s_{m-1}\mu_2$; $\delta_1 = -s_{m-1}\delta_2 - c_{m-1}\mu_2$

$\tilde{d}_m = y_n - \nu_5 d_{m-4} - \nu_4 d_{m-3} - \nu_3 d_{m-2} - \nu_2 d_{m-1}$

(7) Continue quasi minimization

if $(|\mu_1| > |\delta_1|)$ $\xi = -\delta_1/\mu_1$; $s_m = 1/\sqrt{1 + \xi^2}$; $c_m = s_m \xi$

else $\xi = -\mu_1/\delta_1$; $c_m = 1/\sqrt{1 + \xi^2}$; $s_m = c_m \xi$

$\nu_1 = -c_m\delta_1 + s_m\mu_1$; $\tau = -c_m\tilde{\tau}$; $\tilde{\tau} = -s_m\tilde{\tau}$; $d_m = \tilde{d}_m/\nu_1$

$x_m = x_{m-1} + \tau d_m$

(8) If x_m is accurate enough, stop

(9) **End for**

6.2. Complexity and storage analysis. Table 6.1 shows the operation counts for the QMRCGSTAB(1), QMRCGSTAB(2), and QMRCGSTAB(4) algorithms. Since the operation counts for the QMRCGSTAB(2) algorithm are different between odd and even iterations, only their average is listed. The same is true for QMRCGSTAB(4).

TABLE 6.1
Complexity and storage statistics per iteration.

Method	(1)	(2)	(3)	(4)	Total op. count	Storage
QMRCGSTAB(1)	2	6	8	0	28N+2 (MVP + P)	8N+A
QMRCGSTAB(2)	2	7	10	4	38N+2 (MVP + P)	9N+A
QMRCGSTAB(4)	2	9	14	4	50N+2 (MVP + P)	11N+A

(1) Number of matrix vector multiplications

(2) Number of inner product calculations

(3) Number of daxpy operations

(4) Number of other (e.g., vector add) operations

A = storage for the A matrix

P = preconditioning

MVP = matrix vector products.

7. Numerical experiments for the transpose-free QMR algorithms. The set of numerical examples used here is the same as in §4. Table 7.1 shows the iteration counts for the different algorithms. The initial solution x_0 is a random vector, $\tilde{r}_0 = r_0$, and the stopping criterion is $\| r_k \| / \| b \| < 10^{-8}$.

It can be observed from Table 7.1 that the number of iterations needed for the QMRCGSTAB (k) algorithms in most cases is about half of those of GMRES when preconditioning is used. Again, the number of iterations for QMRCGSTAB(k) is in most cases slightly lower when larger k is used. Again, due to unequal costs per iteration between the methods, iteration counts alone do not reflect the actual efficiency, which is affected by many other factors such as total operation counts and the target computer architecture on computers where matrix vector multiplication is much more expensive than vector operations (as on some supercomputers where efficient gather/scatter hardware is not available). Hence again, in addition to giving smoother convergence curves, the QMRCGSTAB(k) algorithms are potentially more efficient.

In addition to the iteration counts, the convergence history for a convection diffusion equation cde63 is also plotted in Fig. 2.

8. Concluding remarks. In this paper, we explored the use of different weight matrices in the quasi-minimal residual algorithms, and we derived several QMR variants such as BQMR(2) and BQMR(3) algorithms and transpose-free QMR variants such as QMRCGSTAB(2) and QMRCGSTAB(4) algorithms. These variants were shown to represent a few of the many possibilities lying in the spectrum of quasi-minimal residual (and transpose-free

TABLE 7.1
Iteration counts for different problems.

Problem	A		B		C		GMRES
	(U)	(P)	(U)	(P)	(U)	(P)	(P)
cde31	65	16	65	16	64	16	24
cde63	119	26	119	26	118	25	39
orsreg1	308	16	305	15	303	15	24
osrirr1	1437	12	1383	12	1358	11	19
orsirr2	882	12	824	12	804	12	19
sherman1	400	18	369	18	362	18	28
sherman3	(a)	65	(a)	65	(a)	64	76
sherman4	110	25	106	23	105	23	33
sherman5	1848	15	1843	14	1812	14	22
saylr1	(a)	9	(a)	9	(a)	8	11
saylr3	345	18	353	18	348	18	28
saylr4	(a)	33	(a)	32	(a)	31	44

(A) QMRCGSTAB(1)

(B) QMRCGSTAB(2)

(C) QMRCGSTAB(4)

(U) No preconditioning

(P) ILUT(0) preconditioning used

(a) > 2000 iterations

QMR) algorithms where on the two ends are nonlook-ahead QMR and a GMRES-like minimal residual algorithm, respectively. We showed that these variants are only slightly more expensive than BQMR(1) and QMRCGSTAB(1) per iteration but they generally give smoother convergence curves. Moreover, in general these variants also give a slightly faster convergence rate. Thus these variants can potentially be more efficient for the following situations:

- when the reduction in iteration count is such that the total operation count is lower (e.g., orsreg1 for BQMR(2));
- when the percentage of nonzeros in the problem matrix is relatively high so that matrix vector multiply is relatively more expensive than the overhead for doing quasi minimization;
- when more powerful preconditioners are used (e.g., ILUT(k) for large k) so that the overhead for doing quasi minimization is relatively inexpensive;
- on certain computer platform (e.g., on some parallel computers or supercomputers with no gather/scatter hardware) where matrix vector multiply is much more expensive than daxpy and dot products.

Acknowledgments. The author would like to thank Roland Freund and Noel Nachtigal for their permission to use their ILUT subroutine and some of their support subroutines. Their comments on this work are also well taken. Most matrix and vector operations are performed through the use of the subroutines available in SPARSKIT. The author would also like to thank

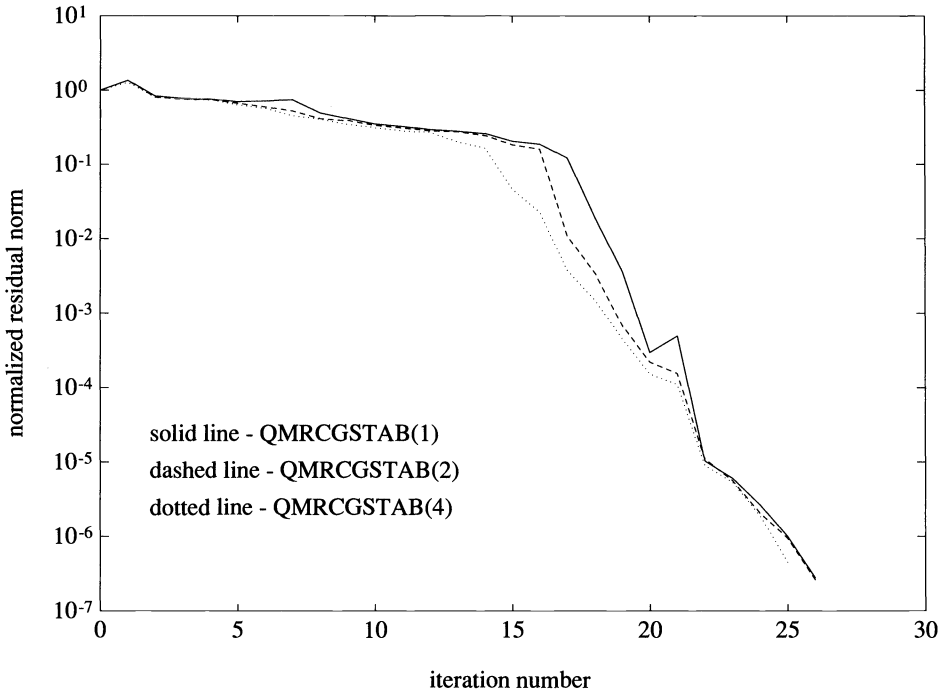


FIG. 2. *Convergence history for the* cde63 *problem* (ILUT(0) *preconditioned*).

Tony Chan for inspiring this work and Raymond Tuminaro for many helpful discussions. Finally, the constructive comments from Martin Gutknecht are greatly appreciated.

REFERENCES

[1] A. T. CHRONOPOULOS AND S. MA, *On Squaring Krylov Subspace Iterative Methods for Nonsymmetric Linear Systems*, Tech. Report 89-67, Computer Science Department, Univ. of Minnesota at Minneapolis, 1989.

[2] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[3] T. F. CHAN, L. DE PILLIS, AND H. VAN DER VORST, *A Transpose-Free Squared Lanczos Algorithm and Application to Solving Nonsymmetric Linear Systems*, CAM Report 91-17, Dept. Mathematics, UCLA, Oct. 1991.

[4] T. F. CHAN, E. GALLOPOULOS, V. SIMONCINI, T. SZETO, AND C. H. TONG, *QMRCGSTAB: A Quasi-minimal Residual Variant of the BiCGSTAB Algorithm for Nonsymmetric Systems*, Center for Supercomputing Research and Development Report 1231, University of Illinois at Urbana-Champaign, May 1992.

[5] 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.

[6] R. FLETCHER, *Conjugate Gradient Methods for Indefinite Systems*, in Proc. Dundee Conference on Numerical Analysis, 1975, Lecture Notes in Mathematics 506, G. A. Watson, ed., Springer-Verlag, Berlin, 1976, pp. 73–89.

[7] R. W. FREUND, *A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems*, Research Institute for Advanced Computer Science Tech. Report 91.18, NASA Ames Research Center, Moffett Field, CA, Sept. 1991.

[8] R. W. FREUND, M. H. GUTKNECHT, AND N. M. NACHTIGAL, *An Implementation of the Look-ahead Lanczos Algorithm for Non-Hermitian Matrices*, Tech. Report 91-09, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, CA, 1991.

[9] 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.

[10] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 409–436.

[11] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 33–53.

[12] Y. SAAD, *The Lanczos biorthogonalization algorithm and other oblique projection methods for solving large unsymmetric systems*, SIAM J. Numer. Anal., 19 (1982), pp. 485–506.

[13] Y. SAAD, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.

[14] Y. SAAD AND M. H. SCHULTZ, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*, Math. Comput., 44 (1985), pp. 417–424.

[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] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[17] C. H. TONG, *A Comparative Study of Preconditioned Lanczos Methods for Nonsymmetric Linear Systems*, Sandia Report, SAND91-8240B Sandia National Laboratries, Livermore, CA, 1992.

[18] H. A. VAN DER VORST, *Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, Preprint, Univ. of Utrecht, the Netherlands, Sept. 1990.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 106–126, January 1994

007

PHASE FIELD COMPUTATIONS OF SINGLE-NEEDLE CRYSTALS, CRYSTAL GROWTH, AND MOTION BY MEAN CURVATURE*

G. CAGINALP† AND E. SOCOLOVSKY‡

Abstract. The phase field model for free boundaries consists of a system of parabolic differential equations in which the variables represent a phase (or "order") parameter and temperature, respectively. The parameters in the equations are related directly to the physical observables including the interfacial width ϵ, which we can regard as a free parameter in computation. The phase field equations can be used to compute a wide range of sharp interface problems including the classical Stefan model, its modification to incorporate surface-tension and/or surface kinetic terms, the Cahn–Allen motion by mean curvature, the Hele–Shaw model, etc. Also included is anisotropy in the equilibrium and dynamical forms generally considered by materials scientists. By adjusting the parameters, the computations can be varied continuously from single-needle dendritic to faceted crystals.

The computational method consists of smoothing a sharp interface problem within the scaling of distinguished limits of the phase field equations that preserve the physically important parameters. The two-dimensional calculations indicate that this efficient method for treating these stiff problems results in very accurate interface determination without interface tracking. These methods are tested against exact and analytical results available in planar waves, faceted growth, and motion by mean curvature up to extinction time.

The results obtained for the single-needle crystal show a constant velocity growth, as expected from laboratory experiments.

Key words. phase field, interface, viscosity methods, solidification, computation of free boundaries

AMS subject classifications. 65M06, 82B24, 35K57

1. Introduction.

The problem of numerical computation of a moving boundary has been under active study from a number of perspectives (see [Gl] for a survey). The phase field approach to free boundaries has two appealing features: (i) a broad spectrum of distinct problems that can be studied by means of a single set of equations, and (ii) the interface in these problems does not need to be tracked explicitly, as the computations use a system of parabolic equations in which the interface has been diffused.

The phase field equations specifically identify all macroscopic parameters, e.g., latent heat, diffusivity, etc., and a parameter ϵ that is a measure of interface thickness.

We begin by defining a set of sharp interface problems in a spatial region $\Omega \subset \mathbb{R}^n$ with the interface defined as $\Gamma(t)$ with $t \in \mathbb{R}^+$. A basic problem is to find a temperature function $T(x, t)$ and a curve (or surface) $\Gamma(t) \subset \Omega$ that satisfy the equations

$$\rho c_{\text{spm}} T_t = K \Delta T \quad \text{in } \Omega \setminus \Gamma(t), \tag{1.1}$$

$$\rho \ell v = K \nabla T \cdot n]_+^- \quad \text{on } \Gamma(t), \tag{1.2}$$

$$T - T_E = -\frac{\sigma}{[s]_E}\kappa - \alpha \frac{\sigma v}{[s]_E} \quad \text{on } \Gamma(t). \tag{1.3}$$

$T_E :=$ equilibrium melting temperature,
$\ell :=$ latent heat per unit mass,
$K :=$ thermal conductivity,
$\sigma :=$ surface tension between two phases,
$\rho :=$ density,

*Received by the editors July 27, 1992; accepted for publication (in revised form) February 23, 1993.

†Mathematics and Statistics Department, University of Pittsburgh, Pittsburgh, Pennsylvania 15260. This author was supported by National Science Foundation grant DMS-9002242.

‡Center for Nonlinear Analysis, Mathematics Department, Hampton University, Hampton, Virginia 23668.

$c_{\text{spm}} :=$ specific heat per unit mass,

$[s]_E :=$ entropy difference between phases per unit volume,

$v :=$ normal velocity (positive if motion is directed toward liquid),

$\kappa :=$ sum of principal curvatures at the particular point on the interface,

$\nabla T \cdot n]_+^- :=$ jump in the normal component of the temperature (from solid to liquid),

$\alpha :=$ kinetic undercooling coefficient.

By defining a dimensionless temperature, $u := (T - T_E)c_{\text{spm}}/\ell$, a diffusion parameter $D := K/\rho c_{\text{spm}}$, and a "capillary length" $d_0 := \sigma c_{\text{spm}}/([s]_E \ell)$, one can rewrite (1.1)–(1.3) as

$$u_t = D\Delta u \quad \text{in } \Omega \setminus \Gamma(t), \tag{1.1'}$$

$$v = D\nabla u \cdot n]_+^- \quad \text{on } \Gamma(t), \tag{1.2'}$$

$$u = -d_0\kappa - \alpha d_0 v \quad \text{on } \Gamma(t). \tag{1.3'}$$

Note that the parameter D simply adjusts the time versus length scales in the problem. Thus one could set $D := 1$ by simply adjusting α (which has units of time/(space)^2, as does $\frac{1}{D}$). Within this scaling, the latent heat ℓ has been incorporated into the dimensionless temperature so that the coefficient of v in (1.2′) is unity. For the limiting situations such as Hele–Shaw in which latent heat approaches zero, one must adopt a scaling such as (1.1)–(1.3), which avoids division by ℓ.

The oldest mathematical model for an interface between two phases, e.g., liquid and solid, is known as the classical Stefan model and incorporates heat diffusion (1.1) and the latent heat due to fusion (1.2). Furthermore, it stipulates that the interfacial temperature remains at the melting temperature, i.e., $u = 0$ replaces (1.2). This last condition amounts to setting $\sigma = 0$ in (1.3) or $d_0 = 0$ in (1.3′). Although the capillarity length d_0 is often very small compared to the other length scales in the problem, such as κ^{-1}, it is nevertheless very significant as a stabilizing force [Ch].

The classical Stefan model's neglect of surface tension and surface kinetics can be partially remedied by the imposition of (1.3) with finite σ and α. The numerical study of the surface tension and kinetics model (1.1)–(1.3) then becomes even more challenging as a consequence of the curvature term in (1.3).

Another sharp interface problem that arises in the context of a fluid between two plates is the Hele–Shaw model, which can be expressed as equations (1.1)–(1.3) with the left-hand side of (1.1) set at zero. For the Hele–Shaw problem, u describes pressure, and the constants ℓ, κ, etc. represent different physical quantities (see [Oc], [Cr] and references contained therein).

Finally, one may consider the Cahn–Allen model of motion by mean curvature problem [AC], in which the interface moves with normal velocity

$$v = \frac{-1}{\alpha}\kappa \tag{1.4}$$

at each point on the interface $\Gamma(t)$.

Anisotropy may be considered in all of the problems above. In typical materials, the anisotropy is present in the surface tension σ and in the kinetic coefficient α so that (1.3′) becomes

$$u = -d_0(\theta)\kappa - \alpha(\theta)d_0(\theta)v, \tag{1.5}$$

where α is now a function of the orientation angle θ, and

$$d_0(\theta) := \frac{\sigma(\theta) + \sigma''(\theta)}{[s]_E \ell} c_{\text{spm}}. \tag{1.5'}$$

The term $\beta(\theta) := \alpha(\theta) d_0(\theta)$ is often called the mobility (see [Ta] for general discussion). The orientation angle θ may be defined as the angle between the normal to the interface and a fixed direction, or the angle between a fixed direction and the line from a central point to a point of the interface.

The precise nature of anisotropy and its interaction with undercooling (liquid below the equilibrium freezing temperature) is important in determining the shape of the interface, and one can obtain both a single-needle crystal and cubic crystal growth from a cubic symmetry material (see §5 for details).

Another approach to free boundary problems involves the use of a phase or "order" parameter φ that specifies the phases ($\varphi \backsimeq 1$ is liquid, $\varphi \backsimeq -1$ is solid) while the level set $\varphi = 0$ is now the interface. This approach has its origins in Landau theory [LL], Cahn–Hilliard type equations [CaHi], [La], and the application of mean field theory to critical phenomena [HH].

Phase field theories are generally rooted in the idea that each molecule or "spin" is under the influence of all others that effectively constitute an average "field" of interactions. This, of course, is a vast simplification over evaluating the sum over all possible states to calculate the partition function and, thereby, the free energy. Preliminary reports on a system of equations closely related to those discussed here (but with a different scaling that does not precisely attain any of the sharp interface models defined above) include [Ca] and [F].

The variable φ represents the mean field or "averaged" description of the phase at each point (t, x) in time-space. This mean field approach has been used with some, but not complete, success in the subtle area of equilibrium critical phenomena. The application to ordinary phase transitions is actually quite different in that the correlation length (i.e., mean distance at which molecules are "aware" of one another's presence) is very small, whereas it is very large in the critical region. In fact, the large correlation length provided the original justification for this approach. (See references in [Ca2] for further discussion.) Another difference between the critical region and ordinary phase transitions is that critical phenomena can rely on universality, which states that the critical exponents should be independent of the details of the system. For solidification, it is generally desirable to specify all material parameters and to obtain quantitatively reliable results.

Using the scaling introduced in [Ca1], we write the phase field equations in the form

$$c_{\text{spm}} T_t + \frac{\ell}{2} \varphi_t = \frac{K}{\rho} \Delta T, \tag{1.6}$$

$$\alpha \epsilon^2 \varphi_t = \epsilon^2 \Delta \varphi + \frac{1}{2}(\varphi - \varphi^3) + \frac{\epsilon}{3} \frac{[s]_E}{\sigma} (T - T_E), \tag{1.7}$$

where ϵ is a parameter that is a measure of the interface width and all other constants are as defined earlier. In 1985 it was shown [Ca1], using asymptotics, that the major free boundary problems could be attained as distinguished limits of the phase field equations in the form (1.6), (1.7). In particular, if ϵ approaches zero while all other parameters including σ are held fixed, then the asymptotic solutions of (1.6), (1.7) are governed to leading order by the surface tension and kinetics model (1.1)–(1.3).

The classical Stefan model can be similarly recovered in the distinguished limit as *both* ϵ and σ vanish with $\epsilon/\sigma \to 0$. The formal asymptotics have been made mathematically rigorous

in some cases of special symmetry such as traveling waves [CN] and general $1d$ and radially symmetric [CC]. Other important free boundary models such as the Hele–Shaw equations and motion by mean curvature are also obtained from the phase field equations [Ca2].

Using the variables defined before (1.1′)–(1.3′), the equations (1.6), (1.7) can be written in the form

$$u_t + \frac{1}{2}\varphi_t = D\Delta u, \tag{1.6′}$$

$$\alpha\epsilon^2\varphi_t = \epsilon^2\Delta\varphi + \frac{1}{2}(\varphi - \varphi^3) + \frac{\epsilon}{3d_0}u. \tag{1.7′}$$

With d_0 and/or α possibly depending on θ, the solutions of (1.6′), (1.7′) approach those of (1.1′), (1.2′), (1.5). In a philosophical sense, this approach to solidification computation problems is similar to artificial viscosity methods [PT] in gas dynamics in that the success of the method depends on whether the interface development is independent of ϵ for some range $(0, \epsilon_0)$.

The boundary conditions imposed on (1.6′), (1.7′) are the same as the sharp interface problem for u, with compatible conditions for φ. In particular, if one imposes Dirichlet conditions $u = u_{\partial_\pm}$, then the corresponding values of φ are the roots of

$$\frac{1}{2}(\varphi_\pm - \varphi_\pm^3) + \frac{\epsilon}{3d_0}u_{\partial_\pm} = 0, \tag{1.8}$$

where the plus and minus denote the liquid and solid portions of the boundary, respectively.

In this paper we discuss the numerical study of the phase field equations with the aim of demonstrating that a single system of equations can be used to compute a broad spectrum of phenomena that are generally associated with the sharp interface problems defined above. The advantages of the phase field model include the following: (i) The variety of phenomena ranging from motion by mean curvature to stable anisotropic crystal growth to single-needle dendrites are all obtained by simply varying parameters. (ii) The system of equations has smooth solutions and does not involve explicit conditions on the interface. Hence the interface is simply the level set $\varphi = 0$ and interface tracking is not required. (iii) All of the physical parameters are clearly identified so that the system can be used to obtain quantitatively reliable numbers. (iv) Computations involving self-interactions of the interface do not pose difficulties for the phase field model.

To illustrate how intersections create problems, we consider the following example. Two solid spheres with a small space between them are surrounded by the liquid that is undercooled so that the spheres grow slowly. At some point the spheres touch and begin to solidify. If one models this situation using (1.1)–(1.3), then there is the appearance of a mathematical singularity due to (1.3) when the spheres touch. This is a mathematical artifact; however, because (1.3) is not a physically complete description of the interfaces in this complex situation, as condition (1.3) has not been defined when the classical normal is not defined. On the other hand, the phase field equations are guaranteed to have a smooth solution (φ, T) if the initial and boundary conditions are sufficiently smooth. Thus the phase field model naturally describes the evolution of intersecting interfaces, such as this example, as it is based on a free energy description of the phases rather than a macroscopic derivation for a particular (smooth) geometry. The situation may be summarized by stating that the sharp interface problems such as (1.1)–(1.3) or (1.4) are a macroscopic approximation for simple geometries, so that a discrepancy between these models and the phase field is generally resolved by noting that the sharp interface model is not a correct approximation for the physical problem when the conditions for their derivation are no longer valid.

Figure 1.1 displays the computational evolution (using the phase field equations) of two such spheres surrounded by a slightly undercooled melt. The spheres increase in size until they touch, at which point the two interfaces fuse into a single one, which then has two points of (negative) high curvature. The curvature of these points diminishes rapidly due to the interface conditions that are implicit in the phase field equations. Figure 1.2 is a computation of the same problem with crystalline anisotropy included in the form discussed at the end of §5. In this case, the initial spheres evolve toward the dynamic Wulff shape discussed in §5 for a single seed. This evolution conforms to the theoretical predictions for an interface condition with anisotropic mobility, i.e., (1.5′) with α depending on orientation. The two seeds meet at a corner and then evolve into a single Wulff shape, i.e., a new square. In both computations, there is an absence of any unphysical singularities or blow-ups. In both Figs. 1.1 and 1.2 all parameters and conditions are identical with those described in §5 for Fig. 5.1, with anisotropy excluded from the equations for Fig. 1.1.

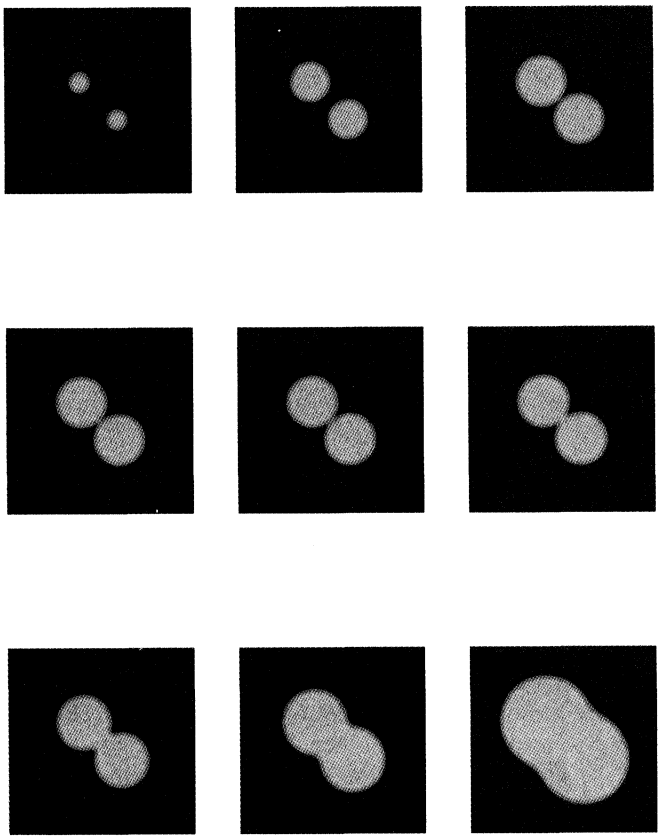


FIG. 1.1. *Self-intersection of interfaces without anisotropy. Two seeds of solid are surrounded by a slightly undercooled melt. The intersection of the interfaces poses no difficulty for the computation of the phase field equations.*

The numerical study of (1.6), (1.7) must confront the issue that the true physical size of ϵ (which is of the order of the width of the interface) is about 10^{-8} cm, while the overall domain of interest may vary between 10^{-1} cm for dendritic growth and 10^{2} cm for casting of metals. Since the grid spacing cannot be significantly larger than ϵ without large errors [CS1], [CS2], a simple application of the phase field equations with physical parameters would imply a grid of at least $10^{7} \times 10^{7}$ for dendritic growth and $10^{10} \times 10^{10}$ for casting, for example. This

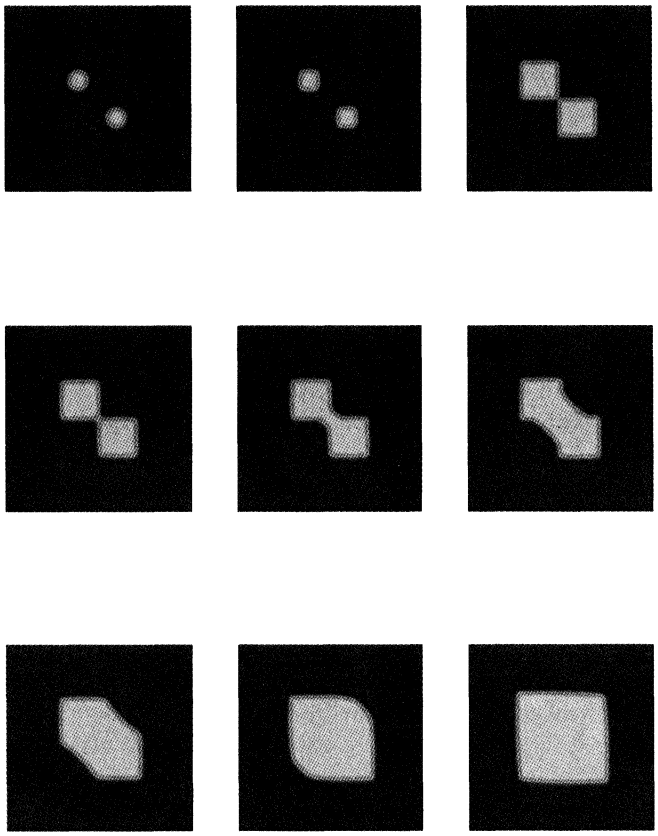


FIG. 1.2. *Self-intersections of interfaces with anisotropy. Two seeds of solid are surrounded by a slightly undercooled melt. The spheres evolve into squares (the Wulff shape) which then merge.*

grid size is prohibitive for practical computations. With this motivation, [CS1] and [CS2] studied the ansatz that ϵ can be made orders of magnitude larger (i.e., the interface can be spread) within the limitations of (a) and (b) below, without significant error, provided that this stretching is done within the proper scaling limits [Ca1], [Ca2] so that the crucial quantities such as the surface tension are not altered even by a small amount. The two key restrictions are that: (a) ϵ must be smaller than the radius of curvature; (b) ϵ cannot be large enough to destroy the double-well nature of the potential $\frac{1}{2}(\varphi - \varphi^3) \pm \frac{\epsilon}{3}\frac{1}{d_0}u$. The restriction (b) is easily remedied with a modification such as that introduced in [CC], where u is multiplied by a factor $(1-\varphi^2)^2$ or a similar function. This has the effect of trivializing the role of temperature outside of the interfacial region where it is usually very small. On the other hand, restriction (a) is fundamental and is intrinsic to the physical problem. In particular, independent of the specific method, one does not expect to resolve an interface with a radius of curvature R by means of a mesh spacing $\triangle x$ that is greater than R. Within this natural limitation, [CS1] and [CS2] showed that quantitatively reliable calculations could be performed using (1.6), (1.7).

In particular, [CS1] and [CS2] studied the one-dimensional and radially symmetric equations corresponding to (1.6), (1.7), compared the computations with the exact solutions for the limiting sharp interface problem, and examined them through self-consistency. Subject to the radius of curvature limitation, it was established that the interface could be stretched by orders of magnitude without significant loss of accuracy, even in physically delicate situations. On the other hand, even a 10% change in surface tension resulted in dramatic changes in the evolution of the interface.

In this paper, our primary aim is to show that in two dimensions a range of sharp interface problems with distinct patterns can be computed using the phase field equations. A secondary objective is to verify that the interface can be stretched with quantitative accuracy when the computations are truly higher dimensional (instead of radially symmetric equations). In the higher-dimensional case, the issue of interface stability is also relevant, so that enlarging ϵ is predicated on the expectation that it does not significantly alter stability properties. We confirm this in both the planar and spherical cases, and again we use self-consistency and comparison with exact solutions.

We begin by considering planar traveling waves for the phase field equations within the modified Stefan limit. Since the velocity of the traveling waves for the sharp interface problem is known [CCh] from the analytic solution, we compare our computations with the exact results. The traveling wave is then perturbed with a sine wave so that the stability properties can be examined in the context of the sharp interface problem. We find that a typical unstable mode in the sharp interface problem is also unstable in the phase field equations.

In §3 we consider the problem of the critical radius, which is an unstable equilibrium in solidification [Ch] and we obtain quantitative agreement with the analytical results. Anisotropic surface tension is then considered as we observe the change in interface shape as a function of both equilibrium and dynamical anisotropy. In equilibrium, these results can be compared with well-established results such as the Wulff construction approach [Wu], [Ta], [CH].

In §§4–6 we demonstrate that a broad spectrum of sharp interface problems can be approximated by the phase field model in the distinguished scaling limits as indicated in [Ca1] and [Ca2]. Section 4 features the limit of motion by mean curvature in which the interface is often characterized by reduction of curvature by becoming more circular and then shrinking until extinction. The extinction time from a circle is readily calculated analytically so that quantitative agreement of the numerical computations is verified along with the qualitative.

In §5 we consider solidification in the case of the single-needle crystal. The computations result in a single needle that moves at essentially constant velocity. Furthermore, this velocity is independent of the initial conditions. This is consistent with the known experimental [GS], [SG], and theoretical [BP], [L], and [KKL] results.

Finally, in §6, we consider solidification under conditions that favor more stable growth, so that the faceted crystal growth is observed in the anisotropic situation. It is noteworthy that a completely smooth set of equations with smooth initial data render faceted shapes, e.g., cubic. The results obtained are consistent with those obtained from the classical Wulff construction methods.

Throughout this study we use finite difference methods [CL], [CS2] with a fixed grid to isolate the issues that are of specific interest. In principle, a combination of phase field methods and adaptive grids with finite elements should provide similar accuracy at a lower cost.

Some of the other numerical studies of solidification include [LF], [SS], [K], and [RT].

2. Plane fronts, stability, and the emergence of single-needle dendrites. We study first the traveling wave solutions for the phase field equations (1.6), (1.7), the existence of which was established in [CN]. Next, we perturb the planar fronts to examine the stability properties of the interface (for which there are no analytical results). In particular, we present the growth of needle-type crystals from perturbations. These computations bridge the gap between the initial Mullins–Sekerka instability [MS] and the steady-state dendrites [BP] that evolve from them.

We consider (1.6′), (1.7′) in the distinguished limit of the surface tension and kinetics model (1.1′)–(1.3′), i.e., ϵ approaches zero with all other variables fixed. For the sharp interface problem, one can verify the existence [CCh], [DHOX] of the following traveling wave solutions

with velocity v^* and $u(t, \infty) = u_{\text{cool}}$:

$$u(t,x) = \begin{cases} u_{\text{cool}} + e^{-v^*(x-v^*t)/D} & x > v^*t, \\ u_{\text{cool}} + 1 & x \le v^*t, \end{cases} \tag{2.1}$$

$$v^* = \frac{-1}{\alpha d_0}(u_{\text{cool}} + 1). \tag{2.2}$$

To ensure the positivity of the velocity v^*, one must assume that

$$u_{\text{cool}} + 1 < 0. \tag{2.3}$$

Note that the velocity v^* and the temperature at the interface $u_{\text{cool}} + 1$ (and in fact for all x in the solid region, $x \le v^*t$) are both determined by the temperature at ∞, i.e., u_{cool}. Also, the latent heat is just unity for our temperature scale.

To state the analogous problem for the phase field equations (1.6′), (1.7′), one must impose conditions on φ as well as on u. A natural set of conditions [CN] is

$$\varphi(-\infty) = \varphi^\epsilon_\infty, \qquad \varphi(\infty) = \varphi^\epsilon_\infty, \tag{2.4}$$

$$u(-\infty) = u_{\text{cool}} + 1, \qquad u(\infty) = u_{\text{cool}}, \tag{2.5}$$

in terms of the moving coordinate $z := x - vt$ where v is to be determined. Here, the values $\varphi^\epsilon_{\pm\infty}$ must satisfy the compatibility condition (1.8).

In terms of the moving coordinates, the phase field equations have the form

$$\epsilon^2\varphi_{zz} = \alpha\epsilon^2 v\varphi_z + \frac{1}{2}(\varphi - \varphi^3) + \frac{\epsilon}{3d_0}u = 0, \tag{2.6}$$

$$Du_{zz} + v\left(u_z + \frac{1}{2}\varphi_z\right) = 0. \tag{2.7}$$

It was proven in [CN] that there exists a unique solution of (2.6), (2.7) subject to (2.4), (2.5) for small ϵ, and the velocity v (which depends on ϵ) approaches v^* given by (2.2). The far-field temperature u_{cool} not only determines the velocity, but the other boundary conditions in (2.4), (2.5) as well [CN]. In the limit of $\epsilon \to 0$, the temperature u in (2.6), (2.7) converges to the temperature (2.1) of the sharp interface model uniformly with respect to the distance measure given by the moving coordinate z. Other analyses of traveling waves for phase field equations in other scaling limits have been performed in [Wi], [BS], and [LBT].

An important question that is not resolved by the theoretical results is the rate at which the velocity $v(\epsilon)$ converges to v^* as ϵ approaches zero. Stated differently, the physically realistic value of ϵ is on the order of Angstroms, whereas, in practical computations, one would need to make ϵ larger by orders of magnitude.

Hence our first goal is to determine if the computation of plane waves is still reliable in the above situation. If so, by taking values of ϵ, which are small but still much larger than the true physical value, the front velocities and stability properties should not differ significantly from the velocity of the sharp interface model.

In our experiments we set the parameters to be $D = 1.0, \alpha = 1.0, d_0 = 0.01$, and $u_\infty = -1.05$, which yield a theoretical velocity $v^* = 5.0$. We then examined the nature of the convergence to the exact solution of the sharp interface model by halving ϵ as shown in Table 1.

We perform the same computation with each value of ϵ, and we examine the relative error of the interface

$$\frac{x(\text{theoretical}) - x(\text{experimental})}{x(\text{theoretical})}$$

at a characteristic point ($t = 0.0051$). Note that while ϵ is cut in half between experiments 1 and 2, the relative error in the interface position is reduced to one-fourth. Similarly, the relative error in velocity is reduced from 0.543 for $\epsilon = 0.00495$ to 0.093 for $\epsilon = 0.00247$. Between experiments 2 and 3, the relative error in the interface is similarly reduced. Since the approximations used in determining velocity are less accurate than those of the interface, there appears to be a finite limit to the relative error in velocity.

TABLE 1
Relative error with different values of ϵ.

Experiment	ϵ	Relative error in interface at $t = 0.00551$	Relative error in velocity at $t = 0.0051$	Maximum relative error in velocity
1	0.00495	0.162	0.543	0.911
2	0.00247	0.048	0.093	0.353
3	0.00124	0.015	0.081	0.135
4	0.00062	0.005	0.033	0.037

In earlier work [CS1], [CS2], we studied plane fronts in the distinguished limit of the classical Stefan model (there are no traveling waves in this case), using a one-dimensional formulation. The significant difference between one- and two-dimensional computations necessitates the testing of plane fronts and traveling waves in the actual two spatial dimensions. In our $2d$ experiments, we reproduce the exact classical Stefan results with a relative error of 0.0008 with 200 mesh points and $\epsilon = \triangle x$. The two sets of computations ($1d$ and $2d$) agree to the first 12 digits, thereby confirming the absence of extraneous grid effects in $2d$.

Hence, our basic conclusions for the $1d$ traveling waves [CS1], [CS2] in the surface tension and kinetics limit remain valid within the $2d$ context. The overall φ profile is essentially identical with the $1d$ calculations. The boundary conditions $\varphi_{\pm}$ are determined by (2.4) in correspondence with (1.8). The φ-level curves are all parallel to the interface and φ is approximately given by the function $\tanh(\frac{x-vt}{2\epsilon})$ as indicated by the theoretical results [CN]. The transition layer, is thus $\mathcal{O}(\epsilon)$ as in the $1d$ case. More specifically, if we take $\varphi = \pm.9$ as the cutoff in the transition layer, then the transition layer has a width 6ϵ; for cutoff $\varphi = \pm.99$ it is 10ϵ. Of course, we know from the theoretical results that the order ϵ correction to φ is crucial in the selection of the interface velocity, so that a correct interface velocity is confirmation of φ profile to order ϵ .

It is also noteworthy that the numerical error does not cause shape instability of the planar front, unlike the perturbation discussed below. This is consistent with the phase field theory since the stability properties of (1.6), (1.7) are the same as those of (1.1)–(1.3) to leading order [J], and the latter can be expected to be stable for perturbations with wavelengths that are on the size of the mesh used. The precise conditions are given in [CCh] and [J].

Next, we consider a planar wave whose interface is perturbed by a sinusoidal function, so that the interface location is given by

$$x(y) = a + b \cos my.$$

The parameters and boundary conditions remain the same as in experiment 1 with the initial temperature set at $u_\infty + 1$ in the solid with exponential decay to the boundary in the x-direction.

In Fig. 2.1 we show snapshots of the time evolution of the interface at $t = 0.0$, 0.00367, 0.00826, 0.01286, 0.01745, 0.022041, and 0.03122. In this experiment, $m = 1$, $a = 0.21$, and $b = 0.1485$. It can be clearly observed that the initial perturbation evolves into a needle-type shape.

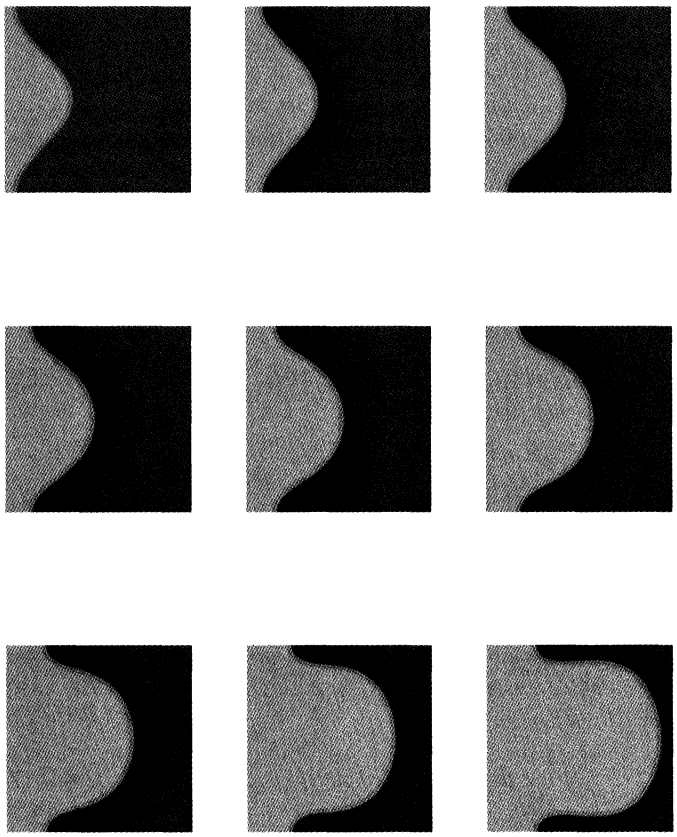


FIG. 2.1. *Single-needle crystal growth due to instability of a planar front. Initial interface at* $x = 0.21 + 0.1485 \cos y$. *Parameters are* $D = 1.0$, $\alpha = 1.0$, $d_0 = 0.01$, *and* $u_\infty = -1.05$, *which yield a Stefan planar front velocity* $v^* = 5.0$.

Hence the development of instabilities where one expects them on physical and theoretical grounds, and the absence of extraneous instabilities due to numerical error, indicate the suitability of using phase field equations to study interfaces.

It was noted in [CS1], [CS2] that there are questions of optimization in choosing ϵ with respect to the mesh spacing Δx. For a fixed grid, practical constraints impose a lower bound on Δx. For a given Δx, the question of the optimal choice of ϵ is more subtle. If ϵ is too small compared to Δx, then the surface tension, which is proportional to $\int (\nabla\varphi \cdot n)^2 dx$, will not be calculated accurately since there will be too few grid points in the interfacial region. This is easily illustrated with the tanh function that approximates φ, so that surface tension is essentially proportional to $\int \operatorname{sech}^2(r/2\epsilon) dr$. Recalling that the transition layer is about six to ten ϵ, depending on the precise cutoff point, a grid spacing of about $\Delta x = \epsilon$ would result in six to ten grid points on the interfacial region. Clearly, any fewer than six points would mean that the surface tension is not accurately computed. Since surface tension is crucial for the stability properties of the interface and this integral also has a key role in determining the $\mathcal{O}(\epsilon)$ terms in the solution, this is an important constraint.

Alternatively, if ϵ is too large, then (a) it diffuses the interface so that geometries with radii of curvatures smaller than ϵ would not be captured, and (b) the $\mathcal{O}(\epsilon)$ dynamics becomes a more significant factor limiting the quantitative accuracy of the interface velocity, as indicated by the asymptotic analysis.

In the two-dimensional problems we found the optimal value of the ratio $\frac{\epsilon}{\Delta x}$ to be $0.75 \leq \frac{\epsilon}{\Delta x} \leq 1.1$, as in the $1d$ problems. This is not very surprising in view of the asymptotic analysis, in which the size of ϵ is significant primarily in the direction perpendicular to the interface.

For the remainder of this paper we will use the ratio $\epsilon/\Delta x = 1$. Performing the same empirical optimization in some of the low symmetry examples implies a similar range for $\epsilon/\Delta x$.

The computation in Figs. 2.1, 3.1, 4.1, 5.1, 6.2, and 6.3 were done using a CRAY YMP with 200 meshpoints per side. Computations performed with 400 meshpoints were very similar, though much more costly. We use the same type of explicit schemes as in [CS1] and [CS2] and similar timescales. The timestep in all cases is the largest allowed by numerical stability considerations. The numerical runs continued until complete solidification or melting.

3. The critical radius and unstable equilibrium. A well-known instability in materials science is the critical radius [Ch, p. 67] in which a solid sphere with the sum of principal curvatures equal to κ_0 is in equilibrium with its melt at temperature u_0 with

$$u_0 = -d_0\kappa_0. \tag{3.1}$$

Under suitable conditions, e.g., Dirichlet boundary conditions, and ℓ, K^{-1} sufficiently small, a perturbation of the sphere to a larger one results in complete solidification and, analogously, for melting (see [CS2] for more discussion). The numerical computations assuming spherical symmetry confirmed this unstable equilibrium and showed that a small change in κ or σ produces a dramatic difference (melting to freezing or vice versa), while a large change in ϵ only produces a small change in velocity.

We study this problem now in the fully two-dimensional geometry (without imposing spherical symmetry), and we find that the spherical geometry is preserved. The values of K, ℓ, etc. are identical to those reported in [CS2], and again there is a 12-digit agreement between the one-dimensional and two-dimensional results for the location of the interface. Again, this confirms the absence of artificial grid effects.

Thus the two-dimensional calculations confirm the existence of this physical instability without causing artificial numerical instabilities nor necessitating the use of explicit equations of motion for the interface. Furthermore, in Fig. 3.1 we illustrate the results obtained when the initial interface is the perturbed circle $r = 0.3(1 + \frac{1}{3}\cos 8\theta)$. The parameters correspond to a critical radius $r_0 = 0.01732$, and, as predicted by (1.5), the interface evolves to a spherical shape and continues to freeze.

The introduction of anisotropy with respect to a fixed angle in the surface tension changes (3.1) into

$$\kappa_0 = -[s]_E u_0/(\sigma(\theta) + \sigma''(\theta)). \tag{3.2}$$

This expression for the curvature implies that the equilibrium shape is a dilation of the Wulff region. Figure 3.2 depicts the Wulff crystal for pivalic acid. This will be discussed in greater detail within the context of crystal growth in §5.

The $2d$ computations on the critical radius and the perturbed circle (Fig. 3.1) are also useful in demonstrating that grid effects do not introduce artificial anisotropy. In particular, the grid effects would be most pronounced for the critical radius problem since the initial configuration is borderline between melting and freezing, so that any extraneous effects would be magnified.

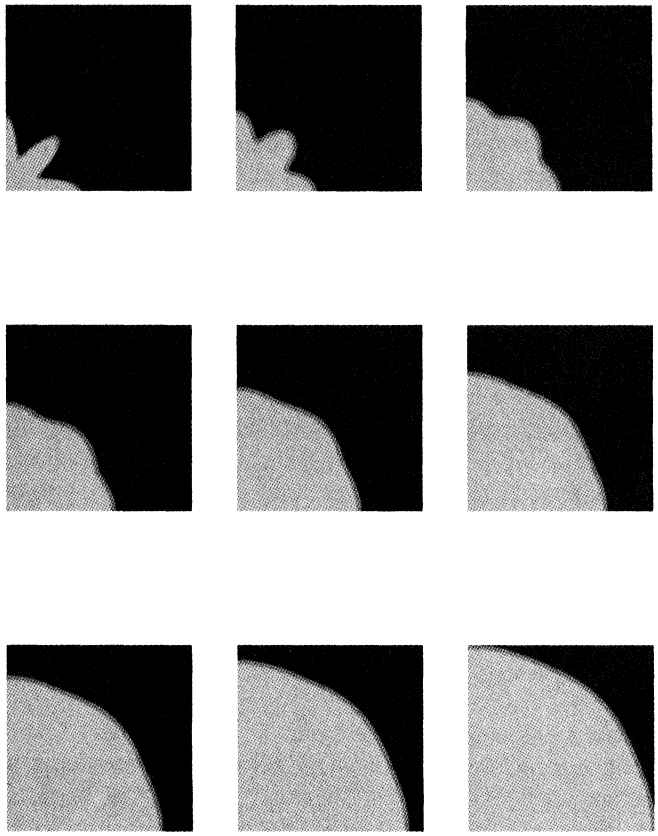


FIG. 3.1. *Stability of perturbations in the case of unstable equilibrium at critical radius. Initial interface at* $r = 0.3(1 + \frac{1}{3}\cos 8\theta)$. *Parameters are* $\alpha = 1.0$, $D = 1.0$, $\ell = 0.01$, $[s]_E = 4$, $\epsilon = 0.005$, *and* $\sigma = 0.034651$, *with critical radius* $r_0 = 0.01732$.

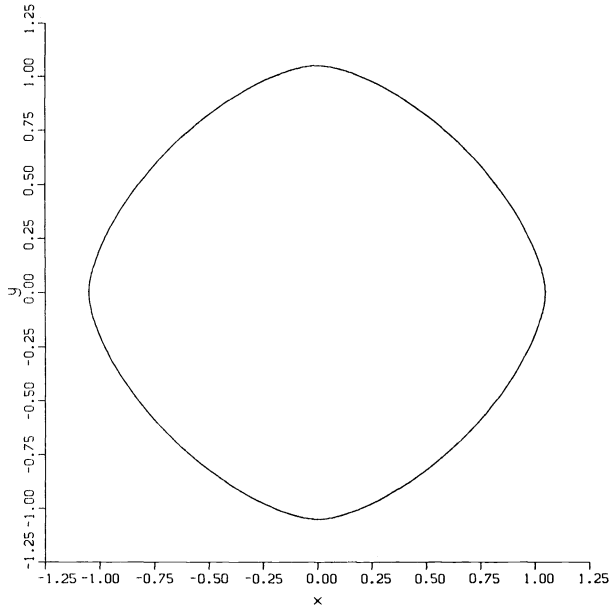


FIG. 3.2. *Wulff crystal for pivalic acid.*

4. The single-needle crystal. In §§4–6 we use the phase field equations to study a spectrum of problems that involve different physical parameters and behavior. One set of numerical experiments (see Figs. 3.1, 4.1, 5.1, and 6.3) are performed with identical initial conditions to illustrate the wide range of applicability.

The first of these is the single-needle crystal solidification at high velocities. Anisotropy is generally regarded as important for this problem as both σ and α vary with respect to orientation angle, θ. A series of physical experiments [SG], [GS] indicate that the role of anisotropy is to select the direction of the growth rather than to determine the shape or velocity of the tip. (However, other measurements, with samples that are not as pure as those used by [GS] suggest that the anistropy of these materials is much closer.) For example, the extent of anisotropy in the surface tension is ten times greater for pivalic acid than succinonitrile, yet the shape and dimensionless velocity (as a function of undercooling) are essentially identical [GS]. In earlier work [CL], the anisotropy in σ has been studied. In these numerical experiments, we study the effects of dynamic anisotropy, which is not as well understood in terms of the physical experiments. As a precise physical form for dynamical anisotropy has not been established, we use a (phenomenological) function that is strongly anisotropic along the diagonal. In particular,

$$\alpha^{-1} = \begin{cases} \alpha_0 + (1-\alpha_0)\left(\cos \frac{4\pi}{\sqrt{2}} d\right)^4 & \text{if } \frac{-\sqrt{2}}{8} < d < \frac{\sqrt{2}}{8}, \\ \alpha_0 & \text{otherwise,} \end{cases} \tag{4.1}$$

where d is the distance to the diagonals (between $\hat{x}$ and $\hat{y}$ axes). Note that the dynamic anisotropy will not generally have a form such as (1.5′), which arises from the second derivatives in the Laplacian.

In a part of the interface in which the temperature and curvature do not vary significantly, the velocity is given by

$$v \cong -\alpha^{-1}(\text{const}), \tag{4.2}$$

so that the diagonal is the preferred direction of growth. Figure 4.1 displays a time sequence in which the initial condition is a circle that is strongly perturbed by a sine function with values of $\alpha_0 = 0.1$. The growth is chiefly in the preferred direction. The diagonal was chosen as the preferred direction to ensure that anisotropic growth would not be the result of grid effects. In fact, the absence of anisotropic growth when α^{-1} is isotropic indicates that grid effects are not significant.

There is considerable experimental [SG], [GS], [DG] and theoretical evidence [BP], [KKL], [L] that suggests that the velocity of the single needle should be constant. This is confirmed by our numerical experiments, even though only linear interpolation is used to determine both the interface location and its velocity.

Furthermore, this velocity is found to be independent of the initial conditions, which is consistent with the experimental and theoretical results cited above. Figure 4.2 depicts the convergence to a selected velocity from distinct initial conditions. The velocity 44.13 is constant to within 0.62%.

5. Faceted crystal growth. The geometric patterns and facets that arise from crystal growth have posed intriguing questions for many years. A very early scheme to describe the equilibrium interface is known as the Wulff construction [Wu], which will be described later in this section. The propagation of facets and the development and elimination of corners has been studied extensively in recent decades from the minimal surfaces and related approaches [Ta].

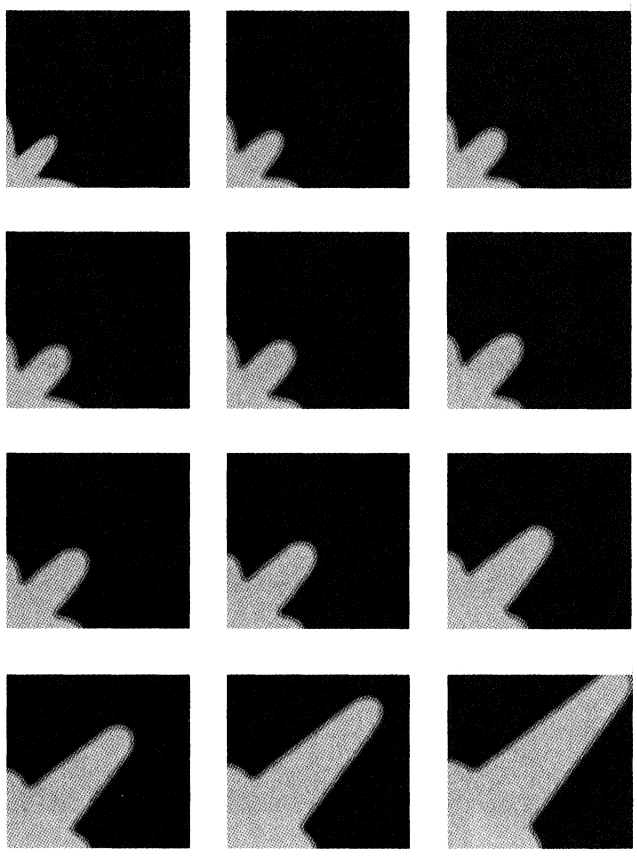


FIG. 4.1. *Single-needle crystal growth due to anisotropic mobility* (4.1) *with* $\alpha_0 = 0.1$. *Initial interface at* $r = 0.3(1 + \frac{1}{3}\cos 8\theta)$. *Parameters are* $D = 1.0$, $\ell = 0.01$, $[s]_E = 4$, $\epsilon = 0.005$, *and* $\sigma = 0.034641$.

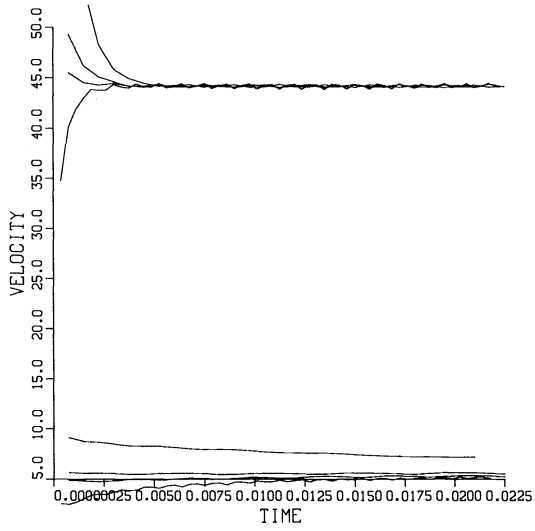


FIG. 4.2. *Tip and axis velocities of needle crystals obtained with different initial interfaces. Needle crystal tip velocity convergence to a constant independent of initial interface. Initial interfaces at* $r = 0.3$, $r = 0.3(1 + \frac{1}{3}\cos m(\theta - \frac{\Pi}{4}))$ *and* $r = 0.3(1 + \frac{1}{3}\cos m\theta)$ *for* $m = 4, 8$. *Parameters are* $D = 1.0$, $\ell = 0.01$, $[s]_E = 4$, $\epsilon = 0.005$, *and* $\sigma = 0.034641$.

In such studies, the corners are explicitly part of the numerical scheme. The evolution of faceted crystal growth poses a particularly strong challenge for a smooth set of parabolic equations, such as the phase field. Given a smooth initial data (e.g., first frame of Fig. 5.1), a material with anisotropy must eventually develop corners and facets and must propagate along planar fronts dictated by the geometry and anisotropy.

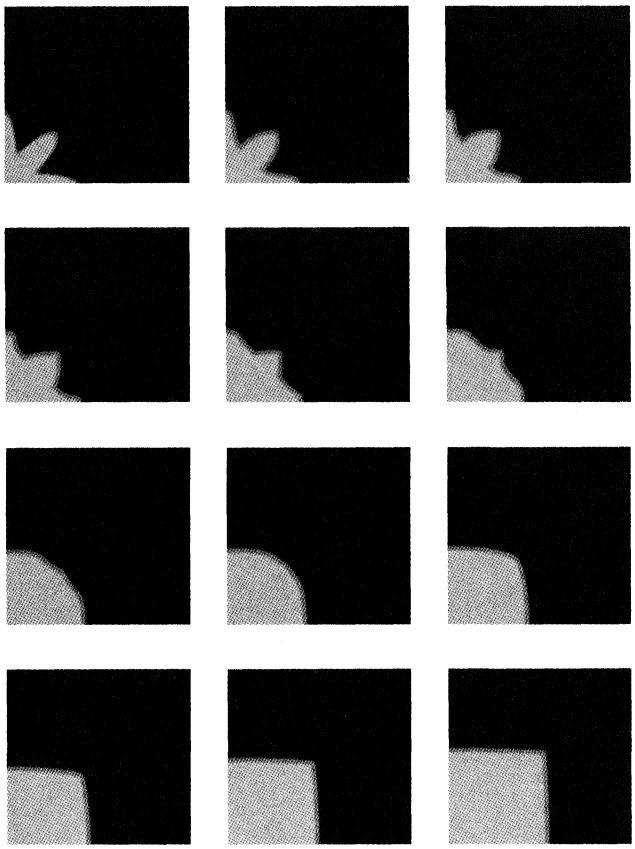


FIG. 5.1. *Faceted crystal growth due anisotropic mobility* $[\alpha^{-1} = .065(1 + 0.818 \cos 4(\theta - \frac{\pi}{4}))]$ *is the angle between interface normal and x-axis. Initial interface at* $r = 0.3(1 + \frac{1}{3} \cos 8\theta)$. *Parameters are* $D = 1$, $\ell = 0.01$, $[s]_E = 4$, $\epsilon = 0.005$, *and* $\sigma = .03464$.

The anisotropy in the phase field equations can involve both $\alpha(\theta)$ and $\sigma(\theta)$, where θ is now defined to be the angle between the normal to the interface and the x-axis, i.e.,

$$\cos\,\theta = \frac{\nabla\bar{\varphi}}{|\nabla\varphi|} \cdot \hat{x}. \tag{5.1}$$

We study the anisotropy by assuming σ and/or α are of the form

$$\sigma(\theta) = \sigma_0[1 + \delta_\sigma \cos\, M(\theta - \theta_\sigma)], \tag{5.2}$$

$$\alpha^{-1}(\theta) = \alpha_0^{-1}[1 + \delta_\alpha \cos\, M(\theta - \theta_\alpha)], \tag{5.3}$$

where M is an integer ($M = 4$ for cubic anisotropy), $\delta_\sigma, \delta_\alpha \in \mathbb{R}^+$ are the amplitudes of the anisotropies, and $\theta_\sigma, \theta_\alpha$ are the "preferred" angles. The latter will be defined more precisely below.

The phase field equations we study are then (1.6′), (1.7′) with d_0 and α given by (1.5′), (5.2), (5.3). In the asymptotic limit as ϵ approaches zero, the solutions to these phase field

equations are governed to leading order by those of (1.1′), (1.2′), (1.5), (1.5′). Since these dimensionless equations are related to the fully dimensional equations by simple substitution, every parameter in our system including those involving anisotropy, e.g., δ_σ, can be obtained directly from experimental data. Of course, we wish to "stretch out" the parameter ϵ, as discussed earlier, for computational convenience.

Next, we consider the issue of preferred direction, which is similar for the phase field and sharp interface models, in view of the remarks above. Consider the sharp interface problem (1.1′), (1.2′), (1.5), (1.5′) in equilibrium so that temperature is constant. Then (1.5), (1.5′) implies that (for some constant C^2)

$$\kappa = \frac{C^2}{\sigma(\theta) + \sigma''(\theta)}, \tag{5.4}$$

$$\sigma(\theta) + \sigma''(\theta) = \sigma_0[1 - (M^2 - 1)\delta_\sigma \cos M(\theta - \theta_\sigma)]. \tag{5.5}$$

Hence, if we assume that δ_σ satisfies $(M^2 - 1)\delta_\sigma < 1$, then the curvature has its maximum when $\theta = \theta_\sigma$. For cubic anisotropy ($M = 4$) with $\theta_\sigma := \frac{\pi}{4}$ this means that the equilibrium shape of the interface is such that the solid protrudes further in the diagonal directions.

This brief illustration is one implication of the Wulff construction [Wu] and can be obtained from the Cahn–Hoffman [CH] $\overline{\xi}$ vector approach or the Herring spheres approach. Note that when $\sigma + \sigma''$ changes sign, then one has facets in the equilibrium shape as a consequence of missing angles or orientations. The values of δ_σ for two materials used in dendritic experiments are 0.05 for pivalic acid and 0.005 for succinonitrile [GS].

The dynamical situation is more complicated even if the kinetic term αd_0 in (1.6) is neglected completely. On the other hand, in a limiting case in which curvature is neglected and $\alpha \neq 0$, one can acquire a basic understanding of the effects of this kinetic term. This analysis amounts to a comparison of velocities of planar fronts with different orientations, so that (1.5), (1.5′) imply

$$v \simeq \frac{C^2}{\alpha(\theta)[\sigma(\theta) + \sigma''(\theta)]}. \tag{5.6}$$

Once again, for small δ_σ and δ_α, and $\theta_\sigma = \theta_\alpha$, the velocity is maximum at $\theta = \theta_\sigma = \theta_\alpha$. Hence in the cubic case ($M = 4$, $\theta_\sigma = \theta_\alpha = \frac{\pi}{4}$), the growth is fastest in the diagonal directions.

These two calculations illustrate the following expectations. An initially spherical solid with cubic anisotropy will tend to grow fastest in the diagonal directions. If the boundary conditions are such that equilibrium is reached (e.g., appropriate Neumann conditions), then the interface will correspond to the Wulff shape, which is shown in Fig. 3.2, i.e., a square with rounded corners. Note that while the equilibrium result (5.4) is exact, the dynamic result (5.6) is only suggestive since the temperature will not be exactly constant and curvature is not precisely zero. Furthermore, one has a natural expectation that the dynamics will be less stable than equilibrium, thereby exaggerating growth in the favored directions. Thus it is possible that the dynamical problem will have facets for smaller anisotropies than those required by the vanishing of the denominator of (5.6), which would be the condition for "corners" or missing angles in equilibrium according to the Wulff construction.

While we consider the usual case in which $\theta_\sigma = \theta_\alpha$, it is worth noting that some materials exhibit dynamical anisotropy with a different preferred direction than that of the equilibrium.

In our numerical computations, we study a material with anisotropy given by (5.2) or (5.3) (with $\delta_\sigma = 0.05$ or $\delta_\alpha = .818$ and $M = 4$ and $\theta_\sigma = \frac{\pi}{4}$ or $\theta_\alpha = \frac{\pi}{4}$.). Note that the dynamic anisotropy appears to be larger, but the coefficient of δ_σ is multiplied by $M^2 - 1 = 15$. Our

value of δ_σ corresponds exactly to that reported by [GS] for pivalic acid. The computations show that a material with such anisotropy will evolve into a shape with four-fold symmetry, regardless of the initial pattern. In particular, Fig. 5.1 displays 12 frames in the phase field computations (1.6′), (1.7′) using (5.3) (i.e., with $d_0(\theta)$ defined by (1.5′). Similar results are obtained with (5.2). The fact that such different sources of anisotropy (5.2) versus (5.3) result in very similar patterns is an important physical conclusion that is not obvious from the form of the equations. The initial shape is the same perturbed circle used in the formation of the single-needle dendrite. By frame 3 of Fig. 5.1, this interface develops some corners and facets, some of which correspond to the favored directions discussed above, while others form as an immediate consequence of the initial conditions. In frames 3–9, these latter facets are gradually eliminated at the expense of the dominant facets distinguished by (5.4) and (5.6). In the last few frames (i.e., 10–12) these dominant planes become extended until they meet at the $x - y$ diagonal with a slightly smoothed-out right angle. The solidification then continues without any change in shape.

A rigorous result [So] has established that an interface moving in accordance with (1.3′) alone, with constant u, would retain the anisotropic shape. While there are no rigorous results for the full set of equations involved in solidification, our numerics show that this is indeed the case for the phase field equations. Since these equations are asymptotically governed by (1.1′)–(1.3′) to leading order, the same conclusion remains valid for the sharp interface problem.

6. Motion by mean curvature. The problem of motion by mean curvature (see [Br] and references contained therein) is to determine a $(d-1)$-dimensional surface (or curve) $\Gamma_s(t)$ in $\Omega \subset \mathbb{R}^d$, such that the (normal) velocity, $v(t,x)$, at each point on the surface satisfies

$$v(t,x) = -\alpha^{-1}\kappa(t,x), \tag{6.1}$$

where κ is again the sum of principal curvatures and α^{-1} is a constant or a function of orientation angle.

A closely related problem is to find the level set

$$\Gamma(t) := \{x \in \Omega : \varphi(t,x) = 0\}, \tag{6.2}$$

where φ is a solution to the single parabolic equation

$$\alpha\epsilon^2\varphi_t = \epsilon^2\Delta\varphi + \frac{1}{2}(\varphi - \varphi^3). \tag{6.3}$$

Equation (6.3), which was introduced by Cahn and Allen [Ca], [AC] in the context of antiphase boundaries is a limiting case of the phase field equations. As the latent heat ℓ approaches zero, while the initial and boundary conditions imposed on u are also zero, the phase field equations (1.6′), (1.7′) have solutions that are governed by those of (6.3) and consequently (6.1) to leading order. Thus the leading order asymptotics for (1.6′), (1.7′) is just a special case of (1.5) in which $u = 0$ [Ca2]. In recent years, the connections with materials science through the phase field equations have stimulated a resurgence of interest in the mathematical theory of (6.3) and other regularizations of (6.1) [BK], [MSc], [GG]. From a formal analysis it is clear that an initial interface of the form of frame 1 in Fig. 6.3 will tend to approach a circle and then shrink until extinction. The time of extinction from a circle is easily calculated.

Our numerical calculations consist of studying (1.6), (1.7) in the limit of small ℓ, with zero boundary and initial conditions for u, and the standard conditions of φ described in (2.10) so that $\varphi_+ = +1$ on the (exterior) liquid and $\varphi_- = -1$ in the (interior) solid in this case. In our first experiment we compared the numerical results with a theoretical solution. We took $\alpha = 1$, $\ell = 0$, zero boundary and initial conditions for u and the initial solid seed to be a circle of radius 0.3. The theoretical solution gives a circle with radius $r(t) = (0.09 - 2t)^{1/2}$, which has an extinction time $T_0 = 0.045$.

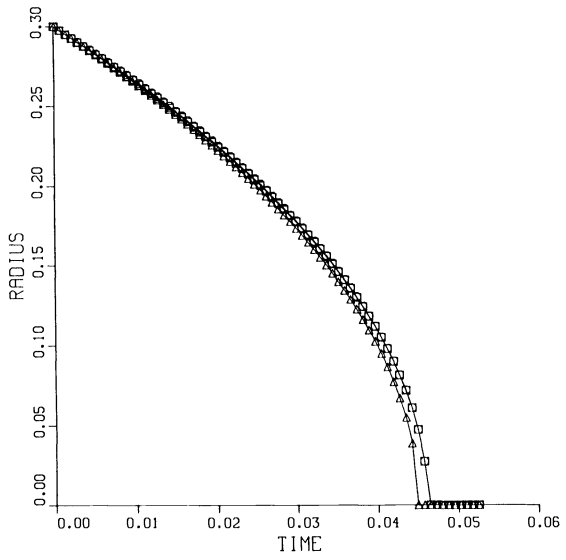


FIG. 6.1. *Motion by mean curvature. Computed and theoretical interface locations for a circular initial interface. Exact radius is* $r(t) = (0.09 - 2t)^{1/2}$, $0 \le t \le 0.045$. *Parameters are* $D = 1.0$, $\ell = 0$, $[s]_E = 4$, $\epsilon = 0.005$, *and* $\sigma = 0.866025$.

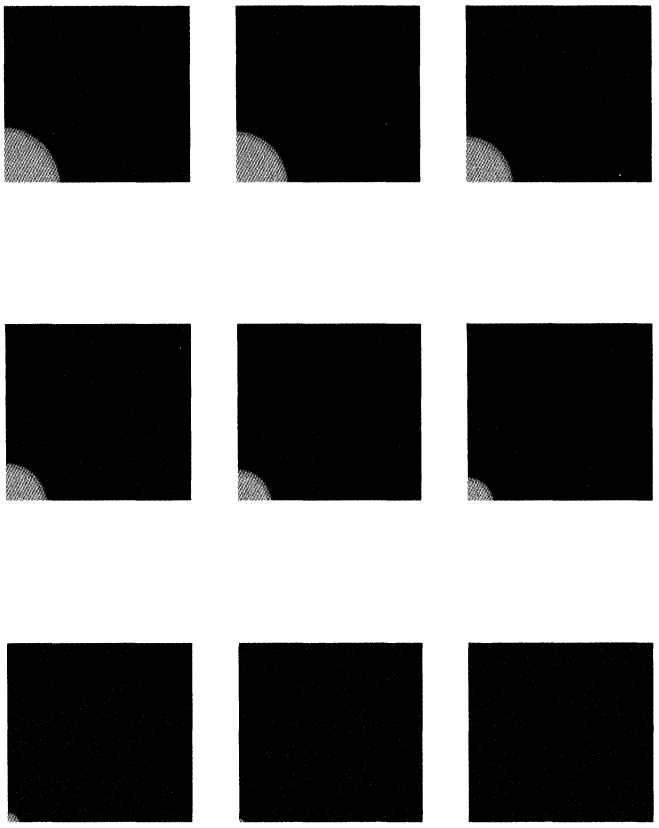


FIG. 6.2. *Motion by mean curvature. Crystal evolution corresponding to the computation of Fig.* 6.1.

In Fig. 6.1 we depict the computed and theoretical radius of the circle as a function of time. There is an agreement of two or more digits up to $t = 0.0345$ and the accuracy deteriorates approaching T_0 as expected. The computational extinction time is between 0.04575 and 0.0465. Figure 6.2 depicts the evolution at times

$$t = 0.0, 0.00675, 0.01425, 0.02175, 0.02925, 0.03675,$$
$$0.04425, 0.04565, 0.0465.$$

The initial condition in frame 1 of Fig. 6.3 is identical to those that we used for faceted crystals and stable solidification. In frames 1–4, the interface moves toward a circle (frame 5) with a radius that is approximately that of the perturbed circle in frame 1. The interface acts in a way to minimize the surface tension during this phase. The surface tension in (6.3) or (1.7) is determined to leading order by the coefficients of $\Delta\varphi$ and the $\frac{1}{2}(\varphi - \varphi^3)$ term.

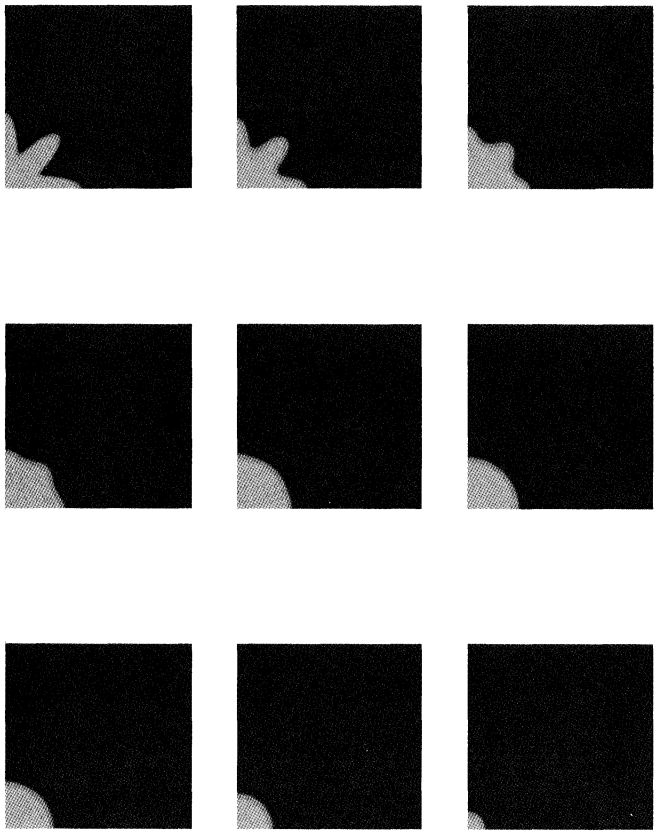


FIG. 6.3. *Motion by mean curvature. Crystal evolution for initial interface* $r = 0.3(1 + \frac{1}{3}\cos 8\theta)$. *Parameters are the same as in Figs.* 6.1 *and* 6.2.

The anisotropic analog of (6.1) involves replacing the constant α by a function of orientation angle as discussed earlier in the single-needle or faceted-crystal growth cases. Note that the anisotropy involved in the surface tension is not relevant in this limit since σ cancels in the asymptotics. In the anisotropic case the evolution of the interface is similar except that the convergence of the interface is toward the appropriate Wulff shape rather than the sphere.

Acknowledgments. The authors thank the Pittsburgh Supercomputing Center for computing time and use of facilities.

REFERENCES

[AC] S. ALLEN AND J. CAHN, *A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening*, Acta Metall., 27 (1972), pp. 1084–1095.

[BK] L. BRONSARD AND R. KOHN, *Motion by mean curvature as the singular limit of Ginzburg–Landau dynamics*, J. Differential Equations, 90 (1991), pp. 211–237.

[BP] M. BEN-AMAR AND Y. POMEAU, *Theory of dendritic growth in a weakly undercooled melt*, Europhys. Lett., 2 (1986), pp. 307–314.

[Br] K. A. BRAKKE, *The motion of a surface by its mean curvature*, Math Notes, Princeton University Press, Princeton, NJ, 1978.

[BS] M. BARBER AND D. SINGLETON, *Travelling waves in phase field models of solidification*, Australian National Univ., preprint, 1992.

[Ca] G. CAGINALP, *The limiting behavior of a free boundary in the phase field model*, Carnegie-Mellon Research Report 82-5, 1982.

[Ca1] ———, *Mathematical models of phase boundaries*, in Material Instabilities in Continuum Problems and Related Mathematical Problems, J. Ball, ed., Heriot-Watt Symposium, 1985-1986, Oxford Science Publications, England, 1988, pp. 35–52.

[Ca2] ———, *Stefan and Hele-Shaw type models as asymptotic limits of the phase field equations*, Phys. Rev. A, 39 (1989), pp. 5887–5896.

[CC] G. CAGINALP AND X. CHEN, *Phase field equations in the singular limit of sharp interface problems*, IMA Volumes in Mathematics and Its Applications, 43 (1990), pp. 1–28.

[CCh] G. CAGINALP AND J. CHADAM, *Stability of interfaces with velocity correction term*, Rocky Mtn. J. of Math., 21 (1991), pp. 617–629.

[Ch] B. CHALMERS, *Principles of Solidification*, R. E. Krieger Publishing Co., Melbourne, FL, 1964.

[CH] J. CAHN AND D. HOFFMAN, *A vector thermodynamics for anisotropic surfaces* II. *Curved and faceted surfaces*, Acta. Met., 22 (1974), pp. 1205–1214.

[CHi] J. W. CAHN AND J. E. HILLIARD, *Free energy of a nonuniform system*, I. *Interfacial free energy*, J. Chem. Phys., 28 (1957), pp. 258–267.

[CL] G. CAGINALP AND J. LIN, *A numerical analysis of an anisotropic phase field model*, IMA J. Appl. Math., 39 (1987), pp. 51–66.

[CN] G. CAGINALP AND Y. NISHIURA, *The existence of travelling waves for phase field equations and convergence to sharp interface models in the singular limit*, Quart. Appl. Math., 49 (1991), pp. 147–162.

[Cr] J. CRANK, *Free and Moving Problems*, Oxford University Press, New York, 1984.

[CS1] G. CAGINALP AND E. SOCOLOVSKY, *Efficient computation of a sharp interface by spreading via phase field methods*, Appl. Math. Lett., 1989, pp. 117–120.

[CS2] ———, *Computation of sharp phase boundaries by spreading the planar and spherically symmetric cases*, J. Comput. Phys., 95 (1991), p. 85–100.

[DG] A. DOUGHERTY AND J. GOLLUB, *Steady state dendritic growth of N_4 B R from solution*, Phys. Rev. A, 38 (1988), pp. 3043–3053.

[DHOX] J. DEWYNNE, S. HOWISON, J. OCKENDON, AND W. XIE, *Asymptotic behavior of solutions to the Stefan problem with a kinetic condition at the free boundary*, J. Australian Math. Soc. Ser. B, 31 (1989), pp. 81–96.

[F] G. FIX, *Free Boundary Problems*, A. Fasano and M. Primicerio, eds., Pitman, Boston, 1983, p. 580.

[GG] Y. GIGA AND S. GOTO, *Geometric evolution of phase boundaries*, IMA Preprint 738, 1990.

[Gl] J. GLIMM, *Lecture notes in physics*, 344 (1989), pp. 177–186.

[GS] M. E. GLICKSMAN AND N. B. SINGH, *Effects of crystal-melt interfacial energy anisotropy on dendritic morphology and growth kinetics*, J. Crystal Growth, 98 (1989), pp. 277–284.

[HH] P. C. HOHENBERG AND B. I. HALPERIN, *Theory of dynamics critical phenomena*, Rev. Modern Phys., 49 (1977), pp. 435–489.

[J] J. JONES, Univ. of Pittsburgh, Pittsburgh, PA, manuscript.

[K] R. KOBAYASHI, *Videotape and Preprint*, Ryokoku Univ., Ryokoku, Japan, 1991.

[KKL] D. KESSLER, J. KOPLIK, AND H. LEVINE, *Pattern selection in fingered growth phenomenon*, Adv. Phys., 37 (1988), pp. 255–339.

[L] J. S. LANGER, *Lectures in the Theory of Pattern Formation*, Chance and Matter, J. Souletie, ed., North Holland, Amsterdam, 1987.

[La] ———, *Theory of the condensation point*, Ann. Phys., 41 (1967), pp. 108–157.

[LBT] H. LOWEN, J. BECHHOEFER, AND L. TUCKERMAN, *Crystal growth at long times*, Simon Fraser Univ., Burnaby, B.C., preprint.

[LF] J. LIN AND G. FIX, *Numerical simulations of nonlinear phase transitions* I. *The isotropic case*, Nonlinear Anal. Theory, Math. Appl., 12 (1988). pp. 811–823.

[LL] J. LANDAU AND E. M. LIFSHITZ, *Statistical Physics*, Addison–Wesley, Reading, MA, 1958.

[MS] W. W. MULLINS AND R. F. SEKERKA, *Morphological stability of a particle growing by diffusion or heat flow*, J. Appl. Phys., 34 (1963), pp. 323–329.

[MSc] P. DE MOTTONI AND M. SCHATZMAN, *Evolution gèometrique d'interfaces*, C.R.A.S., 309 (1989), pp. 453–458.

[Oc] J. OCKENDON, *Linear and nonlinear stability of a class of moving boundary problems*, Proc. Pavia Conference, Free Boundary Problems, E. Magenes, ed., Instituto Nazional di Alta Matematica, Francesco Severi, Rome, 1980.

[PT] R. PEYRET AND T. TAYLOR, *Computational Methods in Fluid Flow*, Springer-Verlag, Berlin, 1983.

[RT] A. ROOSEN AND J. E. TAYLOR, *Simulation of crystal growth with faceted interfaces*, preprint, 1992.

[SG] N. B. SINGH AND M. E. GLICKSMAN, *Free dendritic growth in viscous melts: cyclohexanol*, J. Crystal Growth, 82 (1989), pp. 534–540.

[So] H. M. SONER, *Motion of a set by the curvature of its boundary*, preprint, 1991.

[SS] J. A. SETHIAN AND J. STRAIN, *Crystal growth and dendritic solidification*, Report PAM-509 Berkeley, Univ. of California, Berkeley, 1990.

[Ta] J. E. TAYLOR, *Motion of curves by crystalline curvature including triple junctions and boundary point*, preprint, 1991.

[Wi] J. WILDER, Univ. of West Virginia, Morgantown, preprint, 1990.

[Wu] G. WULFF, *Krystallograph*, Mineral, 34 (1901), p. 499.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 127–138, January 1994

THREE-DIMENSIONAL ADAPTIVE MESH REFINEMENT FOR HYPERBOLIC CONSERVATION LAWS*

JOHN BELL†, MARSHA BERGER‡, JEFF SALTZMAN§, AND MIKE WELCOME†

Abstract. A local adaptive mesh refinement algorithm for solving hyperbolic systems of conservation laws in three space dimensions is described. The method is based on the use of local grid patches superimposed on a coarse grid to achieve sufficient resolution in the solution. A numerical example computing the interaction of a shock with a dense cloud in a supersonic inviscid regime is presented. Detailed timings are given to illustrate the performance of the method in three dimensions.

Key words. adaptive methods, mesh refinement

AMS subject classifications. 65M06, 65M50, 76N15

1. Introduction.

Advanced finite difference methods, by themselves, are unable to provide adequate resolution of three-dimensional phenomena without overwhelming currently available computer resources. High-resolution three-dimensional modeling requires algorithms that focus the computational effort where it is needed. In this paper we describe an extension of the adaptive mesh refinement (AMR) algorithm for hyperbolic conservation laws originally developed in [2], [3] to three space dimensions. AMR is based on a sequence of nested grids with finer and finer mesh spacing in both time and space. These fine grids are recursively embedded in coarser grids until the solution is sufficiently resolved. An error estimation procedure automatically estimates the accuracy of the solution, and grid generation procedures dynamically create or remove rectangular fine grid patches. Special difference equations are used at the interface between coarse and fine grid patches to insure conservation. This is all handled without user intervention by the AMR program.

Two-dimensional versions of the AMR algorithm described here have been used to solve fluid flow problems in a variety of settings and have enabled the study of fluid flow phenomena not previously possible. For example, the extra resolution provided by AMR enabled the computation of a Kelvin–Helmholtz type instability along the slip line in a study of Mach reflection off an oblique wedge [2] and aided in the resolution of the weak von Neumann paradox in shock reflection [7]. When combined with a multifluid capability, the algorithm was used to compute the interaction of a supernova remnant with an interstellar cloud [8] and to categorize refraction patterns when a shock hits an oblique material interface [13]. When extended to use body-fitted coordinates, AMR was used to study diffraction of a shock over an obstacle [11]. In each of these cases the use of adaptive mesh refinement reduced the cost of the computation by more than an order of magnitude. The improved efficiency associated with using AMR may make similar flows in three dimensions computationally tractable.

There are several alternative approaches to focusing computational effort in three-dimensional flows. One approach uses a logically rectangular grid with moving grid points that adjust to the flow [11], [9]. There are several difficulties with this approach. First, it is hard

*Received by the editors December 17, 1991; accepted for publication (in revised form) January 27, 1993.

†Lawrence Livermore Laboratory, Livermore, California 94550. This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory contract W-7405-Eng-48. Partial support under contract W-7405-Eng-48 was provided by the Applied Mathematical Sciences Program of the Office of Energy Research and by Defense Nuclear Agency contract IACRO 90-871.

‡Courant Institute, New York University, New York, New York 10012. The work of this author was supported in part by Air Force Office of Scientific Research grant 86-0148, Department of Energy grant DE-FG02-88ER25053, and by a PYI National Science Foundation grant ASC-8848101.

§Los Alamos National Laboratory, Los Alamos, New Mexico 87545. The work of this author was supported by the U.S. Department of Energy through Los Alamos National Laboratory contract W-7405-ENG-36.

to implement a three-dimensional high-resolution integration scheme for moving rectilinear grids. In our approach the integration scheme need only be defined for uniform rectangular grids; this avoids the complexity and computational cost associated with metric coefficients in the moving grid approach. Furthermore, in three dimensions it is extremely difficult to effectively cluster points to capture unsteady phenomena while maintaining a grid with sufficient smoothness in both space and time to permit effective computation. Even if acceptable grid motion can be determined, the entire computation is usually performed with a fixed number of gridpoints throughout the computation. In contrast, the local grid refinement approach dynamically adjusts the number of zones to match the requirements of the computation. The timestep used in moving mesh codes is also limited by the smallest cell size unless additional work is done by solving the equations implicitly or using techniques that allow each cell to evolve with its own timestep.

Another approach to three-dimensional computations uses adaptive unstructured grids [1], [14]. Unstructured grids offer the most flexibility in optimally placing the gridpoints. They can treat general geometries in a simple uniform way. They also offer a possibility of anisotropic refinement, as might be used for Navier–Stokes boundary layer calculations. However, we favor the use of locally uniform refined patches (the basis for AMR, described later) for their accuracy and wave propagation properties. The development of discretization techniques that avoid degradation for strong shocks on highly irregular meshes remains an open issue. Our use of uniform grids allows us to directly use much of the high resolution difference scheme methodology developed for this flow regime. Uniform patches also have low overhead, both from the computational and the storage point of view. The extra information that is needed, in addition to the actual solution values, is proportional to the number of grids rather than the total number of grid points.

An indication of the robustness of the mesh refinement algorithm is that very few changes were required in extending it from two to three dimensions. However, time-dependent three-dimensional computations push the limits of current machine resources, both in terms of memory and CPU time. For this reason, particular care was taken in the implementation, and a better grid generation algorithm was used to increase the overall efficiency of the code.

The starting point for this paper is the version of AMR presented in [2] and we assume the reader is familiar with that paper. In §2 we describe the differences in the three-dimensional mesh refinement algorithm which have to do with the grid generation algorithm and the error estimator. In §3 we describe the operator split integration scheme used in the numerical experiment. In particular, this section clarifies the interaction of the grid refinement and operator split boundary conditions on patched grids while maintaining conservation at grid interfaces. We include a brief section on implementation since some simple changes that produce much cleaner code have been incorporated. We are also rewriting the code in a hybrid of C++ and FORTRAN. The results of a numerical experiment of a shock-cloud interaction modeling the laboratory experiments of Haas and Sturtevant [12] are presented in §5. Detailed timings are presented as well as memory usage and grid statistics demonstrating that AMR offers significant savings of computational resources and can be an important tool in the study of three-dimensional fluid dynamics.

2. The adaptive mesh refinement algorithm. AMR uses a nested sequence of logically rectangular meshes to solve a partial differential equation (PDE). In this work, we assume the domain is a single rectangular parallelepiped although it may be decomposed into several coarse grids. With the new grid generator described below, grids at the same level of refinement do not overlap. We require that the discrete solution be independent of the particular decomposition of the domain into subgrids. Grids must be properly nested, i.e., a fine grid should be at least one cell away from the boundary of the next coarser level unless it is touching

the boundary of the physical domain. However, a fine grid can cross a coarser grid boundary and still be properly nested. In this case, the fine grid has more than one parent grid. This is illustrated in Fig. 1 in two dimensions. (This set of grids was created for a problem with initial conditions specifying a circular discontinuity.)

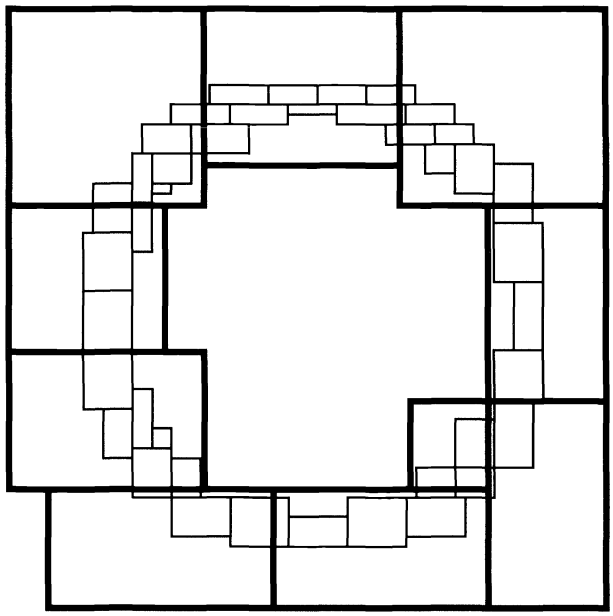


FIG. 1. *A coarse grid with two levels of refined grids. The grids are properly nested, but may have more than one parent grid.*

AMR contains five relatively separate components. The *error estimator* uses Richardson extrapolation to estimate the local truncation error; this determines where the solution accuracy is insufficient. The *grid generator* creates fine grid patches covering the regions needing refinement. *Data structure* routines manage the grid hierarchy allowing access to the individual grid patches as needed. *Interpolation* routines initialize a solution on a newly created fine grid and provide the boundary conditions for integrating the fine grids. *Flux correction* routines insure conservation at grid interfaces by modifying the coarse grid solution for coarse cells that are adjacent to a fine grid.

When all these components are assembled, a typical integration step proceeds as follows. The integration steps on different grids are interleaved so that before advancing a grid, all the finer level grids have been integrated to the same time. One coarse grid cycle is then the basic unit of the algorithm. The variable r denotes the mesh refinement factor in both space and time (typically 4), and $level$ refers to the number of refinements (the coarsest grid is at level 0). The regridding procedure is done every few steps, so any particular step may or may not

involve regridding.

```
Recursive Procedure Integrate (lev)
   Repeat r^lev times
      Regridding time? - error estimation and grid generation
                                 for level lev grids and finer
      step Δt_lev on all grids at level lev
      if (lev+1) grids exist
      then begin
              integrate (lev+1)
              conservation_fixup(lev, lev+1)
           end
   end.

lev = 0  (* coarsest grid level *)
Integrate (level)
```

Single Coarse Grid Integration Cycle

All of these steps are described fully in [2], with the exception of the new grid generation algorithm.

2.1. Grid generation. The grid generation algorithm takes a list of coarse grid points tagged as needing refinement and groups them in clusters. Fine grids are then defined by fitting the smallest possible rectangle around each cluster. The grid generator should produce efficient grids, i.e., rectangles containing a minimum number of cells that are not tagged, without creating a large number of small grids with poor vector performance and excessive boundary overhead.

The grid generator described in [2] uses a simple bisection algorithm. If a single enclosing rectangle is too inefficient, it is bisected in the long direction, the tagged points are sorted into their respective halves, and new enclosing rectangles are calculated. The efficiency is measured by taking the ratio of tagged points to all points in a new fine grid. The procedure is repeated recursively if any of the new rectangles is also inefficient. Since this algorithm uses no geometric information from the tagged points, it often results in too many tiny subgrids and is followed by a merging step. Unfortunately, this results in overlapping grids. Since the memory usage in three-dimensional calculations is at a premium, we want to avoid overlapping. Furthermore, we expect that there will be large numbers of grids in three dimensions which makes the merging step costly.

We now use a new clustering algorithm that uses a combination of signatures and edge detection. Both techniques are common in the computer vision and pattern recognition literature. After much experimentation, described in [4], we settled on what amounts to a "smart bisection" algorithm. Instead of cutting an inefficient rectangle in half, we look for an "edge" where a transition from a flagged point region to a nonflagged one occurs. The most prominent such transition represents a natural line with respect to which the original grid can be partitioned.

We describe the procedure in two dimensions for purposes of illustration. First, the signatures of the flagged points are computed in each direction. Given a continuous function $f(x, y)$, the horizontal and vertical signatures Σ_x and Σ_y are defined as

$$\Sigma_x = \int_y f(x, y)\, dy$$

and

$$\Sigma_y = \int_x f(x, y)\, dx,$$

respectively. For discrete binary images, this is just the sum of the number of tagged points in each row and column. If either signature contains a zero value, then clearly a rectangle can be partitioned into two separate clusters in the appropriate direction. If not, an edge is found by looking for a zero crossing in the second derivative of the signature. If there is more than one such zero crossing, the largest one determines the location for the partitioning of the rectangle. If two zero crossings are of equal strength, we use the one closest to the center of the old rectangle to prevent the formation of long thin rectangles with poor vectorization. This procedure is also applied recursively if the resulting rectangles do not meet the efficiency criterion, with the exception that if no good partition is found and the efficiency is at least 50%, the rectangle is accepted; otherwise, it is bisected in the long direction as a last resort. In computational experiments in two space dimensions, for the same level of accuracy the new algorithm reduces the CPU time by approximately 20%.

Figure 2 illustrates the entire procedure on a sample set of points. The first column on each side is the signature, and next to it is the second derivative. After each partitioning of the points, a new enclosing rectangle is calculated around the tagged points. In this figure, after three steps, the fine grids are acceptably efficient and the procedure stops.

2.2. Error estimation. The second improvement over the basic approach in [2] is the addition of a purely spatial component to the error estimation process to supplement Richardson extrapolation. In Richardson extrapolation, the data on the grid where the error is being estimated is coarsened and then integrated for a timestep. That result is then compared to the result of integrating first and then coarsening. For smooth solutions, the difference in these two results is proportional to the truncation error of the scheme. The motivation for including an additional error measure is to identify structures that are missed by the averaging process associated with the coarsening in the Richardson extrapolation. For example, in gas dynamics a slow-moving or stationary contact surface generates little or no error in the Richardson extrapolation process. In fact, the integrator is not generating any error in this case. However, failure to tag the contact will cause it to be smeared over a coarser grid. The error associated with deciding whether or not to refine the contact surface is associated only with the spatial resolution of the discontinuity, not with errors in integration. Should the contact need to be refined later (for example, if it interacts with another discontinuity), the initial conditions are no longer available to provide higher resolution. For certain special cases a similar phenomenon can also occur for shocks. These problems can be avoided by providing the error estimation routine with the unaveraged grid data so that spatial resolution can also be measured.

Along with the addition of a purely spatial component to the error estimation, we also directly control the process of tagging (and untagging) cells for refinement. For example, a user can insist that only a certain part of the domain is of interest and that the error estimator should be ignored if it says refinement is needed outside of the interesting regions. Similarly, it can force refinement in a particular region independent of the error estimation result.

3. Integration algorithm. For the computational examples presented in §5, we use an operator-split second-order Godunov integration scheme. However, the particular form of the integration scheme is independent of the remainder of the AMR shell. Other integration methods and, in fact, other hyperbolic systems can be easily inserted into the overall AMR framework. The only requirement for the integration scheme is that it be written in flux form, i.e.,

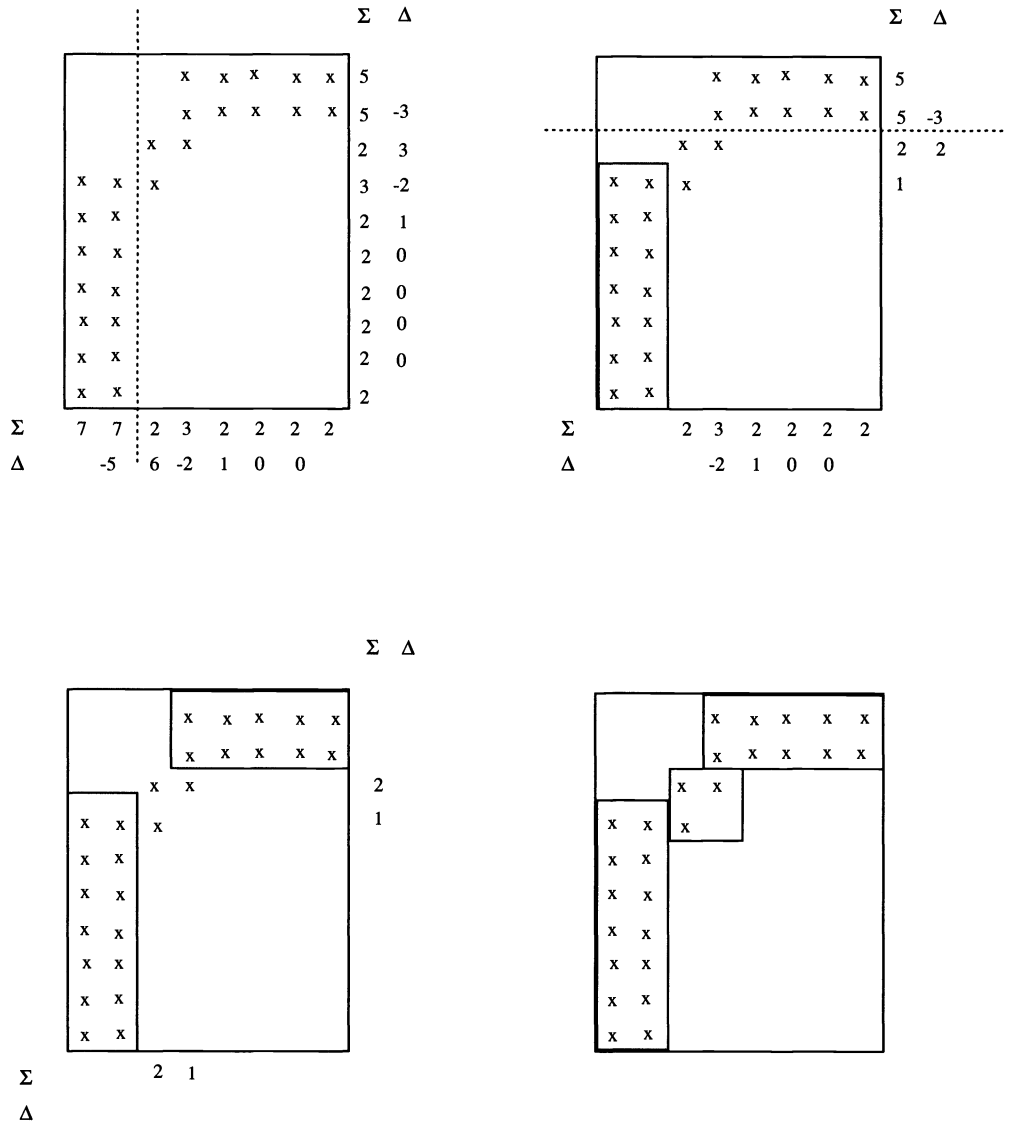


FIG. 2. *The signature arrays* Σ_x, Σ_y *and the second derivatives* Δ_x, Δ_y *used to partition the clusters.*

$$U_{ijk}^{n+1} = U_{ijk}^{n} - \Delta t \left(\frac{F_{i+1/2,j,k} - F_{i-1/2,j,k}}{\Delta x} + \frac{G_{i,j+1/2,k} - G_{i,j-1/2,k}}{\Delta y} + \frac{H_{i,j,k+1/2} - H_{i,j,k-1/2}}{\Delta z} \right), \tag{1}$$

where F, G, H are the numerical fluxes in the x, y, z directions, respectively. In its current form, these numerical fluxes are assumed to be explicitly computable from the values in cell ijk and a localized collection of its neighbors, as is typical of conventional explicit finite difference methods. When the integrator is invoked, it is provided with data on the grid to be integrated as well as sufficient boundary data (based on the scheme's stencil) to advance the solution on the given grid. No special stencils are used at fine/coarse interfaces. Instead, coarse

grid data is linearly interpolated to the fine grid resolution to provide a border of boundary cells, which is provided to the integrator along with the grid data itself. After the integration step, fluxes are adjusted to insure conservation at grid interfaces.

When operator splitting is used with local grid patches, the only thing to note is that extra boundary cells must be integrated during the first sweep to provide accurate boundary values for subsequent sweeps. For example, for a scheme with four points to the side in the stencil, four entire rows of dummy cells along the top and bottom of the grid must be advanced in the x sweep, so that four points are available for the y sweep at the next stage. For very small grids, this additional boundary work can dominate the computational cost of advancing a grid (particularly for difference methods having a broad stencil such as the Godunov algorithm we are using). This additional boundary work, as well as the vectorization issues, places a premium on generating large but efficient grids during regridding.

As an example, if one $60 \times 60 \times 60$ grid were replaced by two $30 \times 60 \times 60$ grids, both grids would redundantly integrate the overlapping boundary cells. This causes approximately 4% of the total computation to be redundant.

4. Implementation. Two simple decisions had a large impact on the implementation of AMR. The first concerned the organization and separation of the problem-dependent and problem-independent parts of the AMR code. The problem-dependent parts include the particular hyperbolic systems to be solved (and a suitable integration scheme), the initial and boundary conditions, the problem domain, and the error estimation routines. When a new problem is being set up, the changes required to the code are localized to a small number of subroutines. The integration subroutine advances the solution of the particular differential equations for a single timestep on a rectangular subgrid and returns fluxes that are required to insure conservation at coarse-fine boundaries. Consequently, adapting an existing integration module for use with the AMR algorithm is routine. The remainder of the AMR shell treats the data in terms of conserved variables where the number of variables is specified as a parameter. Thus, the data structures, memory management, grid generation algorithms, the logic controlling the timestepping and subcycling on subgrids, interior boundary conditions, and the interfacing between grids that insure conservation are completely divorced from the particular system being solved.

The second implementation detail that simplified the programming of AMR and resulted in much cleaner code than in [2] is the use of a global integer index space covering the domain. These integers are used in describing the location of the grids. Based on the initial (user-specified) domain, given in floating point numbers, an integer lattice based on the number of cells in each dimension (nx, ny, and nz) is determined. The domain may then be partitioned into several coarse grids, each located in subcubes with indices between (1, 1, 1) and (nx, ny, nz). If the refinement ratio is r between level l and level $l+1$, then the fine grid cells corresponding to coarse grid cell i, j, k are $r \cdot i + p_i, r \cdot j + p_j, r \cdot k + p_k$, where the $0 \le p_{i,j,k} \le r - 1$. This completely eliminates round-off error problems that would otherwise require careful coding to determine whether two grids overlap or whether a coarse grid is a parent to a fine grid.

We are currently rewriting the AMR algorithm in C++ with calls to FORTRAN routines for the numerically intensive parts. By constructing the appropriate classes we are able to define a grid calculus in which the computation of intersections and, in fact, the entire regridding process is greatly simplified. Using the built-in macro preprocessor, we are able to implement a large portion of the code in a dimension-independent manner with the dimension as a compile time parameter. With the data hiding inherent in C++ we are able to implement AMR in such a way that the underlying data representation is restricted to a few model dependent classes. Changing the data representation, such as to a sparse data representation for a multifluid version

of the code, would be restricted to the member functions of these classes. Since the majority of the run time for AMR is spent integrating large rectangular meshes, the overhead experienced by doing the data management in C++ is just a few percent over an implementation done entirely in FORTRAN. Further optimization of the most important member functions reduces the running time to a few percent less than the pure FORTRAN code.

5. Numerical example. To test the performance of the three-dimensional AMR algorithm, we have modeled the interaction of a Mach 1.25 shock in air hitting an ellipsoidal bubble of Freon-22. The case being modeled is analogous to one of the experiments described by Haas and Sturtevant [12]. The Freon is a factor of 3.0246 more dense than the surrounding air, which leads to a deceleration of the shock as it enters the cloud and a subsequent generation of vorticity that dramatically deforms the bubble.

The computational domain is a rectangular region with length (x) 22.5 cm and width (y) and height (z) of 8.9 cm. The major axis of the bubble is 3.5 cm and is aligned with the z-axis. The minor axes are 2.5 cm with circular cross sections of the bubble in the $x - y$ plane. The bubble is centered at the point $(x, y, z) = (10 \text{ cm}, 0 \text{ cm}, 0 \text{ cm})$. The shocked fluid is placed at points less than or equal to 14.5 cm in the x direction. The shock moves in the direction of increasing x. We use the operator split second-order Godunov method of [6], with Strang splitting. Reflecting boundary conditions are set on the constant z and constant y planes. The inflow and outflow velocities on the constant x planes, as well as the interior fluid velocities, are set so the frame of reference is shifted to one in which the post-shock velocity is zero in order to minimize the x extent of the problem. The numerics preserve a four-fold symmetry in y and z, so we only compute on a quarter of the physical domain. (We have reflected the data in the renderings so that the entire domain is shown.)

We use a simplified treatment of the equations of multifluid gas dynamics. The features of this method include using a single fluid solver, advecting only one additional quantity, and solving a set of equations in conservation form.

We use a γ-law equation of state for each gas with $\gamma_a = 1.4$ for air and $\gamma_f = 1.25$ for Freon. Mixtures of the two gases are modeled with an equation of state defined using both γ's,

$$\Gamma_c = \frac{1}{\frac{f}{\gamma_f} + \frac{(1-f)}{\gamma_a}}$$

for sound speeds, and

$$\Gamma_e = 1 + \frac{1}{\frac{f}{\gamma_f - 1} + \frac{(1-f)}{\gamma_a - 1}}$$

for energy. Here, Γ_c is used to compute an effective sound speed for the mixture with

$$c = \sqrt{\frac{\Gamma_c p}{\rho}}, \tag{2}$$

and f is the mass fraction of the Freon. The harmonic average used to compute Γ_c, used by Colella, Ferguson, and Glaz for multifluid computations [5], expresses the net volume change of a mixture of the gases in terms of their individual compressibilities. The sound speed defined by (2) is used in the integration routine for defining characteristic speeds and for approximate solution of the Riemann problem. We also assume that the two components of a mixed fluid cell are at the same pressure. Pressure is computed from density and internal

energy using Γ_e, namely,

$$p = (\Gamma_e - 1)\,\rho\, e.$$

The harmonic average used to compute Γ_e insures that mixing of the two fluids at the same pressure does not result in a pressure and internal energy change of the composite fluid.

We ran the AMR algorithm with an initial coarse grid of $80 \times 16 \times 16$ with two levels of refinement, each by a factor of four, for 100 coarse grid timesteps. The integration used six conserved quantities (mass, momentum, energy, and mass of Freon). The addition of this extra conserved variable besides the usual five found in three-dimensional gas dynamics does not change the AMR structure. The error estimation procedure was modified so that the finest level grids (level 3) only existed in a neighborhood of the bubble and so that acoustic waves away from the bubble were not refined once the incident shock was well past the bubble. The computation was performed on a Cray-2 and required about 20 hours of CPU time. In Fig. 3, we show volume renderings of the density field at four times during the evolution of the cloud. For the renderings, we interpolated all the data onto a uniformly fine grid. The effective result of the volume rendering is to yield an isosurface of the interface between air and Freon. The earliest frame is shortly after the incident shock has completely passed through the bubble. The bubble evolution qualitatively agrees with the experimental results of Haas and Sturtevant. The dominant features of the flow are a strong jet coming from the back of the bubble and the unstable evolution of the roll-up of the outer portion of the bubble.

During the computation, a total of 1.77×10^9 cells were advanced by the integration routine (not including boundary advancements required by operator splitting) for an effective time of 40 μ-seconds per zone including all of the AMR overhead. Figures 4 and 5 present a more detailed breakdown of the algorithm performance. Figure 4 shows a histogram of the dimensions of the level 3 grids during the entire calculation. In this figure, each dimension is counted separately in adding up the total number of grids of a given size. For this computation the maximum grid dimension was limited to 50 and most grids were at least 24 cells wide in each direction. This was done to limit an individual grid size to 125000 zones times six variables per zone. Since the integration algorithm is operator split and each dimension determines the vector length for one step, this gives a measure of the quality of the grids.

Figure 5 shows the number of level 3 cells as a function of time and indicates a growth in memory requirements as the solution complexity grows. The maximum memory required at any point during the run was 22.6 Mwords.

Independent measurements of the integration algorithm on a single large grid $160 \times 64 \times 64$ indicated a time of 30 μ-seconds per zone. Thus, the additional boundary work, smaller vector lengths and AMR overhead increased the time per zone by 33%. However, to achieve the same resolution, a uniform grid $1280 \times 256 \times 256$ would be required. Such a computation would require 500 Mwords of storage and 1100 hours of CPU time. The net speedup with AMR is a factor of 55.

Although it is difficult to predict in advance, only 40 of the 80 zones in the x direction played a significant role in the computation. The difficulty in prediction of the extent of the problem is shown by cruder mesh calculations, which indicated a larger computation region. An outflow boundary condition to handle the reflected shock off of the bubble could have eliminated a region of length 40 zones in the x direction. Such a computation would require 251 Mwords of storage and 560 hours of CPU time. Alternatively, we estimate that with a carefully designed, exponentially stretched computation mesh, the fixed grid computational cost also could be halved; however, such an approach also requires substantial knowledge of the solution in designing the mesh. Thus, even from a conservative viewpoint, AMR reduced the computational cost by more than a factor of 20 for this problem.

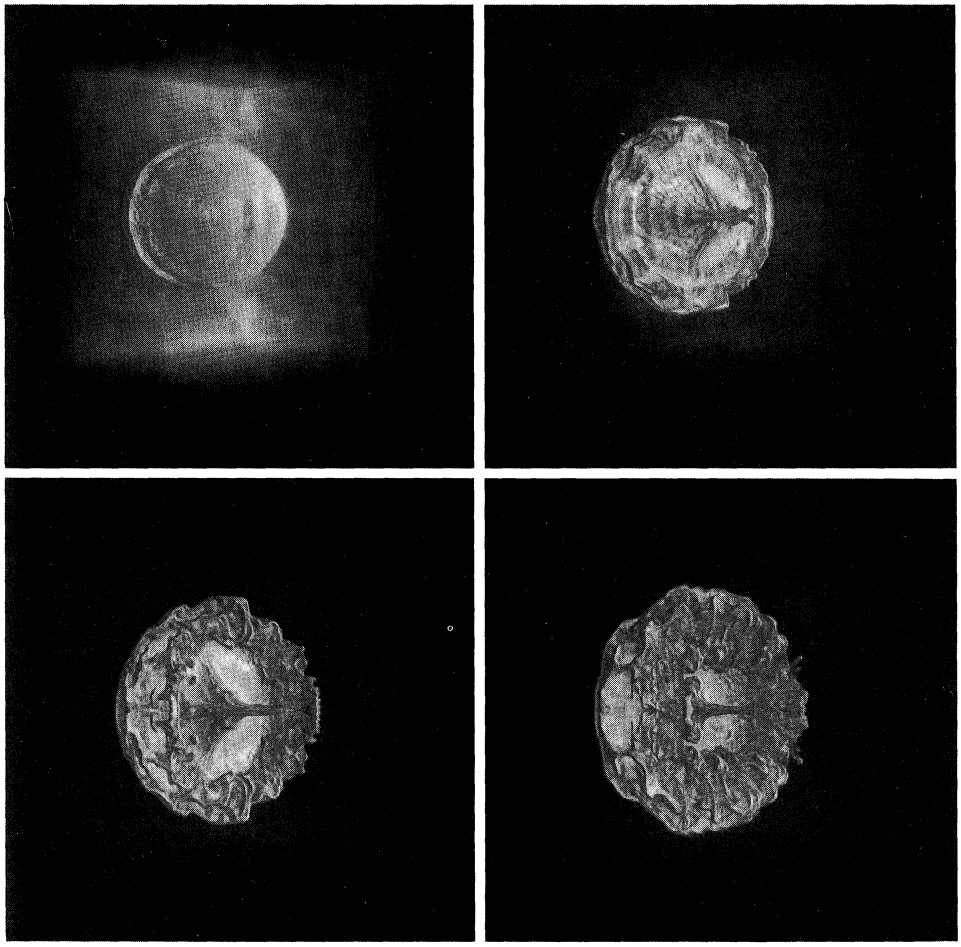


FIG. 3. *Volume renderings of the density during the evolution of the bubble after being hit with a shock wave.*

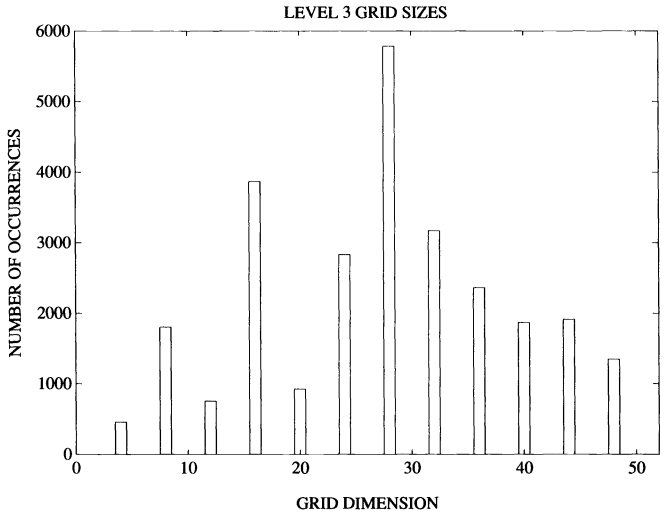


FIG. 4. *Histogram of the linear dimensions of all the level* 3 *grids.*

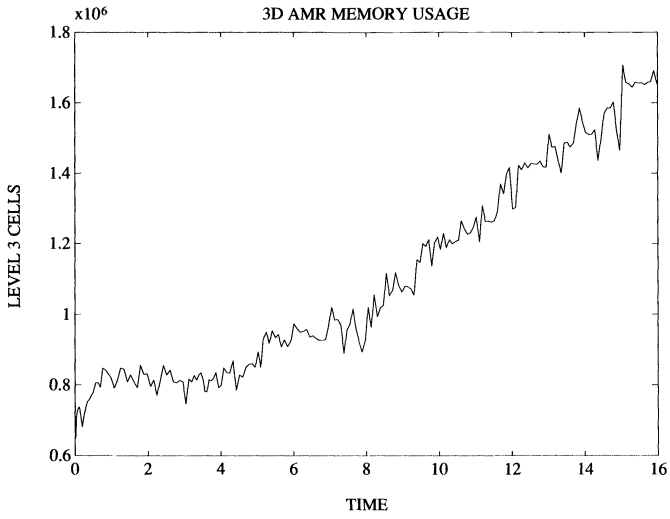


FIG. 5. *The number of level 3 cells as a function of time during the calculations.*

Of course, the performance of AMR is highly problem dependent. For some problems the cost reduction may be greater and for some problems it may be less. However, AMR will be cost effective as long as the average number of coarse cells per timestep that require the finest level of refinement throughout the computation is less than 75% of the total number. Although sophisticated grid placement strategies can reduce the advantage of using AMR, these strategies require considerable knowledge of the solution to be effective and may add considerable difficulties to problem setup. AMR provides a high level of performance while making problem setup routine.

REFERENCES

[1] J. D. BAUM AND R. LÖHNER, *Numerical design of a passive shock deflector using an adaptive finite element scheme on unstructured grids*, AIAA-92-0448 (1992).

[2] M. J. BERGER AND P. COLELLA, *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., 82 (1989), pp. 64–84.

[3] M. J. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[4] M. J. BERGER AND I. RIGOUTSOS, *An algorithm for point clustering and grid generation*, IEEE Trans. Systems Man and Cybernet, 21 (1991), pp. 1278–1286.

[5] P. COLELLA, R. E. FERGUSON, AND H. M. GLAZ, *Multifluid algorithm*, to appear.

[6] P. COLELLA AND H. M. GLAZ, *Efficient solution algorithms for the Riemann problem for real gases*, J. Comput. Phys., 59 (1985), pp. 264–289.

[7] P. COLELLA AND L. F. HENDERSON, *The von Neumann paradox for the diffraction of weak shock waves*, J. Fluid Mech., 213 (1990), pp. 71–94.

[8] P. COLELLA, R. KLEIN, AND C. MCKEE, *The nonlinear interaction of supernova shocks with galactic molecular clouds*, in Proc. Physics of Compressible Turbulent Mixing Intl. Workshop, Princeton, NJ, 1989.

[9] G. S. DIETACHMAYER AND K. K. DROEGEMEIER, *Application of continuous dynamic grid adaption techniques to meteorological modeling. Part i: Basic formulation and accuracy*, Mon. Wea. Rev., 120, (1992), pp. 1675–1706.

[10] H. A. DWYER, M. SMOOKE, AND R. J. KEE, *Adaptive gridding for finite difference solutions to heat and mass transfer problems*, in Numerical Grid Generation, J. Thompson, ed., Elsevier Science Publishing Co., New York, 1982.

[11] I. I. GLASS, J. KAC, D. L. ZHANG, H. M. GLAZ, J. B. BELL, J. A. TRANGENSTEIN, AND P. COLLINS, *Diffraction of planar shock waves over half-diamond and semicircular cylinders: an experimental and numerical comparison*, in Proc. 17th International Symposium on Shock Waves and Shock Tubes, Bethlehem, PA, 1989.

[12] J.-F. HAAS AND B. STURTEVANT, *Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities*, J. Fluid Mech., 181 (1987), pp. 41–76.

[13] L. F. HENDERSON, P. COLELLA, AND E. G. PUCKETT, *On the refraction of shock waves at a slow-fast gas interface*, J. Fluid Mech, to appear.

[14] D. J. MAVRIPILIS AND A. JAMESON, *Multigrid solution of the Euler equations on unstructured and adaptive meshes*, in Proc. Third Copper Mountain Conf. Multigrid Methods, Lecture Notes in Pure and Applied Mathematics, 1987; ICASE Report 87-53.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 139–148, January 1994

009

STABILITY OF THE PARTITIONED INVERSE METHOD FOR PARALLEL SOLUTION OF SPARSE TRIANGULAR SYSTEMS*

NICHOLAS J. HIGHAM† AND ALEX POTHEN‡

Abstract. Several authors have recently considered a parallel method for solving sparse triangular systems with many right-hand sides. The method employs a partition into sparse factors of the product form of the inverse of the coefficient matrix. It is shown here that while the method can be unstable, stability is guaranteed if a certain scalar that depends on the matrix and the partition is small and that this scalar is small when the matrix is well conditioned. Moreover, when the partition is chosen so that the factors have the same sparsity structure as the coefficient matrix, the backward error matrix can be taken to be sparse.

Key words. sparse matrix, triangular system, substitution algorithm, parallel algorithm, rounding error analysis, matrix inverse

AMS subject classifications. primary 65F05, 65F50, 65G05

1. Introduction.

The method of choice for solving triangular systems on a serial computer is the substitution algorithm. Several approaches have been suggested for parallel solution. Implementations of substitution for distributed memory architectures are described by Heath and Romine [12] and Li and Coleman [15], and a short survey of this work is given by Gallivan, Plemmons, and Sameh [9, §3.5.2] (see also [11, §6.4.4]). Implementations of substitution for sparse matrices on shared memory architectures are investigated by Rothberg and Gupta [20]. Algorithms that are not based on substitution are surveyed by Gallivan, Plemmons, and Sameh [9, §3.5], Heller [13], and Ortega and Voigt [16]. A new method has been developed recently for the parallel solution of sparse triangular systems with many right-hand sides when these vectors are not necessarily available at the same time [1]–[3], [8]. The method involves representing the inverse of the coefficient matrix as a product of sparse factors, and can be explained as follows.

If $L \in \mathbb{R}^{n\times n}$ is lower triangular, we can write $L = L_1 L_2 \dots L_n$, where L_k differs from the identity matrix only in the kth column:

$$L_k = \begin{bmatrix} I_{k-1} & & & & \\ & l_{kk} & & & \\ & l_{k+1,k} & 1 & & \\ & \vdots & & \ddots & \\ & l_{nk} & & & 1 \end{bmatrix}. \tag{1.1}$$

The factorization of L can be partitioned

$$L = G_1 G_2 \dots G_m, \tag{1.2}$$

where $1 \le m \le n$ and

$$G_k = L_{i_k} L_{i_k+1} \dots L_{i_{k+1}-1}, \qquad 1 = i_1 < i_2 < \cdots < i_{m+1} = n+1. \tag{1.3}$$

*Received by the editors November 10, 1992; accepted for publication (in revised form) March 4, 1993.

†Department of Mathematics, University of Manchester, Manchester, M13 9PL, England (na.nhigham@na-net.ornl.gov). The work of this author was supported by a Nuffield Science Research Fellowship, Science and Engineering Research Council grant GR/H52139, and the European Economic Community Esprit Basic Research Action Programme, Project 6634 (APPARC: Performance-Critical Applications of Parallel Architectures).

‡Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada (apothen@narnia.uwaterloo.ca, na.pothen@na-net.ornl.gov). The work of this author was supported by National Science Foundation grant CCR-9024954, U. S. Department of Energy grant DE-FG02-91ER25095 at the Pennsylvania State University, and Canadian Natural Sciences and Engineering Research Council grant OGP0008111 at the University of Waterloo.

Note that G_k is the lower triangular matrix equal to the identity except for columns $i_k, \ldots, i_{k+1}-1$, which equal the corresponding columns of $L_{i_k}, \ldots, L_{i_{k+1}-1}$, respectively. Defining $\overline{G}_k = G_k(:, i_k : i_{k+1} - 1)$ (using the colon notation from [11]) we have the relation

$$L = G_1 G_2 \ldots G_m = [\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_m], \tag{1.4}$$

which we will use later. Equation (1.2) yields the partitioned, product-form representation

$$L^{-1} = H_m H_{m-1} \ldots H_1, \qquad H_k = G_k^{-1}. \tag{1.5}$$

For a sparse matrix L, the idea behind the "partitioned inverse method" is to choose the partition (1.2) so that (1.5) represents L^{-1} as a short product of sparse factors. Then $Lx = b$ is solved by forming

$$x = H_m H_{m-1} \ldots H_1 b, \tag{1.6}$$

and the advantage is that x can be computed in m serial steps of parallel matrix–vector multiplication. Thus on a massively parallel SIMD computer such as the Connection Machine CM-2, only m communication steps involving the router are necessary in the algorithm. The scalar multiplications in each product $x_k = H_k x_{k-1}$ (where $x_0 = b$) can be done concurrently in time proportional to v, where v is the maximum number of elements of H_k assigned to a processor, and the additions can be done in logarithmic time [1]. The two extreme cases are $m = n$, which gives a modified version of forward substitution (or forward substitution itself if L has unit diagonal), and $m = 1$, for which the method forms $x = L^{-1} \times b$. For a sparse matrix L we would not take $m = 1$, because $H_1 = L^{-1}$ is usually much denser than L [7, §12.6]. Rather, we would like to minimize m subject to the condition that each factor H_k can be stored in the same space as G_k, since m is the number of serial steps in the parallel evaluation of x. Since we are assuming that many right-hand sides are to be processed, we can afford to spend some computational effort in constructing the partition (1.2).

Algorithms for finding a *best no-fill partition* (1.2) are described in [1]–[3]; such a partition has the smallest possible number of factors (the minimum value of m) subject to the requirement that each G_k is invertible in place. A matrix X is *invertible in place* if $(X^{-1})_{ij} = 0$ whenever $x_{ij} = 0$ for any assignment of (nonzero) numerical values to the nonzeros in X. Note that L_k in (1.1) is invertible in place, so a partition with $m = n$ is always a no-fill partition. When L is sparse, a best no-fill partition could have $m \ll n$. Partitions that incur some fill-in have also been investigated [3].

Algorithms are also given in [1] and [2] for finding a *best reordered partition*: this is a no-fill partition with the fewest factors over all lower triangular matrices PLP^T, where P is a permutation matrix. Let $F = L + L^T$ denote the *filled matrix* corresponding to a Cholesky factor. It is well known that if L is the Cholesky factor of a symmetric positive definite matrix A whose nonzero elements are algebraically independent, then the adjacency graph of F is chordal. By exploiting chordality, very efficient algorithms for computing best reordered partitions in time and space linear in the order of the matrix (rather than the number of nonzeros) can be designed for a Cholesky factor L [1], [19]. Furthermore, algorithms for finding a partition with the fewest factors over all permutations P such that the permuted matrix PFP^T has the same structure as the filled matrix F have also been designed [17], [18]. Note that, in this case, the permutation may change the structure of L, and hence the permutation P has to be applied to A *before* it is factored.

The numerical stability of the partitioned inverse method has not been studied in previous work, either theoretically or in numerical experiments. The numerical stability is clearly questionable because when $m = 1$ (which gives the best no-fill partition for a dense matrix)

the method computes $x = L^{-1} \times b$, and a numerical example in [6, §4] shows that this evaluation need not be backward stable. To answer the question of stability we have done an error analysis of the partitioned inverse method; this analysis is presented in §2. In §3 we describe some numerical experiments that illustrate the analysis and confirm the possible numerical instability of the method.

Our main findings are as follows.

(1) In general, the partitioned inverse method does not satisfy the componentwise backward and forward error bounds enjoyed by the substitution algorithm (namely, (2.1) and (2.2)).

(2) Normwise stability depends on a quantity ρ, defined in (2.13), which is a function of the matrix L and the partition, and which can be arbitrarily large. Specifically, the computed solution $\widehat{x}$ to $Lx = b$ satisfies $(L + \Delta L)\widehat{x} = b$, where $\|\Delta L\|_\infty$ is bounded in (2.12); the relative error $\|x - \widehat{x}\|_\infty / \|x\|_\infty$ is bounded in (2.19). If ρ is of order 1, which is guaranteed if L is well conditioned, the partitioned inverse method is both normwise backward stable and normwise forward stable.

(3) If L is sparse and each G_k is invertible in place (as is guaranteed by a best no-fill or best reordered partition), then the backward error matrix ΔL mentioned in (2) can be taken to have the same sparsity structure as L.

Another way to summarize the stability of the partitioned inverse method is to say that the method is only conditionally stable, with the backward error dependent on the condition number of L. The partitioned inverse method therefore provides another example, to add to those discussed by Demmel [4], of how parallelism can conflict with stability.

In future work we intend to examine how particular sparsity structures and other special properties of L affect the stability of the partitioned inverse method.

2. Error analysis. In this section we give an error analysis of the partitioned inverse method for solving $Lx = b$. To keep the analysis general we will not make any assumptions about sparsity. As our model of floating point arithmetic we take

$$\begin{aligned} fl(x \pm y) &= x(1+\alpha) \pm y(1+\beta), \qquad |\alpha|, |\beta| \le u, \\ fl(x \text{ op } y) &= (x \text{ op } y)(1+\delta), \qquad |\delta| \le u, \quad \text{op} = *, /, \end{aligned}$$

where u is the unit roundoff. This model admits machines that lack a guard digit in addition and subtraction. We place a hat over a variable to indicate a computed quantity.

For later comparison we summarise what can be said about the substitution algorithm. The computed $\widehat{x}$ satisfies (see, for example, [22, p. 150] or [14])

$$(L + \Delta L)\widehat{x} = b, \qquad |\Delta L| \le \big((n+1)u + O(u^2)\big)|L|. \tag{2.1}$$

(Absolute values and inequalities are interpreted componentwise for matrices.) This result shows that there is a componentwise tiny backward error matrix ΔL that has the same sparsity structure as L. From (2.1) it is easy to obtain the forward error bound

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \le (n+1)u \,\mathrm{cond}(L, x) + O(u^2), \tag{2.2}$$

which contains the Bauer–Skeel condition number

$$\mathrm{cond}(L, x) = \frac{\| \, |L^{-1}||L||x| \, \|_\infty}{\|x\|_\infty}.$$

This bound may be weakened to

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \le (n+1)u\kappa_\infty(L) + O(u^2), \tag{2.3}$$

where $\kappa_\infty(L) = \|L\|_\infty \|L^{-1}\|_\infty$. Our aim is to see how close the partitioned inverse method comes to achieving the ideal bounds (2.1) and (2.2). Note that if L is sparse, the constant $n+1$ in (2.1)–(2.3) can be replaced by $p+1$, where p is the maximum number of nonzeros per row over all the rows of L. Similarly, the constants c_k that we define below to equal the partition widths can be redefined to take account of sparsity.

First, we consider the computation of the factors $H_k = G_k^{-1}$ of L^{-1} (from L). Because of its special structure, G_k is formed without error, and we assume that G_k^{-1} is computed by one of the several stable methods described by Du Croz and Higham in [6] (for example, its columns may be computed one at a time by forward substitution). For each of these methods applied to G_k, precisely one of the following two residual bounds holds, depending on the method:

$$|\widehat{H}_k G_k - I| \le \bigl(c_k u + O(u^2)\bigr)|\widehat{H}_k||G_k|, \tag{2.4}$$

$$|G_k \widehat{H}_k - I| \le \bigl(c_k u + O(u^2)\bigr)|G_k||\widehat{H}_k|, \tag{2.5}$$

where $c_k = i_{k+1} - i_k + 1$. Each residual bound implies the forward error bound

$$|\widehat{H}_k - G_k^{-1}| \le c_k u |H_k||G_k||H_k| + O(u^2). \tag{2.6}$$

(Since we are working to first order, $\widehat{H}_k$ and H_k are interchangeable in all the bounds.)

Applying standard error analysis of matrix–vector products [11, p. 66] to (1.6) we obtain

$$\widehat{x} = (\widehat{H}_m + \Delta_m)(\widehat{H}_{m-1} + \Delta_{m-1}) \dots (\widehat{H}_1 + \Delta_1) b, \tag{2.7}$$

where

$$|\Delta_k| \le \bigl(c_k u + O(u^2)\bigr)|\widehat{H}_k|. \tag{2.8}$$

If the inner products that occur in the matrix–vector products are evaluated using the fan-in algorithm for summation, then the constant c_k in (2.8) can be replaced by $\log_2 c_k$. We can rewrite (2.7) as $(L + \Delta L)\widehat{x} = b$, where

$$L + \Delta L = (H_1 + E_1)^{-1}(H_2 + E_2)^{-1} \dots (H_m + E_m)^{-1}, \tag{2.9}$$

with

$$E_k = \Delta_k + (\widehat{H}_k - H_k). \tag{2.10}$$

Now we consider the sparsity of ΔL. First, we note that if H_k is computed by forward substitution, then, by (2.1), its jth column $\widehat{h}_j$ satisfies $(G_k + F_k^{(j)})\widehat{h}_j = e_j$, where e_j is the jth unit vector and $|F_k^{(j)}| \le \bigl((n+1)u + O(u^2)\bigr)|G_k|$, so that $F_k^{(j)}$ has the same sparsity structure as G_k. It follows that if G_k is invertible in place, then $\widehat{H}_k$ (whose jth column is that of $(G_k + F_k^{(j)})^{-1}$) has the same sparsity structure as G_k. The same is true for any of the stable methods in [6] because, as explained there, these methods all incur essentially the same rounding errors. Next, we observe that by (2.8), Δ_k has the same sparsity structure as $\widehat{H}_k$, and, therefore, if G_k is invertible in place, then $(H_k + E_k)^{-1} = (\widehat{H}_k + \Delta_k)^{-1}$ has the same sparsity structure as G_k. From (2.9) and the structural relation $L = G_1 G_2 \dots G_m$, we conclude that *if each G_k is invertible in place then the backward error matrix ΔL has the same sparsity structure as L*. It remains to bound ΔL.

From (2.9) we have

$$\Delta L = -\sum_{k=1}^{m} H_1^{-1} \dots H_{k-1}^{-1} \cdot H_k^{-1} E_k H_k^{-1} \cdot H_{k+1}^{-1} \dots H_m^{-1} + O(u^2),$$

so that

$$|\Delta L| \le \sum_{k=1}^{m} |H_1^{-1} \dots H_{k-1}^{-1}| \cdot |H_k^{-1} E_k H_k^{-1}| \cdot |H_{k+1}^{-1} \dots H_m^{-1}| + O(u^2).$$

If (2.4) holds, then, from (2.8) and (2.10),

$$\begin{aligned} |H_k^{-1} E_k H_k^{-1}| &= |H_k^{-1} \Delta_k H_k^{-1} + H_k^{-1}(\widehat{H}_k H_k^{-1} - I)| \\ &\le c_k u |H_k^{-1}||H_k||H_k^{-1}| + c_k u |H_k^{-1}||H_k||G_k| + O(u^2) \\ &= 2c_k u |G_k||H_k||G_k| + O(u^2), \end{aligned}$$

and precisely the same bound is obtained if we use (2.5).

Define $d_n = 2 \max_k c_k$. To obtain a convenient bound for ΔL we use (1.4), together with the observation that since $|G_k||H_k||G_k|$ differs from the identity only in columns $i_k, \dots, i_{k+1} - 1$, it can be treated like $|G_k|$ when we invoke (1.4). We have

$$\begin{aligned} |\Delta L| &\le d_n u \sum_{k=1}^{m} |G_1| \dots |G_{k-1}| \cdot |G_k||H_k||G_k| \cdot |G_{k+1}| \dots |G_m| + O(u^2) \\ &\le d_n u \sum_{k=1}^{m} \big[|\overline{G}_1|, \dots, |\overline{G}_{k-1}|, \overline{|G_k||H_k||G_k|}, |\overline{G}_{k+1}|, \dots, |\overline{G}_m| \big] + O(u^2) \\ &\le d_n u \big((m-1)|L| + \sum_{k=1}^{m} [0, \dots, 0, \overline{|G_k||H_k||G_k|}, 0, \dots, 0] \big) + O(u^2) \\ &= d_n u \big((m-1)|L| + \sum_{k=1}^{m} |G_k||H_k||G_k| - (m-1)I \big) + O(u^2). \end{aligned} \tag{2.11}$$

This bound is not of the form (2.1) that holds for substitution, because of the summation term. If $m = 1$, the bound is $|\Delta L| \le 2(n+1)u|L||L^{-1}||L| + O(u^2)$. When $m = n$, the relation $|L_k||L_k^{-1}||L_k| \le 3|L_k|$ allows us to simplify the bound to $|\Delta L| \le 4(n+2)u|L| + O(u^2)$, which is of the same form as in (2.1).

Taking norms in (2.11) we obtain

$$\|\Delta L\|_\infty \le d_n u (m - 1 + \rho) \|L\|_\infty + O(u^2), \tag{2.12}$$

where

$$\rho = \frac{\big\| \sum_{k=1}^{m} |G_k||G_k^{-1}||G_k| - (m-1)I \big\|_\infty}{\|L\|_\infty}. \tag{2.13}$$

The scalar $\rho \ge 1$ might be loosely described as a growth factor for the partitioned inverse method, although it is not related to the growth factor in Gaussian elimination. For any $m < n$, ρ can be arbitrarily large, but for $m = n$ it is easy to show that $\rho \le 3$. Under scaling of the system, ρ behaves as follows: if $D_1 = \operatorname{diag}(d_i)$ and $D_2 = \operatorname{diag}(e_i)$ are nonnegative diagonal matrices and we scale $Lx = b \to (D_1 L D_2) \cdot (D_2^{-1} x) = D_1 b$, then

$$\rho \to \overline{\rho} = \frac{\big\| \sum_{k=1}^{m} D_1^{(k)} |G_k||G_k^{-1}||G_k| D_2^{(k)} - (m-1)I \big\|_\infty}{\|D_1 L D_2\|_\infty},$$

where $D_1^{(k)} = \operatorname{diag}(1, \dots, 1, d_{i_k}, \dots, d_n)$ and $D_2^{(k)} = \operatorname{diag}(1, \dots, 1, e_{i_k}, \dots, e_{i_{k+1}-1}, 1, \dots, 1)$ (recall that i_k is defined in (1.3)). This expression suggests that ρ is fairly insensitive to the scaling of the rows and columns of L.

We see from (2.12) that if L and the partition are such that ρ is of order one, then the partitioned inverse method is normwise backward stable; that is, the computed solution $\widehat{x}$ solves a system obtained by making a tiny normwise perturbation to L. Using (2.12), the sparse backward error property noted earlier can be expressed as follows. Define $Z \in \mathbb{R}^{n\times n}$ by

$$z_{ij} = \begin{cases} 1 & \text{if } l_{ij} \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

Then, if each G_k is invertible in place,

$$(L + \Delta L)\widehat{x} = b, \qquad |\Delta L| \leq \big(d_n u(m - 1 + \rho)\|L\|_\infty + O(u^2)\big)Z, \tag{2.15}$$

where the matrix inequality both bounds Δl_{ij} and shows that ΔL has the same sparsity structure as L.

Two useful upper bounds can be obtained for ρ. By examining the form of the matrix whose norm is the numerator in (2.13), it is easy to show that $\rho \leq m \max_k \kappa_\infty(G_k)$. From the relation

$$\begin{aligned} G_k^{-1} &= G_{k+1} \ldots G_m L^{-1} G_1 \ldots G_{k-1} \\ &= G_{k+1} \ldots G_m \begin{bmatrix} I_{i_k-1} & 0 \\ 0 & L^{-1}(i_k\!:\!n, i_k\!:\!n) \end{bmatrix}, \end{aligned}$$

we have $\|G_k^{-1}\|_\infty \leq \max(1, \|L\|_\infty)\max(1, \|L^{-1}\|_\infty)$. As ρ is invariant under scalar multiplication of L, we can assume, without loss of generality, that $\|L\|_\infty = 1$, and hence we have, for all partitions,

$$\rho \leq m\kappa_\infty(L).$$

We conclude that the normwise backward error for the partitioned inverse method is bounded by a multiple of $\kappa_\infty(L)u$. Although this bound may be very weak when L is ill conditioned, it shows that *if L is well conditioned then the partitioned inverse method is guaranteed to be normwise backward stable.*

It is interesting to note that dependence of the backward error on the condition number occurs also in block LU factorization [5]. Another example of this dependence is a parallel triangular system solver analysed by Sameh and Brent [21], for which a backward error result with $\|\Delta L\| \leq c_n u\kappa^2(L)\|L\|$ is obtained. It seems to be a rule of thumb that if we attempt to improve the parallelism of Gaussian elimination or substitution, we will achieve only conditional stability, with the backward error potentially proportional to some function of the condition number.

Now we turn to the forward error. One way to obtain a forward error bound is to expand the equation $\widehat{x} = (H_m + E_m)(H_{m-1} + E_{m-1}) \ldots (H_1 + E_1)b$, which follows from (2.7) and (2.10). For $m = 1$, this leads to the bound

$$\begin{aligned} \frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} &\leq 2(n+1)u \frac{\| \, |L^{-1}||L||L^{-1}||b| \, \|_\infty}{\|x\|_\infty} + O(u^2) \\ &\leq 2(n+1)u\,\theta \| \, |L^{-1}||L| \, \|_\infty + O(u^2), \end{aligned} \tag{2.16}$$

where

$$\theta = \frac{\| \, |L^{-1}||b| \, \|_\infty}{\|L^{-1}b\|_\infty} \geq 1. \tag{2.17}$$

The scalar θ can be regarded as a measure of forward stability for the $L^{-1}b$ method. Note that θ is large only when there is a lot of cancellation through subtraction in the product $L^{-1}b$. Gill, Murray, and Wright [10, §4.7.2] analyse the $L^{-1}b$ method under the simplifying assumption that the computed inverse is the correctly rounded inverse. Their forward error bound from a normwise analysis is proportional to $\kappa_\infty(L)(\|L^{-1}\|_\infty\|b\|_\infty/\|x\|_\infty)$, and so is consistent with our bound.

For general m, a more useful bound is obtained by manipulating the backward error result. Since $(L+\Delta L)\widehat{x}=b$ implies $|x-\widehat{x}|\le|L^{-1}||\Delta L||\widehat{x}|$, we obtain from (2.11)

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty}\le d_n u\frac{\left\|\left((m-1)|L^{-1}||L|+|L^{-1}|\left(\sum_{k=1}^m|G_k||H_k||G_k|-(m-1)I\right)\right)|x|\right\|_\infty}{\|x\|_\infty}+O(u^2). \tag{2.18}$$

The summation term precludes this bound from matching the ideal bound (2.2), but (2.18) does share with (2.2) the very desirable property that it is independent of the row scaling of the system. We can weaken (2.18) to obtain

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty}\le d_n u(m-1+\rho)\kappa_\infty(L)+O(u^2), \tag{2.19}$$

where ρ is the growth factor in (2.13) (of course, this bound could have been obtained directly using (2.12)). Hence if ρ is of order one, then the partitioned inverse method satisfies a bound of the form (2.3), that is, it is normwise forward stable.

3. Numerical experiments. We describe two numerical experiments that illustrate the error analysis and confirm the potential instability of the partitioned inverse method. Our experiments were performed in Matlab, which has unit roundoff $u\approx 1.1\times 10^{-16}$. Statistics that we report include

$$\begin{aligned}\text{nberr}&=\min\{\epsilon:|b-L\widehat{x}|\le\epsilon\|L\|_\infty ee^T|\widehat{x}|\},\\ \text{sberr}&=\min\{\epsilon:|b-L\widehat{x}|\le\epsilon\|L\|_\infty Z|\widehat{x}|\},\\ \text{cberr}&=\min\{\epsilon:|b-L\widehat{x}|\le\epsilon|L||\widehat{x}|\},\end{aligned}$$

and

$$\text{ferr}=\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty},$$

where $e=(1,1,\dots,1)^T$ and Z is defined in (2.14). The quantity nberr is the usual normwise backward error, written in a way that shows its connection with the "sparse normwise backward error" sberr. From (2.12) it follows that, to first order, nberr $\le d_n u(m-1+\rho)$, and, if each G_k is invertible in place, sberr satisfies the same bound, by (2.15). The componentwise backward error cberr is $O(u)$ for substitution, by (2.1). We mention that in both experiments, modifying the backward errors nberr, sberr, and cberr to include a b term (thus allowing b to be perturbed in the definition of backward error) changes the backward errors by at most a factor 2.

In our first experiment $L=R^T$, where $V=QR$ is a QR factorization of the 15×15 Vandermonde matrix with (i,j) element $((j-1)/(n-1))^{i-1}$, and $b=Le$. This linear system is taken from [6, §4]. We solved the system $Lx=b$ by using the partitioned inverse method with "fixed-width" partitions (1.3) having $i_{k+1}=i_k+p$, for several values of p. Results are reported in Table 3.1; since L is dense, nberr $\approx$ sberr, so we do not give the sberr values. We see that as p increases, the normwise backward error increases and the algorithm loses

backward stability. In these examples, both (2.12) and (2.18) are a factor $\approx 10^3$ from being equalities for $p \geq 4$, and the bound $\rho \leq m \max_k \kappa_\infty(G_k)$ is clearly very weak. The ideal forward error bounds (2.2) and (2.3) are 6.43e−4 and 3.87e−3, respectively; ferr exceeds both values for $p \geq 6$, so the algorithm also loses forward stability. The quantity θ in (2.17) has the value $\theta = 3.60\text{e}11$, so the bound (2.16) predicts forward instability when $m = 1$, but is a factor $\approx 10^8$ from being an equality. We also solved $Lx = c$, where $c = (-1, 1, -1, 1, \ldots)^T$, and the result for $p = 15$ is reported in the last line of Table 3.1. Here, $\theta = 1$, and, as predicted by (2.16), the algorithm performs in a forward stable manner; the tiny backward error is not predicted by our analysis.

TABLE 3.1
Dense system $Lx = b$, $n = 15$.

$\kappa_\infty(L) = 2.18\text{e}12$, $\text{cond}(L, x) = 3.62\text{e}11$

p	m	nberr	ferr	ρ	$\max_k \kappa_\infty(G_k)$
1	15	0.00	2.04e−6	3.00	8.38e6
2	8	4.98e−18	5.68e−6	2.65e1	1.55e7
4	4	5.77e−16	4.94e−4	1.49e3	9.51e8
6	3	2.14e−14	4.36e−3	3.62e4	9.51e8
8	2	9.10e−14	8.42e−3	5.68e5	1.03e11
10	2	4.73e−13	7.34e−3	2.04e6	1.41e10
12	2	6.63e−13	1.11e−2	2.72e6	3.83e9
15	1	6.60e−13	1.13e−2	2.78e6	2.18e12
	$Lx = c$, $\text{cond}(L, x) = 3.90\text{e}4$				
15	1	9.07e−23	8.78e−11	2.78e6	2.18e12

In our second experiment L is a 20×20 matrix with 58 nonzeros; the entries and their locations were chosen randomly and then manipulated "by hand" to produce interesting behaviour. The inverse of L has 93 nonzeros. We solved the system $Lx = b$, where $b = Le$, using five different partitions. Partition A corresponds to forming $\widehat{x} = L^{-1} \times b$ ($m = 1$), Partition B has $m = 2$ with $G_1 = L_1 L_2 \ldots L_{14}$ and $G_2 = L_{15} \ldots L_{20}$, Partition C is a best no-fill partition, Partition D is a best reordered partition, and Partition E gives a variant of forward substitution ($m = n$). (Recall that Partition D corresponds to a partition of a symmetric permutation of L that preserves its lower triangular structure.) In Table 3.2 we report the backward errors and ρ (the system is too ill conditioned for us to determine the forward errors, but the computed solutions probably have no correct digits).

TABLE 3.2
Sparse system $Lx = b$, $n = 20$.

$\kappa_\infty(L) = 2.69\text{e}28$, $\text{cond}(L, x) = 1.47\text{e}16$

Partition $(i_1, i_2, \ldots, i_{m+1})$	nberr	sberr	cberr	ρ
A: (1,21)	5.49e−9	4.26e−6	2.47e−1	7.36e19
B: (1,15,21)	1.83e−14	4.26e−6	9.48e−2	1.06e14
C: (1,2,3,4,11,12,13,19,21)	4.03e−24	1.55e−14	6.96e−14	5.67e3
D: (1,5,9,11,17,19,21)	4.93e−24	1.55e−14	6.96e−14	5.67e3
E: (1,2,3,...,21)	7.27e−24	3.80e−17	1.70e−16	3.00

The results confirm two properties suggested by the analysis.

(1) For partitions in which the factors are not invertible in place (Partitions A and B in the table), the sparse backward error can greatly exceed the normwise backward error.

(2) Even a best no-fill partition can yield sparse or componentwise backward errors appreciably larger than those for substitution.

We have been unable to construct a numerical example where the sparse backward error is large even when ρ is small, which the analysis suggests may be possible for partitions where the factors are not invertible in place. Our limited experience with the partitioned inverse method suggests that, like substitution, it frequently achieves surprisingly small forward and backward errors in practice. However, in view of the possible instability it is wise to compute one of the backward errors and make an *a posteriori* test for stability. Alternatively, if many right-hand sides are to be handled, it may be preferable to compute ρ or estimate its upper bound $m\kappa_\infty(L)$ before solving the systems. If any of these tests reveal or predict instability, substitution could be used instead.

Acknowledgements. We thank Des Higham for suggesting improvements to the manuscript.

REFERENCES

[1] F. L. ALVARADO, A. POTHEN, AND R. S. SCHREIBER, *Highly parallel sparse triangular solution*, Research Report CS-92-51, Dept. Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, Oct. 1992. To appear in Sparse Matrix Computations: Graph Theory Issues and Algorithms, J. A. George, J. R. Gilbert, and J. W. H. Liu, eds., IMA Volumes in Mathematics and its Applications, Springer-Verlag Berlin.

[2] F. L. ALVARADO AND R. S. SCHREIBER, *Optimal parallel solution of sparse triangular systems*, SIAM J. Sci.Comput., 14 (1993), pp. 446–460.

[3] F. L. ALVARADO, D. C. YU, AND R. BETANCOURT, *Partitioned sparse A^{-1} methods*, IEEE Trans. Power Systems, 5 (1990), pp. 452–458.

[4] J. W. DEMMEL, *Trading off parallelism and numerical stability*, in Linear Algebra for Large Scale and Real-Time Applications, M. S. Moonen, G. H. Golub, and B. L. D. Moor, eds., Vol. 232, NATO ASI Series E, Kluwer Academic Publishers, Dordrecht, 1993, pp. 49–68.

[5] J. W. DEMMEL, N. J. HIGHAM, AND R. S. SCHREIBER, *Block LU factorization*, Numerical Analysis Report 207, University of Manchester, England, Feb. 1992; J. Numer. Linear Algebra Appl., submitted.

[6] J. J. DU CROZ AND N. J. HIGHAM, *Stability of methods for matrix inversion*, IMA J. Numer. Anal., 12 (1992), pp. 1–19.

[7] I. S. DUFF, A. M. ERISMAN, AND J. K. REID, *Direct Methods for Sparse Matrices*, Oxford University Press, 1986.

[8] M. K. ENNS, W. F. TINNEY, AND F. L. ALVARADO, *Sparse matrix inverse factors*, IEEE Trans. Power Systems, 5 (1990), pp. 466–472.

[9] K. A. GALLIVAN, R. J. PLEMMONS, AND A. H. SAMEH, *Parallel algorithms for dense linear algebra computations*, SIAM Rev., 32 (1990), pp. 54–135.

[10] P. E. GILL, W. MURRAY, AND M. H. WRIGHT, *Numerical Linear Algebra and Optimization, Volume* 1, Addison–Wesley, Reading, MA, 1991.

[11] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, Maryland, 1989.

[12] M. T. HEATH AND C. H. ROMINE, *Parallel solution of triangular systems on distributed-memory multiprocessors*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 558–588.

[13] D. HELLER, *A survey of parallel algorithms in numerical linear algebra*, SIAM Rev., 20 (1978), pp. 740–777.

[14] N. J. HIGHAM, *The accuracy of solutions to triangular systems*, SIAM J. Numer. Anal., 26 (1989), pp. 1252–1265.

[15] G. LI AND T. F. COLEMAN, *A new method for solving triangular systems on distributed-memory message-passing multiprocessors*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 382–396.

[16] J. M. ORTEGA AND R. G. VOIGT, *Solution of partial differential equations on vector and parallel computers*, SIAM Rev., 27 (1985), pp. 149–240.

[17] B. W. PEYTON, A. POTHEN, AND X. YUAN, *Partitioning a chordal graph into transitive subgraphs for parallel sparse triangular solution*, Tech. Report CS-92-55, Dept. Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, Dec. 1992; Linear Algebra Appl., to appear.

[18] B. W. PEYTON, A. POTHEN, AND X. YUAN, *A clique tree algorithm for partitioning a chordal graph into transitive subgraphs*, Tech. Report CS-93-27, Dept. Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada, May 1993.

[19] A. POTHEN AND F. L. ALVARADO, *A fast reordering algorithm for parallel sparse triangular solution*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 645–653.

[20] E. ROTHBERG AND A. GUPTA, *Parallel* ICCG *on a hierarchical memory multiprocessor—Addressing the triangular solve bottleneck*, Parallel Comput., 18 (1992), pp. 719–741.

[21] A. H. SAMEH AND R. P. BRENT, *Solving triangular systems on a parallel computer*, SIAM J. Numer. Anal., 14 (1977), pp. 1101–1113.

[22] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 149–163, January 1994

010

TWO-LEVEL LOCAL REFINEMENT PRECONDITIONERS FOR NONSYMMETRIC AND INDEFINITE ELLIPTIC PROBLEMS*

RICHARD E. EWING†, SVETOZARA I. PETROVA‡, AND PANAYOT S. VASSILEVSKI§

Abstract. Preconditioners of optimal order for nonselfadjoint and indefinite elliptic boundary value problems discretized on grids with local refinement are constructed. The proposed technique utilizes solution of a discrete problem on a uniform coarse grid; then, the reduced problem is handled by a generalized conjugate gradient (GCG) method. The reduced problem is coercive if the initial coarse mesh is sufficiently fine and is local, solving only for the unknowns on the subdomains where local refinement has been introduced. The reduced problem can be preconditioned by a preconditioner for the symmetric positive definite matrix arising from the symmetric and coercive principal part of the original bilinear form restricted to the subdomains containing local refinement. This problem also utilizes a uniform grid. In the numerical tests, the recent algebraic multilevel (AMLI) preconditioners [Axelsson and Vassilevski, *SIAM J. Numer. Anal.*, 27 (1990), pp. 1569–1590; Saad and Schultz, *SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 856–869], which are of optimal order for selfadjoint and coercive elliptic problems, were used.

Key words. indefinite problems, nonsymmetric problems, optimal order preconditioner, local refinement, two-level method, generalized conjugate gradients, elliptic problems

AMS subject classifications. 65F10, 65N20, 65N30

1. Introduction.

Recently much interest has been devoted to the construction of preconditioners for solving discretized elliptic and parabolic problems on grids with local refinement (for example, see the so-called fast adaptive composite (FAC) grid method of McCormick [20], McCormick and Thomas [22], Mandel and McCormick [19], and the monograph by McCormick [21]. Also, a symmetrized version of the FAC grid-method was proposed in Bramble, Ewing, Pasciak, and Schatz [9] and studied further in Ewing, Lazarov, and Vassilevski [15] for elliptic problems and in Ewing, Lazarov, Pasciak, and Vassilevski [13] for parabolic problems with local refinement in time and space. These methods have been also generalized to the case of multilevel local refinement by Widlund [29], Dryja and Widlund [12], Ewing, Lazarov, and Vassilevski [14], and Vassilevski, Petrova, and Lazarov [28]. The additive version of the multigrid method as proposed by Bramble, Pasciak, and Xu [10] is also suitable for multilevel local refinement.

Another approach, based on the two-level method of Bank and Dupont [7] and studied further in Axelsson and Gustafsson [3], was successfully exploited in Ewing and Vassilevski [17] in the context of local refinement methods for two-level and multilevel grids. In the present paper, we extend one of the two-grid preconditioners (the so-called two-level block Gauss–Seidel preconditioner) to the case of nonsymmetric and possibly indefinite finite element discretization matrices. The extension is based on the paper of Vassilevski [26] and uses the fact observed by Schatz [25] that, within a certain subspace, the bilinear form corresponding to our elliptic problem is coercive. In matrix notation, this means that the coefficient matrix of the discrete problem has a Schur complement that has a positive definite symmetric part. In the context of local refinement, this Schur complement is obtained by first solving the original problem discretized by an initial uniform coarse grid. Hence, this Schur complement is related to the unknowns corresponding to the grids in the subdomains where we have introduced

*Received by the editors February 12, 1992; accepted for publication (in revised form) March 16, 1993. This research has been supported in part by the National Science Foundation grant INT-8914472.

†Institute for Scientific Computation, Texas A&M University, 326 Teague Research Center, College Station, Texas 77843–3404 (ewing@ewing.tamu.edu).

‡Center of Informatics and Computer Technology, Bulgarian Academy of Sciences,"Acad. G. Bontchev" str., block 25 A, 1113 Sofia, Bulgaria (zara@grz08u.unileoben.ac.at).

§Department of Mathematics, University of California at Los Angeles, 405 Hilgard Avenue, Los Angeles, California 90024–1555 (panayot@math.ucla.edu).

local refinement. This is again a problem with similar discretization to the original problem, but now in uniformly refined subregions. The difference with the result from Vassilevski [26] is in the preconditioner derived for the reduced Schur complement that is based on preconditioners for the principal symmetric and coercive part of the original bilinear form restricted to the subdomains where we have uniformly refined grid. This means that we do not have to precondition the symmetric positive definite principal part of the original matrix of the problem in the entire domain with complicated grid with local refinement (as the method from Vassilevski [26] would have required), but only its restriction to the subdomains with uniformly refined grid.

In the present paper, we prove that the reduced problem can be preconditioned in a generalized conjugate gradient (GCG) method by any preconditioner for the symmetric and coercive principal part of the bilinear form restricted to the subdomains where we have introduced local refinement. By exploiting the technique from Axelsson and Vassilevski [6] (see also [5]), we can allow the coarse-grid solvers to be somewhat inexact; but this is not discussed further in this paper.

The outline of the paper is as follows. In §2, we introduce the problem and formulate the preconditioner. The convergence properties of the preconditioner are studied in §3. Finally, in §4, a number of numerical tests that illustrate our theory are presented.

2. Construction of the preconditioner. Let Ω be a given two-dimensional (2D) polygon or a three-dimensional (3D) polytope, and let Γ_N be a given part of the boundary $\partial\Omega$ on which Neumann boundary conditions are imposed. We assume that the Dirichlet portion, $\Gamma_D = \partial\Omega \backslash \Gamma_N$, is a nontrivial part of $\partial\Omega$. We denote by $W_2^j(\Omega)$ the standard L^2-based Sobolev spaces on Ω.

We consider the second-order boundary value problem in a variational setting. Given $f \in L^2(\Omega)$ and $g_N \in W_2^{-1/2}(\Gamma_N)$, find $u \in W_2^1(\Omega)$ such that

$$
\begin{aligned}
a(u,\phi) &\equiv \int_\Omega \left[\sum_{i,j} k_{i,j}(x) \frac{\partial u}{\partial x_i} \frac{\partial \phi}{\partial x_j} + (\mathbf{b}\cdot\nabla u)\phi + c_0 u\phi \right] dx \\
&= \int_\Omega f\phi dx + \int_{\Gamma_N} g_N \phi d\Gamma \quad \text{for all } \phi \in W_2^1(\Omega), \qquad \phi = 0 \text{ on } \Gamma_D,
\end{aligned} \tag{2.1}
$$

and $u = 0$ on Γ_D.

The bounded measurable coefficients $k_{i,j}(x)$ define a symmetric matrix that is assumed to be uniformly positive definitive for $x \in \bar{\Omega}$. We also assume that c_0 is a measurable and bounded function and that the vector field $\mathbf{b}$ is sufficiently smooth.

The principal symmetric and coercive part of $a(\cdot,\cdot)$ is

$$
a_0(u,\phi) = \int_\Omega \sum_{i,j} k_{i,j} \frac{\partial u}{\partial x_i} \frac{\partial \phi}{\partial x_j} dx. \tag{2.2}
$$

We discretize the boundary value problem above on an initial coarse mesh $\tilde{\omega}$. For definiteness, we consider the case of triangular grids for a polygonal domain Ω. The finite element discretization space $\tilde{V} \subset W_2^1(\Omega)$ is assumed to be spanned by piecewise linear functions that are continuous in Ω and vanish on Γ_D. The nodes (vertices of triangles that do not lie on Γ_D) define the initial (coarse) grid $\tilde{\omega}$. We denote by $\tilde{h}$ the discretization parameter on $\tilde{\omega}$. The discrete finite element problem reads: find $u_{\tilde{h}} \in \tilde{V}$ such that for all $\phi \in \tilde{V}$, we have

$$
a(u_{\tilde{h}}, \phi) = \int_\Omega f\phi dx + \int_{\Gamma_N} g_N \phi d\Gamma.
$$

Let $\{\tilde{\phi}_i\}_{i=1}^{\tilde{n}}$ be a standard nodal basis for $\tilde{V}$. Then we can compute the corresponding stiffness matrix $\tilde{A}$,

$$\tilde{A} = \left\{a\left(\tilde{\phi}_j, \tilde{\phi}_i\right)\right\}_{i,j=1}^{\tilde{n}}$$

and formulate the finite element problem as a linear algebraic system

$$\tilde{A}\tilde{\mathbf{x}} = \tilde{\mathbf{b}},$$

where $\tilde{\mathbf{x}}$ is the coefficient vector of $u_{\tilde{h}}$ expanded in terms of the basis $\{\tilde{\phi}_i\}_{i=1}^{\tilde{n}}$ and

$$\tilde{\mathbf{b}} = \left(\int_\Omega f\tilde{\phi}_i dx + \int_{\Gamma_N} g_N\tilde{\phi}_i d\Gamma\right)_{i=1}^{\tilde{n}}.$$

In general, we may not be satisfied by the solution $u_{\tilde{h}}$ on the coarse mesh $\tilde{\omega}$. Thus, after a proper a posteriori analysis, one may decide to refine the mesh. This process can be repeated successively. In practice, the refinement will usually take place only in certain subregions where the currently computed approximate solution has a large gradient. We study the following model situation. Let $\Omega_1 \subset \Omega$ be a subregion where we introduce a finer mesh ω_1. We assume that Ω_1 is covered by coarse-grid elements. The refinement is performed by subdividing any coarse-grid element in Ω_1 into a fixed number of congruent ones. At the interfaces that arise between the refined and unrefined elements, we have to introduce so-called "slave" nodes. The values of the functions at these nodes in the corresponding fine finite element space $V \supset \tilde{V}$ are obtained by interpolation (linear interpolation between the vertices of the adjacent coarse-grid element that is not refined). This means that the values at these nodes are not degrees of freedom. We denote the finer grid (also called the composite grid) obtained in this manner by ω. We have $\omega_1 = \Omega_1 \cap \omega$; that is, ω_1 consists of all the nodes (old nodes and those added in the refinement) in Ω_1. The fine-grid space can also be defined as follows:

$$V = \tilde{V} + V_0^{(1)},$$

where $V_0^{(1)}$ is the standard finite element space of piecewise linear functions in Ω_1 that vanish on the interfaces between Ω_1 and the unrefined part of Ω. Let $\{\phi_i\}_{i=1}^n$ be the standard nodal basis in V. Then one can consider the following hierarchical basis in V:

$$\left\{\hat{\phi}_i\right\}_{i=1}^n \equiv \left\{\tilde{\phi}_i\right\}_{i=1}^{\tilde{n}} \cup \{\phi_i\}_{i=\tilde{n}+1}^n.$$

We compute the stiffness matrices with respect to the above hierarchical basis, obtaining

$$A = \left\{a\left(\hat{\phi}_j, \hat{\phi}_i\right)\right\}_{i,j=1}^n, \qquad A_0 = \left\{a_0\left(\hat{\phi}_j, \hat{\phi}_i\right)\right\}_{i,j=1}^n.$$

Our ultimate requirement is the following:

(I) Problems on the standard, possibly uniform, grids can be solved efficiently; for example, there exists a standard efficient software code for such solution processes.

In our case, such grids will be $\tilde{\omega}$, ω_1, and any grid obtained by uniform refinement of any of their regular parts. For example, we can think of rectangular subgrids. We now partition the composite-grid matrices into the following natural two-by-two block forms:

$$A = \begin{pmatrix} \tilde{A} & G \\ F & H \end{pmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega\backslash\tilde{\omega}, \end{matrix} \qquad A_0 = \begin{pmatrix} \tilde{A}_0 & G_0 \\ F_0 & H_0 \end{pmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega\backslash\tilde{\omega}. \end{matrix}$$

We note that, since we use hierarchical basis functions, the first block on the diagonal of A or (A_0) is just $\tilde{A}$ (respectively, $\tilde{A}_0$), the coarse-grid matrix. We will need the Schur complements

$$W = H - F\tilde{A}^{-1}G \quad \text{and} \quad W_0 = H_0 - F_0\tilde{A}_0^{-1}G_0. \tag{2.3}$$

Our next goal will be to construct a preconditioner for W, the reduced Schur form of A. To do this we will need the stiffness matrix

$$A_0^{(1)} = \left\{a_{0,\Omega_1}\left(\hat{\phi}_j, \hat{\phi}_i\right)\right\}_{x_i, x_j \in \omega_1}.$$

This is the stiffness matrix corresponding to the symmetric and coercive bilinear form $a_0(\cdot,\cdot)$ restricted to Ω_1; it is defined in the subregion Ω_1 using the finite element space $V_0^{(1)}$ of functions that vanish on the interfaces between Ω_1 and the unrefined part of Ω. We similarly have the hierarchical block form

$$A_0^{(1)} = \begin{pmatrix} \tilde{A}_0^{(1)} & G_0^{(1)} \\ F_0^{(1)} & H_0^{(1)} \end{pmatrix} \begin{matrix} \}\tilde{\omega}_1, \\ \}\omega_1 \backslash \tilde{\omega}_1. \end{matrix}$$

Here $\tilde{\omega}_1$ and ω_1 are the coarse and fine grids in Ω_1, respectively. Note that $\omega_1 \backslash \tilde{\omega}_1 = \omega \backslash \tilde{\omega}$. We note that, in the computation, the hierarchical basis stiffness matrices are used only implicitly, cf. Yserentant [32] or Bank, Dupont, and Yserentant [8].

We also need the Schur complement

$$W_0^{(1)} = H_0^{(1)} - F_0^{(1)}\tilde{A}_0^{(1)^{-1}}G_0^{(1)}. \tag{2.4}$$

Note that $W_0^{(1)}$ is symmetric and positive definite.

Consider now the composite-grid problem

$$A\mathbf{x} = \mathbf{b},$$

where

$$\mathbf{x} = \begin{bmatrix} \tilde{\mathbf{x}} \\ \mathbf{x}_2 \end{bmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega \backslash \tilde{\omega}, \end{matrix} \qquad \mathbf{b} = \begin{bmatrix} \tilde{\mathbf{b}} \\ \mathbf{b}_2 \end{bmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega \backslash \tilde{\omega}, \end{matrix}$$

which, after a coarse-grid solution, is reduced to

$$W\mathbf{x}_2 = \tilde{\mathbf{b}}_2 = \mathbf{b}_2 - F\tilde{A}^{-1}\tilde{\mathbf{b}}. \tag{2.5}$$

We will precondition W in a GCG method, e.g., the method from Axelsson [2] or the generalized minimal residual (GMRES) method from Saad and Schultz [24]. The construction of the preconditioner $Z^{(1)}$ for W is given below.

Let $M^{(1)}$ be a symmetric positive definite (s.p.d.) preconditioner of $A_0^{(1)}$ such that, for some positive constants d_0 and d_1, we have

$$d_0\mathbf{v}^{(1)^T}M^{(1)}\mathbf{v}^{(1)} \le \mathbf{v}^{(1)^T}A_0^{(1)}\mathbf{v}^{(1)} \le d_1\mathbf{v}^{(1)^T}M^{(1)}\mathbf{v}^{(1)} \tag{2.6}$$

for all $\mathbf{v}^{(1)}$ corresponding to the nodes in ω_1. Then $Z^{(1)}$ is defined through the relation

$$M^{(1)^{-1}} = \begin{bmatrix} * & * \\ * & Z^{(1)^{-1}} \end{bmatrix} \begin{matrix} \}\tilde{\omega}_1, \\ \}\omega_1 \backslash \tilde{\omega}_1. \end{matrix} \tag{2.7}$$

Thus, to solve the system

$$Z^{(1)}\mathbf{v}_2^{(1)} = \mathbf{b}_2^{(1)},$$

we solve the larger system

$$M^{(1)}\mathbf{v}^{(1)} = \begin{bmatrix} 0 \\ \mathbf{b}_2^{(1)} \end{bmatrix} \begin{matrix} \}\tilde{\omega}_1, \\ \}\omega_1\backslash\tilde{\omega}_1, \end{matrix}$$

and then $\mathbf{v}_2^{(1)} = Z^{(1)^{-1}}\mathbf{b}_2^{(1)}$ is the second block component of $\mathbf{v}^{(1)}$,

$$\mathbf{v}^{(1)} = \begin{bmatrix} * \\ \mathbf{v}_2^{(1)} \end{bmatrix} \begin{matrix} \}\tilde{\omega}_1, \\ \}\omega_1\backslash\tilde{\omega}_1. \end{matrix}$$

We recall that ω_1 is a standard, uniform grid. We see that solving systems with the preconditioner $Z^{(1)}$ is based on solving systems with the preconditioner $M^{(1)}$ for $A_0^{(1)}$ on a standard, uniform grid; we assumed in (I) that this can be efficiently realized. Also note that, to compute the actions of the Schur complement W, we need to solve a coarse-grid problem with $\tilde{A}$, again on a standard, uniform grid.

3. Convergence properties of the preconditioner. At first we study the properties of the Schur complement W given in (2.3). We use the notation that $|\cdot|_{s,\Omega}$ and $\|\cdot\|_{s,\Omega}$ denote the seminorms in the Sobolev space $W_2^s(\Omega)$. We need the following result, which is the essence of many iterative methods for nonsymmetric and/or indefinite elliptic problems; cf., e.g., Mandel [18] for multigrid methods for nonsymmetric problems, Yserentant [33] for hierarchical basis methods for nonsymmetric and indefinite problems, Cai and Widlund [11] for certain additive domain decomposition methods, and the general approaches developed in Xu and Cai [31] for additive preconditioners, and Xu [30] for his multiplicative version. This result was also used in Vassilevski [26] where iterative methods in a subspace (orthogonal to a coarse-grid finite element space) were proposed.

LEMMA 3.1. *Assume the following approximation property for the coarse-grid finite element space $\tilde{V}$: given $f \in L^2(\Omega)$, consider the adjoint problem: find $v \in W_2^1(\Omega)$, $v = 0$ on Γ_D, such that*

$$a(\phi, v) = (f, v) \qquad \text{for all } \phi \in W_2^1(\Omega), \quad \phi = 0 \quad \text{on } \Gamma_D. \tag{3.1}$$

We make the assumption that for some accuracy level $\epsilon > 0$, there exists a function $\tilde{\phi} \in \tilde{V}$ such that

$$\left|v - \tilde{\phi}\right|_{1,\Omega} \leq \epsilon |f|_{0,\Omega}. \tag{3.2}$$

Consider the space

$$H = \left\{ w \in W_2^1(\Omega),\ w = 0 \text{ on } \Gamma_D \text{ and } a\left(w, \tilde{\phi}\right) = 0 \quad \text{for all } \tilde{\phi} \in \tilde{V} \right\}.$$

Then, the following properties of the bilinear form $a(\cdot, \cdot)$ hold:

$$a(w, w) \geq (1 - 0(\epsilon))a_0(w, w) \quad \text{for all } w \in H$$

and

$$a(v, w) \leq (1 + 0(\epsilon))\sqrt{a_0(v, v)}\sqrt{a_0(w, w)} \quad \text{for all } w \in H,$$

and all $v \in W_2^1(\Omega)$, $v = 0$ on Γ_D.

Proof. As already mentioned, this result can be found in one form or another in many papers. In the present form (which is suitable for our application) it was proved in Vassilevski [26]. For completeness we provide the proof here also.

For simplicity, we assume that $\Gamma_D = \partial\Omega$. We follow the standard method of "duality argument" (due to Aubin [1] and Nitsche [23], and used in Schatz [25], Mandel [18], Yserentant [33] and, more recently, in Cai and Widlund [11], Vassilevski [26], Xu and Cai [31], and Xu [30]). If v is the solution of the corresponding adjoint problem (3.1), and for any $w \in H$, we have

$$\begin{aligned}(w, f) &= a(w, v)\\ &= a\left(w, v - \tilde{\phi}\right) \quad \text{for all } \tilde{\phi} \in \tilde{V}\\ &= a_0\left(w, v - \tilde{\phi}\right) + \int_\Omega (\mathbf{b}\cdot\nabla w + c_0 w)\left(v - \tilde{\phi}\right)\\ &\le a_0\left(w, v - \tilde{\phi}\right) + c\|w\|_1 \left|v - \tilde{\phi}\right|_0\\ &\le c|w|_1 \left|v - \tilde{\phi}\right|_1,\end{aligned}$$

where we have used the boundedness of the vector field $\mathbf{b}$, the coefficient matrix $[k_{i,j}]$, and c_0. Since $\tilde{\phi} \in \tilde{V}$ is arbitrary, using (3.2), we obtain,

$$(w, f) \le c|w|_1 \inf\left\{\left|v - \tilde{\phi}\right|_1,\ \tilde{\phi} \in \tilde{V}\right\} \le c\epsilon|w|_1\,|f|_0.$$

Thus for $f = w$, we obtain

$$|w|_0 \le c\epsilon|w|_1 \le c\epsilon\sqrt{a_0(w, w)} \tag{3.3}$$

by using the coercivity of $a_0(\cdot,\cdot)$, the principal symmetric part of the bilinear form $a(\cdot,\cdot)$. Then, using integration by parts ($\mathbf{b}$ has been assumed to be sufficiently smooth) and (3.3), we see that

$$\begin{aligned}a(w, w) &= a_0(w, w) + \int_\Omega \left(c_0 - \frac{1}{2}\mathrm{div}\mathbf{b}\right) w^2\\ &\ge a_0(w, w) - \left|c_0 - \frac{1}{2}\mathrm{div}\mathbf{b}\right|_\infty |w|_0^2\\ &\ge \left[1 - O(\epsilon^2)\right] a_0(w, w),\end{aligned}$$

which is the first desired inequality. The second inequality is again obtained through integration by parts. We have

$$\begin{aligned}a(w, v) &= a_0(w, v) + \int_\Omega [(\mathbf{b}\cdot\nabla w)v + c_0 wv]\\ &= a_0(w, v) - \int_\Omega w\ \mathrm{div}(\mathbf{b}v) + \int_\Omega c_0 wv\\ &\le a_0(w, v) + c|w|_0\,|v|_1\\ &\le [1 + 0(\epsilon)]\sqrt{a_0(w, w)}\sqrt{a_0(v, v)},\end{aligned}$$

where we have used (3.3), the coercivity of $a_0(\cdot,\cdot)$, and the Schwarz inequality. □

Consider now the hierarchical block-matrix forms of A and A_0:

$$A = \begin{bmatrix} \tilde{A} & G \\ F & H \end{bmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega\backslash\tilde{\omega}, \end{matrix} \quad \text{and} \quad A_0 = \begin{bmatrix} \tilde{A}_0 & G_0 \\ F_0 & H_0 \end{bmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega\backslash\tilde{\omega}. \end{matrix}$$

As already noted, the first blocks equal the corresponding coarse-grid stiffness matrices since we use hierarchical basis functions. The blocks $\tilde{A}$ and $\tilde{A}_0$ are the stiffness matrices at the initial (coarse) discretization level.

The coefficient vector of a function $\tilde{v} \in \tilde{V} \subset V$ with respect to the hierarchical basis in V has the block form,

$$\begin{bmatrix} \tilde{\mathbf{v}} \\ 0 \end{bmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega\backslash\tilde{\omega}. \end{matrix}$$

Then, if $w \in V$ has a coefficient vector

$$\mathbf{w} = \begin{bmatrix} \tilde{\mathbf{w}} \\ \mathbf{w}_2 \end{bmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega\backslash\tilde{\omega}, \end{matrix}$$

and

$$a(w, \tilde{v}) = 0 \quad \text{for all } \tilde{v} \in \tilde{V},$$

we have

$$\begin{bmatrix} \tilde{\mathbf{v}} \\ 0 \end{bmatrix}^T A \begin{bmatrix} \tilde{\mathbf{w}} \\ \mathbf{w}_2 \end{bmatrix} = 0.$$

Thus

$$\tilde{\mathbf{v}}^T \left(\tilde{A}\tilde{\mathbf{w}} + G\mathbf{w}_2 \right) = 0 \quad \text{for all } \tilde{\mathbf{v}}.$$

This implies that

$$\tilde{\mathbf{A}}\tilde{\mathbf{w}} + G\mathbf{w}_2 = 0.$$

Having this in mind, we see that the space

$$H = \left\{ w \in V : a\left(w, \tilde{\phi}\right) = 0 \quad \text{for all } \tilde{\phi} \in \tilde{V} \right\}$$

in matrix-vector form can be written as

$$H = \left\{ \mathbf{w} : \tilde{\mathbf{A}}\tilde{\mathbf{w}} + G\mathbf{w}_2 = 0 \right\}.$$

Next we state the following main result from Vassilevski [26].

LEMMA 3.2. *There exist two positive constants* $\gamma_1 = 1 - O(\epsilon)$ *and* $\gamma_2 = 1 + O(\epsilon)$ *such that, under the approximation property stated in Lemma* 3.1, *the following spectral equivalence relations between the Schur complements* W *of* A *and* $W_0 = H_0 - F_0\tilde{A}_0^{-1}G_0$ *of* A_0 *hold*:

$$\mathbf{v}_2^T W \mathbf{w}_2 \geq \gamma_1 \mathbf{v}_2^T W_0 \mathbf{v}_2 \quad \textit{for all } \mathbf{v}_2 \in \mathbb{R}^{\tilde{n}}$$

and

$$\mathbf{v}_2^T W \mathbf{w}_2 \le \gamma_2 \left(\mathbf{v}_2^T W_0 \mathbf{v}_2\right)^{1/2} \left(\mathbf{w}_2^T W_0 \mathbf{w}_2\right)^{1/2} \quad \textit{for all } \mathbf{v}_2, \mathbf{w}_2 \in \mathbb{R}^{\tilde{n}}.$$

Proof. From Lemma 3.1, we have

$$\mathbf{v}^T A \mathbf{v} \ge \gamma_1 \mathbf{v}^T A_0 \mathbf{v} \tag{3.4}$$

for any $\mathbf{v} = \begin{bmatrix} \tilde{\mathbf{v}} \\ \mathbf{v}_2 \end{bmatrix}_{\omega\setminus\tilde{\omega}}^{\tilde{\omega}}$ in the subspace H satisfying $\tilde{A}\tilde{\mathbf{v}} + G\mathbf{v}_2 = 0$; here $\gamma_1 = 1 - O(\epsilon)$. Since $A\mathbf{v} = \begin{bmatrix} 0 \\ W_{\mathbf{v}_2} \end{bmatrix}$, the inequality above implies that

$$\mathbf{v}_2^T W \mathbf{v}_2 \ge \gamma_1 \inf_{\tilde{\mathbf{v}}} \mathbf{v}^T A_0 \mathbf{v} = \gamma_1 \mathbf{v}_2^T W_0 \mathbf{v}_2.$$

From Lemma 3.1, we also have that

$$\mathbf{v}^T A \mathbf{w} \le \hat{\gamma}_2 \left(\mathbf{v}^T A_0 \mathbf{v}\right)^{1/2} \left(\mathbf{w}^T A_0 \mathbf{w}\right)^{1/2},$$

where $\hat{\gamma}_2 = 1 + O(\epsilon)$, for arbitrary vectors $\mathbf{v} = \begin{bmatrix} \tilde{\mathbf{v}} \\ \mathbf{v}_2 \end{bmatrix}\Big\}_{\omega\setminus\tilde{\omega}}^{\tilde{\omega}}$ and all $\mathbf{w} = \begin{bmatrix} \tilde{\mathbf{w}} \\ \mathbf{w}_2 \end{bmatrix}\Big\}_{\omega\setminus\tilde{\omega}}^{\tilde{\omega}}$ from the subspace H, satisfying $\tilde{A}\tilde{\mathbf{w}} + G\mathbf{w}_2 = 0$. Then we have

$$\begin{aligned}
\mathbf{v}_2^T W \mathbf{w}_2 &= \mathbf{v}^T A \mathbf{w} \\
&\le \hat{\gamma}_2 \left(\inf_{\tilde{\mathbf{v}}} \mathbf{v}^T A_0 \mathbf{v}\right)^{1/2} \left(\mathbf{w}^T A_0 \mathbf{w}\right)^{1/2} \\
&\le \hat{\gamma}_2 \left(\mathbf{v}_2^T W_0 \mathbf{v}_2\right)^{1/2} \left(\mathbf{w}^T A_0 \mathbf{w}\right)^{1/2} \\
&\le \hat{\gamma}_2 / \sqrt{\gamma_1} \left(\mathbf{v}_2^T W_0 \mathbf{v}_2\right)^{1/2} \left(\mathbf{w}_2^T W \mathbf{w}_2\right)^{1/2}
\end{aligned}$$

from (3.4). We set $\mathbf{v}_2 = \mathbf{w}_2$ in this inequality, and thus obtain

$$\left(\mathbf{w}_2^T W \mathbf{w}_2\right)^{1/2} \le \hat{\gamma}_2 / \sqrt{\gamma_1} \left(\mathbf{w}_2^T W_0 \mathbf{w}_2\right)^{1/2}.$$

Substituting the last inequality in the preceding one, we get

$$\mathbf{v}_2^T W \mathbf{w}_2 \le \hat{\gamma}_2^2 / \gamma_1 \left(\mathbf{v}_2^T W_0 \mathbf{v}_2\right)^{1/2} \left(\mathbf{w}_2^T W_0 \mathbf{w}_2\right)^{1/2}.$$

Denoting $\gamma_2 = \hat{\gamma}_2^2 / \gamma_1 = 1 + O(\epsilon)$, we thus obtain the second desired inequality. □

We next need the following result.

LEMMA 3.3 (Strengthened Cauchy Inequality, cf. Axelsson and Gustafsson [3] or Bank and Dupont [7]). *There exists a constant* $\gamma \in (0, 1)$ *such that*

$$a_0\left(\phi, \tilde{\phi}\right) \le \gamma \left(a_0(\phi, \phi)\right)^{1/2} \left(a_0\left(\tilde{\phi}, \tilde{\phi}\right)\right)^{1/2}$$

for all $\tilde{\phi} \in \tilde{V}$ *and all* $\phi \in V$ *that satisfy* $\phi|_{\tilde{\omega}} = 0$. *In matrix notation, we have*

$$\mathbf{v}_2^T F_0 \tilde{\mathbf{v}} \le \gamma \left(\mathbf{v}_2^T H_0 \mathbf{v}_2\right)^{1/2} \left(\tilde{\mathbf{v}}^T \tilde{A}_0 \tilde{\mathbf{v}}\right)^{1/2}$$

for all vectors $\mathbf{v}_2 \in \mathbb{R}^{n-\tilde{n}}$, $\tilde{\mathbf{v}} \in \mathbb{R}^{\tilde{n}}$. *In general, the constant* γ *depends only on the shape of the elements, also on* $\tilde{h}/h$ *(the ratio of the coarse grid over the fine grid) and on the local ellipticity constant* σ *defined as follows*:

$$\sigma = \sup_{T \in \tilde{\mathcal{J}}} \frac{\mu_2(T)}{\mu_1(T)};$$

here $\mu_1(T)$ *and* $\mu_2(T)$ *are the bounds of the expression*

$$\mu_1(T)|\xi|^2 \leq \sum_{i,j} k_{i,j}(x)\xi_i\xi_j \leq \mu_2(T)|\xi|^2 \quad \text{for all } x \in T \in \tilde{\mathcal{J}}.$$

In particular, this means that γ *remains bounded away from unity, independent of the possible jumps of the coefficients if these occur only across edges of elements from* $\tilde{\mathcal{J}}$ *(the set of initial coarse triangles). The asymptotic behavior of* γ *in terms of* $\tilde{h}/h$ *is (cf., e.g., Vassilevski* [27])

$$\gamma^2 = 1 - \begin{cases} C\dfrac{1}{1+\log\frac{\tilde{h}}{h}} & \text{for 2D domain } \Omega, \\ C\frac{h}{\tilde{h}}, & \text{for 3D domain } \Omega. \end{cases}$$

We assume throughout this paper that the aspect ratio, $\tilde{h}/h$, *is bounded.*

A corollary from Lemma 3.3 is the following lemma (cf. Axelsson and Gustafsson [3]).

LEMMA 3.4. *The following inequality holds*:

$$\mathbf{v}_2^T H_0 \mathbf{v}_2 \leq \frac{1}{1-\gamma^2}\mathbf{v}^T A_0 \mathbf{v} \quad \text{for all } \mathbf{v} = \begin{bmatrix} \tilde{\mathbf{v}} \\ \mathbf{v}_2 \end{bmatrix} \begin{matrix} \}\tilde{\omega}, \\ \}\omega\backslash\tilde{\omega}. \end{matrix}$$

Proof. We have, using the strengthened Cauchy inequality,

$$\begin{aligned} \mathbf{v}^T A_0 \mathbf{v} &= \mathbf{v}_2^T H_0 \mathbf{v}_2 + \tilde{\mathbf{v}}^T \tilde{\mathbf{A}}_0 \tilde{\mathbf{v}} + 2\mathbf{v}_2^T F_0 \tilde{\mathbf{v}} \\ &\geq \mathbf{v}_2^T H_0 \mathbf{v}_2 + \tilde{\mathbf{v}}^T \tilde{\mathbf{A}}_0 \tilde{\mathbf{v}} - 2\gamma \left(\mathbf{v}_2^T H_0 \mathbf{v}_2\right)^{1/2} \left(\tilde{\mathbf{v}}^T \tilde{\mathbf{A}}_0 \tilde{\mathbf{v}}\right)^{1/2} \\ &= (1-\gamma^2)\mathbf{v}_2^T H_0 \mathbf{v}_2 + \left[\gamma\sqrt{\mathbf{v}_2^T H_0 \mathbf{v}_2} - \sqrt{\tilde{\mathbf{v}}^T \tilde{\mathbf{A}}_0 \tilde{\mathbf{v}}}\right]^2 \\ &\geq (1-\gamma^2)\mathbf{v}_2^T H_0 \mathbf{v}_2, \end{aligned}$$

which is the desired inequality. □

We also need the following basic lemma (cf. Ewing and Vassilevski [17]).

LEMMA 3.5. *The following inequalities between the Schur complement* $W_0^{(1)} = H_0^{(1)} - F_0^{(1)} \tilde{A}_0^{(1)^{-1}} G_0^{(1)}$ *of* $A_0^{(1)}$ *(see* (2.4)*), the Schur complement* W_0 *of* A_0 *(see* (2.3)*), and the block* H_0 *of* A_0 *hold*:

$$\mathbf{v}_2^T W_0 \mathbf{v}_2 \leq \mathbf{v}_2^T W_0^{(1)} \mathbf{v}_2 \leq \mathbf{v}_2^T H_0 \mathbf{v}_2 \leq \frac{1}{1-\gamma^2}\mathbf{v}_2^T W_0 \mathbf{v}_2 \quad \text{for all } \mathbf{v}_2 \in \mathbb{R}^{n-\tilde{n}}.$$

Proof. By a main property of Schur complements of s.p.d. matrices, we have,

$$\begin{aligned}
\mathbf{v}_2^T W_0 \mathbf{v}_2 &= \inf_{\tilde{\mathbf{v}}} \begin{bmatrix} \tilde{\mathbf{v}} \\ \mathbf{v}_2 \end{bmatrix}^T A_0 \begin{bmatrix} \tilde{\mathbf{v}} \\ \mathbf{v}_2 \end{bmatrix} \\
&\le \inf_{\tilde{\mathbf{v}}} \left\{ \begin{bmatrix} \tilde{\mathbf{v}} \\ \mathbf{v}_2 \end{bmatrix}^T A_0 \begin{bmatrix} \tilde{\mathbf{v}} \\ \mathbf{v}_2 \end{bmatrix} : \tilde{\mathbf{v}} = \begin{bmatrix} \tilde{\mathbf{v}}^{(1)} \\ 0 \end{bmatrix} \begin{matrix} \}\tilde{\omega}_1 \\ \}\tilde{\omega}\backslash\tilde{\omega}_1 \end{matrix} \right\} \\
&= \inf_{\tilde{\mathbf{v}}^{(1)}} \begin{bmatrix} \tilde{\mathbf{v}}^{(1)} \\ \mathbf{v}_2 \end{bmatrix}^T A_0^{(1)} \begin{bmatrix} \tilde{\mathbf{v}}^{(1)} \\ \mathbf{v}_2 \end{bmatrix} \\
&= \mathbf{v}_2^T W_0^{(1)} \mathbf{v}_2.
\end{aligned}$$

We have, by Lemma 3.4,

$$\mathbf{v}_2^T H_0 \mathbf{v}_2 \le \frac{1}{1-\gamma^2} \mathbf{v}^T A_0 \mathbf{v},$$

and hence

$$\mathbf{v}_2^T H_0 \mathbf{v}_2 \le \frac{1}{1-\gamma^2} \inf_{\tilde{\mathbf{v}}} \mathbf{v}^T A_0 \mathbf{v} = \frac{1}{1-\gamma^2} \mathbf{v}_2^T W_0 \mathbf{v}_2.$$

Finally, noting that $H_0 = H_0^{(1)}$ and

$$\mathbf{v}_2^T W_0^{(1)} \mathbf{v}_2 = \mathbf{v}_2^T \left(H_0^{(1)} - F_0^{(1)} \tilde{A}_0^{(1)^{-1}} G_0^{(1)} \right) \mathbf{v}_2 \le \mathbf{v}_2^T H_0 \mathbf{v}_2,$$

the proof is complete. □

Now we can prove our main result.

THEOREM 3.6. *The preconditioning matrix $Z^{(1)}$ induced by the preconditioner $M^{(1)}$ for $A_0^{(1)}$ (see (2.6) and (2.7)), is spectrally equivalent to the reduced Schur matrix W of A, and the following spectral equivalence relations hold:*

$$\begin{aligned}
&\mathbf{v}_2^T W \mathbf{w}_2 \le d_1 \gamma_2 \left(\mathbf{v}_2^T Z^{(1)} \mathbf{v}_2 \right)^{1/2} \left(\mathbf{w}_2^T Z^{(1)} \mathbf{w}_2 \right)^{1/2} \quad \text{for all } \mathbf{v}_2, \mathbf{w}_2 \in \mathbb{R}^{n-\tilde{n}}, \\
&\mathbf{v}_2^T W \mathbf{v}_2 \ge (1-\gamma^2) d_0 \gamma_1 \mathbf{v}_2^T Z^{(1)} \mathbf{v}_2 \quad \text{for all } \mathbf{v}_2 \in \mathbb{R}^{n-\tilde{n}}.
\end{aligned}$$

The constants γ_1 and γ_2 are from Lemma 3.2; *the constants d_0 and d_1 are from* (2.6), *and $\gamma \in (0, 1)$ is the constant from the strengthened Cauchy inequality from Lemma* 3.3.

Proof. We have, from (2.6), noting that $Z^{(1)}$ is a Schur complement of $M^{(1)}$,

$$d_0 \mathbf{v}_2^T Z^{(1)} \mathbf{v}_2 \le \mathbf{v}_2^T W_0^{(1)} \mathbf{v}_2 \le d_1 \mathbf{v}_2^T Z^{(1)} \mathbf{v}_2 \quad \text{for all } \mathbf{v}_2 \in \mathbb{R}^{n-\tilde{n}}.$$

Then the proof simply follows from Lemma 3.2, the above inequalities, and Lemma 3.5. □

COROLLARY 3.7. *The generalized conjugate gradient methods, e.g., those from Axelsson* [2], *or the* GMRES method *from Saad and Schultz* [24], *applied to solve the reduced system* (2.5) *with W as a coefficient matrix using $Z^{(1)}$ as a preconditioning matrix, have an asymptotic convergence factor that depends only on the constant $(\gamma_2/\gamma_1)(1-\gamma^2)^{-1}\kappa$, where $\kappa = d_1/d_0$ is the condition number of $M^{(1)^{-1}} A_0^{(1)}$.*

Remark 1. The approximation property of the coarse finite element space $\tilde{V}$ assumed in Lemma 3.2 is ensured, for example, if the adjoint problem (3.1) has certain (very weak) regularity, say, for some $\alpha > 0$ and any $f \in L^2(\Omega)$ the solution v of (3.1) belongs to $W_2^{1+\alpha}(\Omega)$.

Then we can let $\epsilon = O(\tilde{h}^{\alpha})$ in (3.2), where $\tilde{h}$ is the coarse-grid discretization parameter. We also note that the ratio $\gamma_2/\gamma_1 = 1 + O(\epsilon) = 1 + O(\tilde{h}^{\alpha})$ in this case. Hence, the convergence properties of the preconditioner $Z^{(1)}$ are asymptotically the same (if $\tilde{h}$ is sufficiently small) as if the original problem was assumed symmetric. We also remark that $1/(1-\gamma^2)$ is bounded when $\tilde{h}/h$ is bounded.

4. Numerical experiments. In this section we present numerical results for the bilinear form

$$a(u,\varphi) = \int_{\Omega} k\nabla u \cdot \nabla\varphi dx + \int_{\Omega} (\mathbf{v}\cdot\nabla u)\varphi dx - \lambda \int_{\Omega} u\varphi dx.$$

Here Ω is the unit square. We impose homogeneous Dirichlet boundary conditions on the boundaries $\{x = 0\}$ and $\{y = 0\}$ and homogeneous natural boundary conditions on $\{x = 1\}$ and $\{y = 1\}$.

We test the following problems.

Problem 1 (symmetric and positive definite bilinear form). This case corresponds to the bilinear form $a_0(\cdot,\cdot)$ from (2.2) with $k = 1 + x^2 + y^2$, i.e., $\mathbf{v} = 0$ and $\lambda = 0$.

Problem 2 (nonsymmetric bilinear form). The coefficients are

1. $v_1 = -\cos\theta(x\cos\theta + y\sin\theta)$,
   $v_2 = -\sin\theta(x\cos\theta + y\sin\theta)$,
2. $v_1 = -10\cos\theta(x\cos\theta + y\sin\theta)$,
   $v_2 = -10\sin\theta(x\cos\theta + y\sin\theta)$,

where θ is 45°; in this case, $\lambda = 0$ and $k = 1 + x^2 + y^2$.

Problem 3 (symmetric but possibly indefinite bilinear form). This case corresponds to a Helmholtz-like equation, i.e., $k = 1 + x^2 + y^2$ and $\mathbf{v} = 0$. We test this problem with $\lambda = 10, 50, 500$.

For all problems, the exact solution is

$$u(x,y) = x\left(1 - \frac{1}{2}x\right) y\left(1 - \frac{1}{2}y\right).$$

In the discretization, we use the finite element methods with piecewise linear elements on isosceles right-angled triangles. The results are listed in Tables 1–6 where N_c is the number of nodes in the initial coarse grid coordinate, i.e., the initial mesh size is $h_c = 1/N_c$. The subregion Ω_1 where the local refinement is introduced is chosen to be the northeast 1/16 part of the whole domain Ω and the local refinement is done with parameter $n_0 = h_c/h_f = 2, 4, 8$, where h_f is the grid size in the subdomain Ω_1.

TABLE 1
Iterative convergence results for Problem 1.

$\mathbf{v} = 0$ and $\lambda = 0$

N_c	n_0	iter	arfac
8	2	4	0.126
	4	5	0.186
	8	7	0.255
16	2	4	0.111
	4	5	0.217
	8	7	0.289
32	2	3	0.153
	4	5	0.196
	8	6	0.280

TABLE 2
Iterative convergence results for Problem 2, (1).

$\lambda = 0, \theta = 45°$

N_c	n_0	iter	arfac	$\tilde{A}$	W
8	2	4	0.127	coercive	coercive
	4	5	0.182	coercive	coercive
	8	7	0.255	coercive	coercive
16	2	4	0.111	coercive	coercive
	4	5	0.217	coercive	coercive
	8	7	0.289	coercive	coercive
32	2	3	0.153	coercive	coercive
	4	5	0.196	coercive	coercive
	8	6	0.280	coercive	coercive

TABLE 3
Iterative convergence results for Problem 2, (2).

$\lambda = 0, \theta = 45°$

N_c	n_0	iter	arfac	$\tilde{A}$	W
8	2	4	0.128	coercive	coercive
	4	6	0.213	coerceve	coercive
	8	7	0.273	coercive	coercive
16	2	4	0.113	coercive	coercive
	4	6	0.214	coercive	coercive
	8	7	0.292	coercive	coercive
32	2	3	0.152	coercive	coercive
	4	5	0.201	coercive	coercive
	8	6	0.284	coercive	coercive

TABLE 4
Iterative convergence results for Problem 3.

$\mathbf{v} = 0$ and $\lambda = 10$

N_c	n_0	iter	arfac	$\tilde{A}$	W
8	2	4	0.130	indefinite	coercive
	4	5	0.186	indefinite	coercive
	8	7	0.255	indefinite	coercive
16	2	4	0.112	indefinite	coercive
	4	5	0.217	indefinite	coercive
	8	7	0.289	indefinite	coercive
32	2	3	0.153	indefinite	coercive
	4	5	0.196	indefinite	coercive
	8	6	0.280	indefinite	coercive

TABLE 5
Iterative convergence results for Problem 3.

$\mathbf{v} = 0$ and $\lambda = 50$

N_c	n_0	iter	arfac	$\tilde{A}$	W
8	2	4	0.134	indefinite	coercive
	4	5	0.185	indefinite	coercive
	8	7	0.253	indefinite	coercive
16	2	4	0.115	indefinite	coercive
	4	5	0.217	indefinite	coercive
	8	7	0.288	indefinite	coercive
32	2	3	0.153	indefinite	coercive
	4	5	0.196	indefinite	coercive
	8	6	0.280	indefinite	coercive

TABLE 6
Iterative convergence results for Problem 1.

$\nu = 0$ and $\lambda = 500$

N_c	n_0	iter	arfac	$\tilde{A}$	W
8	2	8	0.211	indefinite	noncoercive
	4	10	0.339	indefinite	noncoercive
	8	11	0.455	indefinite	noncoercive
16	2	5	0.160	indefinite	coercive
	4	7	0.284	indefinite	coercive
	8	8	0.354	indefinite	coercive
32	2	3	0.126	indefinite	coercive
	4	5	0.204	indefinite	coercive
	8	6	0.295	indefinite	coercive

We solve the corresponding reduced discrete problems (2.5) where the right-hand side vector $\mathbf{b}$ corresponds to the exact solution $u(x, y)$ above. We use the GCG method from Axelsson [2] with a stopping criterion

$$\Delta = \mathbf{r}_2^T \mathbf{r}_2 < \epsilon, \qquad \epsilon = 10^{-12},$$

where $\mathbf{r}_2$ is the residual vector and the parameters iter and arfac in the tables are, respectively, the number of iterations and the average reduction factor for reaching the above accuracy, i.e.,

$$\text{arfac} = \left(\sqrt{\frac{\Delta}{\Delta_0}}\right)^{1/\text{iter}},$$

where $\Delta_0 = \mathbf{r}_2^{(0)^T} \mathbf{r}_2^{(0)}$ and $\mathbf{r}_2^{(0)}$ is the initial residual vector.

For problems in the subregion we use the AMLI preconditioner (algebraic multilevel preconditioner) from Axelsson and Vassilevski [4] as modified in Vassilevski [27]. We also indicate in Tables 2–6 whether $\tilde{A}$ and W are positive definite (coercive) or not.

From the numerical results presented, we see that the proposed extension of the iterative refinement methods from Ewing and Vassilevski [16], [17] works as predicted by the theory. If the coarse-grid space $\tilde{V}$ gives more accurate approximation to the considered boundary value problem, i.e., when N_c is larger, we see that the GCG method has better convergence properties in the sense that the number of iterations (iter) decreased for a fixed $n_0 = h_c/h_f = 2, 4, 8$.

For the case in Table 6, for $N_c = 8$, $n_0 = 2, 4, 8$ and $\lambda = 500$, we obtain noncoercive Schur complement W. However, it turns out that the GCG method, after a short starting phase, performs as in the coercive case. This behavior can be explained as in Yserentant [33]. This is due to the fact that the GCG method is also convergent when we have a number of eigenvalues (but not too many) with negative real part. The number of such eigenvalues depends only on λ (if the discretization is sufficiently accurate) and is independent of the preconditioner used.

We also observe from Tables 4, 5, and 6 that the method is not very sensitive with respect to λ. We get about the same number of iterations for $\lambda = 10$, 50, and 500 in the case when the Schur complement W is coercive.

REFERENCES

[1] J. P. AUBIN, *Approximation of Elliptic Boundary–Value Problems*, Wiley-Interscience, New York, 1972.

[2] O. AXELSSON, *A generalized conjugate gradient, least squares method*, Numer. Math., 51 (1987), pp. 209–227.

[3] O. AXELSSON AND I. GUSTAFSSON, *Preconditioning and two-level multigrid methods of arbitrary degree of approximations*, Math. Comp., 40 (1983), pp. 219–242.

[4] O. AXELSSON AND P. S. VASSILEVSKI, *Algebraic multilevel preconditioning methods*, II, SIAM J. Numer. Anal., 27 (1990), pp. 1569–1590.

[5] ———, *A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 625–644.

[6] ———, *Construction of variable-step preconditioners for inner-outer iteration methods*, Proc. IMACS Conf. Iterative Methods, Apr. 1–4, 1991, Brussels, Belgium, Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen, eds., North Holland, 1992, pp. 1–14.

[7] R. BANK AND T. DUPONT, *Analysis of a two-level scheme for solving finite element equations*, Report CNA–159, Center for Numerical Analysis, Univ. of Texas at Austin, 1980.

[8] R. BANK, T. DUPONT, AND H. YSERENTANT, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[9] 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*, Computer Meth. Appl. Mech. Engrg., 67 (1988), pp. 149–159.

[10] J. H. BRAMBLE, J. E. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[11] X.-C. CAI AND O. B. WIDLUND, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Statist. Comput., 13, (1992), pp. 243–258.

[12] M. DRYJA AND O. B. WIDLUND, *Multilevel additive methods for elliptic finite element problems*, Parallel Algorithms for PDE's, Proc. 6th GAMM-Seminar, Kiel, Jan. 16–21, 1990, W. Hackbusch, ed., Vieweg-Verlag, Braunschweig, Wiesbaden, pp. 58–69.

[13] R. E. EWING, R. D. LAZAROV, J. E. PASCIAK, AND P. S. VASSILEVSKI, *Domain decomposition type iterative techniques for parabolic problems on locally refined grids*, SIAM J. Numer. Anal., 30 (1993), pp. 1537–1558.

[14] R. E. EWING, R. D. LAZAROV, AND P. S. VASSILEVSKI, *Local refinement techniques for elliptic problems on cell-centered grids*, III: *Algebraic multilevel BEPS-preconditioners*, Numer. Math., 59 (1991), pp. 431–452.

[15] ———, *Local refinement techniques for elliptic problems on cell-centered grids*, II: *Optimal order two-grid preconditioners*, J. Numer. Linear Algebra Appl., to appear.

[16] R. E. EWING AND P. S. VASSILEVSKI, *Two-level iterative refinement preconditioners*, Proc. 5th Conf. Domain Decomposition Methods for Partial Differential Equations, May 6–8, 1991, Norfolk, VA, Society for Industrial and Applied Mathematics, 1992, pp. 262–270.

[17] ———, *Optimal order iterative refinement preconditioners*, J. Numer. Linear Algebra Appl., to appear.

[18] J. MANDEL, *Multigrid convergence for nonsymmetric, indefinite variational problems and one smoothing step*, Appl. Math. Comput., 19 (1986), pp. 201–216.

[19] J. MANDEL AND S. MCCORMICK, *Iterative solution of elliptic equations with refinement: The two-level case*, Domain Decomposition Methods, T. F. Chan, R. L. Glowinski, J. Periaux, and O. B. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 81–92.

[20] S. MCCORMICK, *Fast adaptive composite grid* (FAC) *methods: Theory for the variational case*, Computing, Suppl., 5 (1984), pp. 115–121.

[21] ———, *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, 1989.

[22] S. MCCORMICK AND J. THOMAS, *The fast adaptive composite grid method* (FAC) *for elliptic boundary value problems*, Math. Comp., 46 (1986), pp. 439–456.

[23] J. NITSCHE, *Ein Kriterium für die Quasi–Opimalität des Ritzchen Verfahren*, Numer. Math., 11 (1968), pp. 346-348.

[24] 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.

[25] A. H. SCHATZ, *An observation concerning Ritz–Galerkin methods with indefinite bilinear forms*, Math. Comp., 28 (1974), pp. 959–962.

[26] P. S. VASSILEVSKI, *Preconditioning nonsymmetric and indefinite finite element matrices*, J. Numer. Linear Alg. Appl., 1 (1992), pp. 59–76.

[27] ———, *Hybrid* V*-cycle algebraic multilevel preconditioners*, Math. Comp., 58 (1992), pp. 489–512.

[28] P. S. VASSILEVSKI, S. I. PETROVA, AND R. D. LAXAROV, *Preconditioning elliptic problems on grids with multilevel local refinement*, Mathematika Balkanica, to appear.

[29] O. B. WIDLUND, *Optimal iterative refinement methods*, Domain Decomposition Methods, T. F. Chan, R. L. Glowinski, J. Periaux, and O. B. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 114–125.

[30] J. XU, *A new class of iterative methods for nonselfadjoint or indefinite problems*, SIAM J. Numer. Anal., 29 (1992), pp. 303–319.

[31] J. XU AND X.-C. CAI, *A preconditioned GMRES method for nonsymmetric or indefinite problems*, Math. Comp., 59 (1992), pp. 311–319.
[32] H. YSERENTANT, *On the multilevel splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.
[33] ———, *On the multilevel splitting of finite element spaces for indefinite elliptic boundary value problems*, SIAM J. Numer. Anal., 23 (1986), pp. 581–595.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 164–171, January 1994

011

FAST BAND-TOEPLITZ PRECONDITIONERS FOR HERMITIAN TOEPLITZ SYSTEMS*

RAYMOND H. CHAN† AND PING TAK PETER TANG‡

Abstract. This paper considers the solutions of Hermitian Toeplitz systems where the Toeplitz matrices are generated by nonnegative functions f. The preconditioned conjugate gradient method with well-known circulant preconditioners fails in the case when f has zeros. This paper employs Toeplitz matrices of fixed bandwidth as preconditioners. Their generating functions g are trigonometric polynomials of fixed degree and are determined by minimizing the maximum relative error $\|(f-g)/f\|_\infty$. It is shown that the condition number of systems preconditioned by the band-Toeplitz matrices are $O(1)$ for f, with or without zeros. When f is positive, the preconditioned systems converge at the same rate as other well-known circulant preconditioned systems. An a priori bound of the number of iterations required for convergence is also given.

Key words. Toeplitz matrix, generating function, preconditioned conjugate gradient method, Chebyshev approximation, Remez algorithm

AMS subject classifications. 65F10, 65F15

1. Introduction.

In this paper, we consider solutions of n-by-n Hermitian Toeplitz systems $A_n x = b$ by the preconditioned conjugate gradient method. The Toeplitz matrices A_n are assumed to be generated by 2π-periodic continuous real-valued functions f defined on $[-\pi, \pi]$, i.e., the entries of A_n are given by the Fourier coefficients of f:

$$[A_n]_{j,k} = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-i(j-k)x}dx \quad \forall 0 \le j,k < n.$$

We emphasize that the generating function f is given in some applications of Toeplitz systems. Typical examples are the kernels of the Wiener–Hopf equations, see Gohberg and Fel'dman [12, p. 82], the spectral density functions in stationary stochastic process, see Grenander and Szegö [14, p. 171], and the point-spread functions in image deblurring, see Oppenheim [16, p. 200].

If the generating function f is positive, the preconditioned conjugate gradient method with circulant preconditioners has proved to be a successful method—the preconditioned systems converge superlinearly when f is smooth, see, for instance, Chan and Strang [3] and Chan and Yeung [5]. However, these circulant preconditioners do not work in general when f has zeros. A specific example is the one-dimensional discrete Laplacian given by the tridiagonal matrix trid$[-1, 2, -1]$. Its generating function is $f(x) = 4\sin^2 x$, which has a zero at $x = 0$. The corresponding Strang circulant preconditioner, see [17], is actually singular. (See also the numerical results in §4 for the performance of the T. Chan [8] circulant preconditioner in the case where f has zeros.)

Recently, Chan [4] proposed using band-Toeplitz matrices $B_{n,\ell}$ as preconditioners for f that has zeros. These preconditioners are constructed by matching their generating function g with f at those zeros of f. It is proved that if the order of the zero of f is 2ℓ, then the condition number $\kappa(A_n)$ of A_n is $O(n^{2\ell})$, whereas $\kappa(B_{n,\ell}^{-1}A_n)$ is $O(1)$. However, when f is positive, the band-Toeplitz preconditioned systems converge much slower than those preconditioned by circulant preconditioners.

Our main aim in this paper is to design band-Toeplitz preconditioners that work when f has zeros and yet their preconditioned systems converge at the same rate as the circulant

*Received by the editors January 30, 1992; accepted for publication (in revised form) March 16, 1993.

†Department of Mathematics, University of Hong Kong, Hong Kong. This author's research was supported in part by Office of Naval Research contract N00014-90-J-1695 and Department of Energy grant DE-FG03-87ER25037.

‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439.

preconditioned systems even when f is positive. Our idea is to increase the bandwidth of the band-Toeplitz preconditioner to get extra degrees of freedom, which enable us not only to match the zeros in f but also to minimize the relative error $||(f-g)/f||_\infty$. The minimizer g is found by a version of the Remez algorithm proposed by Tang [18]. The algorithm also computes the minimum relative error, which ultimately gives an a priori bound on the number of iterations required for convergence.

We note that the band-Toeplitz preconditioner we proposed has bandwidth ℓ that depends only on the order of the zeros of f and is independent of n, the size of the matrix. Hence for any vector x, $B_{n,\ell}^{-1}x$ can be obtained by band solver in $O(\ell^2 n)$ operations. In contrast, solution of circulant systems requires $O(n \log n)$ operations. We remark that in [15], Ku and Kuo have considered using products of lower- and upper-triangular band-Toeplitz matrices as preconditioners. Their resulting preconditioners are in general non-Toeplitz and hence are different from ours.

The outline of the paper is as follows. In §2, we analyze the convergence rate of our preconditioned systems $B_{n,\ell}^{-1}A_n$ in terms of the generating functions g of B_n and f of A_n. In §3, we describe the Remez algorithm and how it is applied to construct the generating function g and hence the preconditioner $B_{n,\ell}$. In §4, we present numerical results that confirm our analysis in §2. In §5, we discuss the use of regularization, a technique that is relevant in computations corresponding to f having zeros and especially when n is large. Finally, concluding remarks are given in §6.

2. Convergence analysis. In this section, we analyze the convergence rate of the preconditioned conjugate gradient method in terms of the generating functions f and g.

We first note that if f is nonnegative, then A_n is always positive definite.

LEMMA 2.1. *Let $f_{\min}$ and $f_{\max}$ be the minimum and maximum of f in $[-\pi, \pi]$. If $f_{\min} < f_{\max}$, then for all $n > 0$,*

$$f_{\min} < \lambda_i(A_n) < f_{\max}, \quad i = 1, \ldots, n,$$

where $\lambda_i(A_n)$ is the ith eigenvalue of A_n. In particular, if $f \geq 0$, then A_n are positive definite for all n.

The proof of the lemma can be found in Chan [4]. Next we give a bound on the condition number of the preconditioned systems.

THEOREM 2.2. *Let f be the generating function of A_n and let g be the generating function of a band-Toeplitz matrix $B_{n,\ell}$:*

$$g(x) = \sum_{j=-(\ell-1)}^{\ell-1} b_j e^{ijx}, \qquad b_j = \bar{b}_{-j}.$$

Then, if

$$\left\| \frac{f-g}{f} \right\|_\infty = h < 1,$$

then $B_{n,\ell}$ is positive definite and

$$\kappa(B_{n,\ell}^{-1}A_n) \leq \frac{1+h}{1-h}, \qquad n = 1, 2, 3, \ldots.$$

Proof. By assumption, we have

$$f(x)(1-h) \leq g(x) \leq f(x)(1+h) \quad \forall x \in [-\pi, \pi].$$

Clearly, $g(x)$ is nonnegative. In particular, by Lemma 2.1, $B_{n,\ell}$ is positive definite for $n = 1, 2, 3, \ldots$. Since A_n and $B_{n,\ell}$ are Toeplitz matrices, we have

$$u^* A_n u = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) \left| \sum_{j=0}^{n-1} u_j e^{ijx} \right|^2 dx$$

and

$$u^* B_n u = \frac{1}{2\pi} \int_{-\pi}^{\pi} g(x) \left| \sum_{j=0}^{n-1} u_j e^{ijx} \right|^2 dx$$

for an arbitrary (complex) n-vector $u^* = [u_0, u_1, \ldots, u_{n-1}]$, see Grenander and Szegö [14]. Hence, we get

$$(1) \qquad (1-h) u^* A_n u \le u^* B_n u \le (1+h) u^* A_n u.$$

Since $B_{n,\ell}$ is also positive definite, we finally have

$$\kappa(B_{n,\ell}^{-1} A_n) \le \frac{1+h}{1-h}. \qquad \square$$

By standard error analysis of the conjugate gradient method, see, for instance, Axelsson and Barker [1, p. 14], we conclude that the number of iterations required for convergence is bounded by

$$\frac{1}{2} \left(\frac{1+h}{1-h} \right) \log \left(\frac{1}{\tau} \right) + 1,$$

where τ is the tolerance. Since h is given explicitly in the Remez algorithm, we have an a priori bound on the number of iterations required for convergence.

3. Construction of preconditioner by the Remez algorithm. In this section, we describe how we construct our band-Toeplitz preconditioner. For ease of presentation, we assume the matrix A_n is real symmetric. Handling the complex Hermitian case will become clear once the symmetric case is explained. Since A_n is symmetric, its generating function f is even, in addition to being 2π-periodic and continuous. It suffices to consider f on $[0, \pi]$. The generating function of a band-Toeplitz matrix $B_{n,\ell}$,

$$B_{n,\ell} = \begin{pmatrix} b_0 & b_1 & \cdots & b_{\ell-1} & & & & \\ b_1 & \ddots & \ddots & & \ddots & & & \\ \vdots & \ddots & \ddots & \ddots & & \ddots & & \\ b_{\ell-1} & & \ddots & \ddots & \ddots & & \ddots & \\ & \ddots & & \ddots & \ddots & \ddots & & b_{\ell-1} \\ & & \ddots & & \ddots & \ddots & \ddots & \vdots \\ & & & \ddots & & \ddots & \ddots & b_1 \\ & & & & b_{\ell-1} & \cdots & b_1 & b_0 \end{pmatrix}$$

is given by

$$g(x) = b_0 + b_1(2\cos(x)) + b_2(2\cos(2x)) + \cdots + b_{\ell-1}(2\cos((\ell-1)x)).$$

Thus, in the case where $f > 0$ on $[0, \pi]$, determining the optimal P is a standard linear minimax approximation problem:

$$\text{minimize}_{p_0, p_1, \ldots, p_{\ell-1}} \|1 - P(x)\|_\infty ,$$

where

$$P(x) = \sum_{j=0}^{\ell-1} p_j \phi_j(x), \quad \phi_0 = 1/f(x) \quad \text{and} \quad \phi_j(x) = 2\cos(jx)/f(x) \quad \text{for } j > 0.$$

Note that $P = g/f$. This optimal P (and hence g) can be obtained by a standard Remez algorithm (see Cheney [10], for example). We, however, use the version proposed by Tang [18], which can be extended to handle the case when $f(x_0) = 0$ for some $x_0 \in [0, \pi]$. We now describe this version of the Remez algorithm briefly; after that, the extension will also be explained.

Given $\phi_0(x) = 0$, and $\phi_j(x) = 2\cos(jx)/f(x)$, $j = 1, \ldots, \ell - 1$, we are to solve

Problem $\mathcal{P}$.

$$\text{Minimize} \quad h \text{ subject to } \quad h \geq s\left(1 - \sum_{j=0}^{\ell-1} p_j \phi_j(x)\right), \quad (s, x) \in \{-1, 1\} \times [0, \pi].$$

One can think of Problem $\mathcal{P}$ as a linear programming problem (by, say, replacing $[0, \pi]$ by a finite set of points). The dual of this problem is given by the following.

Problem $\mathcal{D}$.

$$\text{Maximize} \quad \sum_{s,x} s \cdot r_{s,x} \text{ subject to } \quad r_{s,x} \geq 0, \quad \text{and} \quad \sum_{s,x} r_{s,x} \phi_j(x) s = 0, \quad j = 0, 1, \ldots, \ell-1.$$

It is observed in [18] that even without discretizing the domain $[0, \pi]$, the Simplex algorithm can be applied to Problem $\mathcal{D}$. The preconditioners in the next section are obtained by this computation.

Now, suppose that $f(x_0) = 0$. In practice, x_0 is often known. Because $f \geq 0$ (lest A_n has negative eigenvalues for large enough n), we have $f'(x_0) = 0$ also. Suppose that $f''(x_0) \neq 0$, then we would determine P by imposing the constraint $g(x_0) = 0$, that is,

$$p_0 + 2\sum_{j=1}^{\ell-1} p_j \cos(jx_0) = 0.$$

This linear constraint on the coefficients p_j's can be naturally added to Problem $\mathcal{P}$ and translated to its dual form in Problem $\mathcal{D}$. In general, the case when $f^{(k)}(x_0) = 0$ for $k = 0, 1, \ldots, m$ can be handled by the constraints $g^{(k)}(x_0) = 0$ for $k = 0, 1, \ldots, m - 1$.

We note that when A_n is complex Hermitian, f will not be even necessarily (but still continuous real-valued and 2π-periodic). The domain of approximation becomes $[-\pi, \pi]$ and the approximant will be trigonometric polynomials with sin and cos. Similar constraints can be imposed when $f(x_0) = 0$ for some x_0.

Let us end the section by discussing the computational cost of our method. As pointed out in [18], the number of Simplex iterations needed to determine g is proportional to ℓ. In

practice, all of our experiments took less than 2ℓ iterations. Moreover, after an initial LU decomposition of an ℓ-by-ℓ matrix, each Simplex iteration requires only a modification to the decomposition after a rank-1 change. The total effort for location g is $O(\ell^3)$. We stress the fact that g is independent of n. Thus, as long as f is fixed and a sufficient bandwidth ℓ is reached, the entries for $B_{n,\ell}$ are determined for all n.

In each iteration of the preconditioned conjugate gradient method, we have to compute matrix vector multiplications of the form $A_n x$ and $B_{n,\ell}^{-1} y$. We note that $A_n x$ can be computed in $O(n \log n)$ operations by first embedding A_n into a $2n$-by-$2n$ circulant matrix and then perform the multiplication by the Fast Fourier Transform (see Strang [17]). The vector $z = B_{n,\ell}^{-1} y$ can be obtained by solving the banded system $B_{n,\ell} z = y$ with any band solvers, see, for instance, Golub and Van Loan [13], or Wright [19] for a parallel one. Typically, we will decompose $B_{n,\ell}$ into some triangular factors and then solve the system by a backward and forward solve. The cost of obtaining the triangular factors is $O(\ell^2 n)$, and each subsequent solve will cost $O(\ell n)$, as the triangular factors will also be banded.

Recall that the number of iterations is independent of the size of the matrix n; we therefore conclude that the total complexity of our method is $O(n \log n + \ell^2 n)$.

4. Numerical results. In this section, we compare the convergence rate of the band-Toeplitz preconditioner with a circulant preconditioner on five different generating functions. They are $\cosh x$, $x^4 + 1$, $1 - e^{-x^2}$, $(x - 1)^2(x + 1)^2$, and x^4. The first two functions are positive while the others have either one or two distinct zeros. The matrices A_n are formed by evaluating the Fourier coefficients of the generating functions.

We note that when $f(x) = 1 - e^{-x^2}$, its Fourier coefficients cannot be evaluated exactly. In this case, we approximate them by

$$\begin{aligned} a_j &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ijx} dx \\ &\approx \frac{1}{2n} \sum_{k=0}^{2n-1} f\left(\frac{k\pi}{n} - \pi\right) e^{-ij(k\pi/n - \pi)}, \quad j = 0, \pm 1, \pm 2, \ldots, \end{aligned}$$

where the last expression is evaluated by using the Fast Fourier Transform.

In our tests, the vector of all ones is the right-hand side vector, the zero vector is the initial guess and the stopping criterion is $||r_q||_2 / ||r_0||_2 \leq 10^{-7}$, where r_q is the residual vector after q iterations. All computations are done by Matlab on a Sun workstation. Tables 1–5 show the number of iterations required for convergence with different choices of preconditioners. In the tables, I denotes that no preconditioner is used, C is the T. Chan circulant preconditioner [8], and $B_{n,\ell}$ is the band-Toeplitz preconditioner with half-bandwidth ℓ.

We note that for the cases when the f's are positive, our preconditioners, with half-bandwidths four to five, work as well as the circulant preconditioners. In the cases when the f's have zeros, our preconditioned systems still converge at a rate that is independent of the sizes of the matrices. For the circulant preconditioned systems, however, the numbers of iterations required grow as the sizes of the matrices increase.

5. Regularization. We note that when f has zeros, the system of equations $A_n x = b$ will become very ill conditioned when n is large. Thus the usefulness of the solution x can be in doubt even though we can solve for it quickly using our preconditioners. In this case, one can employ the technique of regularization to alleviate the problem. One approach is to solve the appended least squares problem:

$$\min_x \left\| \begin{bmatrix} A_n \\ \mu P_n \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|_2 .$$

TABLE 1
Numbers of iterations for $f(x) = \cosh x$.

n	I	C	$B_{n,2}$	$B_{n,3}$	$B_{n,4}$	$B_{n,5}$
16	9	6	9	7	6	5
32	16	6	10	7	6	6
64	21	5	11	8	6	6
128	23	5	10	8	6	6
256	24	5	10	7	6	6

TABLE 2
Numbers of iterations for $f(x) = x^4 + 1$.

n	I	C	$B_{n,2}$	$B_{n,3}$	$B_{n,4}$	$B_{n,5}$
16	10	9	9	8	8	7
32	22	7	16	11	8	7
64	37	7	22	12	8	7
128	56	6	25	12	8	7
256	67	6	26	12	8	7

TABLE 3
Numbers of iterations for $f(x) = 1 - e^{-x^2}$.

n	I	C	$B_{n,2}$	$B_{n,3}$	$B_{n,4}$	$B_{n,5}$
16	9	6	9	7	4	3
32	14	7	15	7	5	3
64	24	8	17	8	5	3
128	42	10	17	8	5	3
256	77	13	17	8	5	3

Here μ is the regularization parameter and the n-by-n matrix P_n is the regularization operator that tries to smooth the solution x to a certain degree. Choosing P_n as the $2k$th difference operator will force the solution to have a small $2k$th derivative. We note that the corresponding P_n will be a banded Hermitian matrix with half-bandwidth $k + 1$. Typical choices of P_n are the n-by-n identity matrix and the one-dimensional discrete Laplacian matrix. Choosing the regularization parameter μ, on the other hand, is usually not a trivial problem. One may need to solve the least squares problem for several values of μ to determine the best one; see Eldén [11].

TABLE 4
Numbers of iterations for $f(x) = (x - 1)^2(x + 1)^2$.

n	I	C	$B_{n,3}$	$B_{n,4}$	$B_{n,5}$	$B_{n,6}$
16	11	9	9	9	8	7
32	27	14	13	11	9	7
64	74	17	16	11	8	7
128	193	22	18	11	8	7
256	465	28	19	11	8	7

TABLE 5
Numbers of iterations for $f(x) = x^4$.

n	I	C	$B_{n,3}$	$B_{n,4}$	$B_{n,5}$	$B_{n,6}$
16	12	10	9	9	9	7
32	34	16	15	10	11	9
64	119	26	21	13	11	9
128	587	77	24	15	12	10
256	> 1000	179	27	16	12	10

The solution to the least squares problem can be obtained by solving the normal equation:

$$(A_n^2 + \mu^2 P_n^2)x = A_n b.$$

An obvious choice of preconditioners for the normal equation is the band matrix $B_{n,\ell}^2 + \mu^2 P_n^2$. Its half-bandwidth is $\max(2\ell - 1, 2k + 1)$. By using (1), we can easily show that

$$\kappa\left\{(B_{n,\ell}^2 + \mu^2 P_n^2)^{-1}(A_n^2 + \mu^2 P_n^2)\right\} \leq \left(\frac{1+h}{1-h}\right)^2.$$

In contrast, even if C_n is a good circulant preconditioner for A_n, the matrix $C_n^2 + \mu^2 P_n^2$ will no longer be circulant. However, we remark that regularization techniques using other circulant preconditioners have been considered in Chan, Nagy, and Plemmons [7].

6. Concluding remarks. By understanding the Toeplitz preconditioner from the point of view of minimax approximation of the corresponding generating functions, we can construct band-Toeplitz preconditioners that offer fast convergence rates even when the matrix to be preconditioned has a generating function with a zero. Moreover, our preconditioner with modest bandwidth is also an excellent choice for f without a zero. We emphasize that for a given f, the entries of the preconditioners are unchanged as n increases. Thus, we need to invoke the Remez algorithm once for each f. We note, moreover, that the Cholesky factors of $B_{n,\ell}$ can be used to build the Cholesky factors of $B_{n+1,\ell}$. That can reduce the cost of factorization of the band-Toeplitz preconditioner. Finally, we remark that our preconditioner can also be adapted easily to give a good preconditioner for Toeplitz-plus-band systems of the form $(A_n + D_n)x = b$, where D_n is an arbitrary band matrix. Toeplitz-plus-band systems appear in solving Fredholm integral-differential equations, see Delves and Mohamed [9, p. 343] and, in signal processing literature, see Carayannis, Kalouptsidis, and Manolakis [2].

For such systems, direct Toeplitz solvers and the preconditioned conjugate gradient method with circulant preconditioners will not work. However, one can use $B_{n,\ell} + D_n$ as a preconditioner and use the proof mentioned in Chan and Ng [6] to derive basically the same result that we have in Theorem 2.2, namely that

$$\kappa\left\{(B_{n,\ell} + D_n)^{-1}(A_n + D_n)\right\} \leq \frac{1+h}{1-h}.$$

Hence the number of iterations required for convergence is still fixed independent of the size of the matrices. Since $B_{n,\ell} + D_n$ is a band matrix, the system $(B_{n,\ell} + D_n)x = y$ can still be solved efficiently by band solvers for any vector y.

REFERENCES

[1] O. Axelsson and V. Barker, *Finite Element Solution of Boundary Value Problems, Theory and Computation*, Academic Press Inc., New York, 1984.

[2] G. Carayannis, N. Kalouptsidis, and D. Manolakis, *Fast recursive algorithms for a class of linear equations*, IEEE Trans. Acoust. Speech Signal Process., 30 (1982), pp. 227–239.

[3] R. Chan and G. Strang, *Toeplitz Equations by Conjugate Gradients with Circulant Preconditioner*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 104–119.

[4] R. Chan, *Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions*, IMA J. Numer. Anal., 11 (1991), pp. 333–345.

[5] R. Chan and M. Yeung, *Circulant preconditioners for Toeplitz matrices with positive continuous generating functions*, Math. Comp., 58 (1992), pp. 233–240.

[6] R. Chan and M. Ng, *Fast iterative solvers for Toeplitz-plus-band systems*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 1013–1019.

[7] R. CHAN, J. NAGY, AND R. PLEMMONS, *Circulant preconditioned Toeplitz least squares iterations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 80–97.
[8] T. CHAN, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.
[9] L. DELVES AND J. MOHAMED, *Computational Methods for Integral Equations*, Cambridge University Press, Cambridge, U.K., 1985.
[10] E. CHENEY, *Introduction to Approximation Theory*, Chelsea, New York, 1986.
[11] L. ELDÉN, *An algorithm for the regularization of ill-conditioned, banded least squares problems*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 237–254.
[12] I. GOHBERG AND I. FEL'DMAN, *Convolution Equations and Projection Methods for Their Solutions*, Transaction of Mathematical Monographs, 41, American Mathematical Society, Providence, Rhode Island, 1974.
[13] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, 2nd Ed., The Johns Hopkins University Press, Baltimore, Maryland, 1989.
[14] U. GRENANDER AND G. SZEGÖ, *Toeplitz Form and Its Applications*, 2nd Ed., Chelsea, New York, 1984.
[15] K. KU AND C. KUO, *Minimum-phase LU factorization preconditioners for Toeplitz matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1470–1487.
[16] A. OPPENHEIM, *Applications of Digital Signal Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1978.
[17] G. STRANG, *A proposal for Toeplitz matrix calculations*, Stud. Appl. Math., 74 (1986), pp. 171–176.
[18] P. TANG, *A fast algorithm for linear complex Chebyshev approximation*, Math. Comp., 51 (1988), pp. 721–739.
[19] S. J. WRIGHT, *Parallel algorithms for banded linear systems*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 824–842.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 172–184, January 1994

012

SOME REMARKS ON A MULTIGRID PRECONDITIONER*

JINCHAO XU† AND JINSHUI QIN†

Abstract. This paper is devoted to a class of multilevel preconditioners developed in [*Math Comp.*, 55 (1990), pp. 1–22] by Bramble, Pasciak, and Xu and in the Ph.D. thesis of Xu at Cornell University, Ithaca, NY, 1989. A simple proof is given for optimal estimation of the conditioning. A derivation of matrix representations and a description of efficient implementation techniques are also given. Some modified preconditioners are proposed combining the hierarchical basis method. Numerical examples are also given that compare various preconditioners discussed in this paper.

Key words. finite element, hierarchical basis, multigrid, preconditioner

AMS subject classifications. 65F10, 65F35, 65M60, 65N30

1. Introduction.

We consider the following model boundary-value problem:

$$\begin{aligned} -\Delta U &= F \quad \text{in } \Omega, \\ U &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{1.1}$$

where $\Omega \subset \mathbb{R}^d$ is a polyhedral domain.

Let $H^1(\Omega)$ be the standard Sobolev space consisting of square integrable functions with square integrable derivatives of first order and let $H_0^1(\Omega)$ be a subspace of $H^1(\Omega)$ consisting of functions that vanish on $\partial\Omega$ (in an appropriate sense). Clearly, $U \in H_0^1(\Omega)$ is the solution of

$$a(U, \chi) = (F, \chi) \quad \forall\, \chi \in H_0^1(\Omega), \tag{1.2}$$

where

$$(F, \chi) = \int_\Omega F\chi dx \quad \text{and} \quad a(U, \chi) = \int_\Omega \nabla U \cdot \nabla \chi dx. \tag{1.3}$$

We approximate the above problem by finite element discretizations on a quasi-uniform triangulation $\mathcal{T}$ of Ω. We assume that the triangulation $\mathcal{T}$ is constructed by a successive refinement process. That is, $\mathcal{T} = \mathcal{T}_j$ for some $j > 1$ and $\mathcal{T}_k$ for $k \leq j$ are a nested sequence of quasi-uniform triangulations, which consist of simplexes $\mathcal{T}_k = \{\tau_k^i\}$ of size h_k for $k = 1, \ldots, j$ such that $\Omega = \cup_i \tau_k^i$, where the quasi-uniformity constants are independent of k (cf. Ciarlet [6]). These triangulations should be nested in the sense that any simplex $\tau_{k-1}^i \in \mathcal{T}_{k-1}$ can be written as a union of some simplexes of $\{\tau_k^i\} \in \mathcal{T}_k$. We further assume that there is a constant $\eta > 1$, independent of k, such that h_k is proportional to η^{-k}. In the two-dimensional case, for example, the finer grid is obtained by connecting the midpoints of the edges of the triangles of the coarser grid with $\mathcal{T}_1$ being the initial triangulation.

Corresponding to each triangulation $\mathcal{T}_k$, a finite element space $\mathcal{M}_k$ is defined by

$$\mathcal{M}_k = \{v \in H_0^1(\Omega) : v|_\tau \in \mathcal{P}_1(\tau) \quad \forall\, \tau \in \mathcal{T}_k\},$$

where $\mathcal{P}_1$ is the space of linear polynomials. As a result, we have a nested sequence of subspaces as follows:

$$\mathcal{M}_1 \subset \mathcal{M}_2 \subset \cdots \subset \mathcal{M}_j.$$

*Received by the editors May 15, 1990; accepted for publication (in revised form) March 16, 1993. This work was supported in part by the National Science Foundation.

†Department of Mathematics, Pennsylvania State University, University Park, Pennsylvania 16802 (xu@math.psu.edu or qin@math.psu.edu).

We shall drop the subscript j and denote $\mathcal{M} = \mathcal{M}_j$ for the finest space.[1] The finite element discretization on the space $\mathcal{M}$ for (1.2) is finding a solution $u \in \mathcal{M}$ such that

$$a(u, v) = (F, v) \quad \forall\, v \in \mathcal{M}. \tag{1.4}$$

For each k, $\mathcal{M}_k$ has a natural nodal basis $\{\phi_i^k\}_{i=1}^{n_k}$ ($n_k = \dim \mathcal{M}_k$) that satisfies

$$\phi_i^k(x_l^k) = \delta_{il} \quad \forall\, i, l = 1, \ldots, n_k, \tag{1.5}$$

where $\{x_l^k : l = 1, \ldots, n_k\} \equiv \mathcal{N}_k$ is the set of all interior nodal points of the triangulation $\mathcal{T}_k$ on which $\mathcal{M}_k$ is defined. By means of the nodal basis functions on the finest $\mathcal{M}$, (1.4) is reduced to the following algebraic system:

$$A\alpha = b, \tag{1.6}$$

where

$$A = ((\nabla\phi_i, \nabla\phi_l))_{n\times n},\ \alpha = (\alpha_i)_{i=1}^n \quad \text{and} \quad b = ((F, \phi_i))_{i=1}^n \tag{1.7}$$

(with $n = n_j$), and A is often called the stiffness matrix, which is symmetric and positive definite.

It is well known that the condition number of A is $O(h^{-2})$, and hence (1.6) is an ill-conditioned system. Thus preconditioners (used with the preconditioned conjugate gradient method) are often constructed for its solution. What we shall study in this paper is a class of multilevel preconditioners (in the form of functional operators) proposed in [2] and [10].

Since the publication of [2], [10], there are a number of works devoted to the improvements of the estimate of the conditioning. One improvement was made by Zhang [14]. The optimal estimate was first proved by Oswald [9] using Besov space technique. Alternative proofs of the optimal estimate have been given recently by Bramble and Pasciak [1], Bornemann and Yserentant [3], and Xu [11]. In this paper, we shall present a simple proof for such an estimate.

The preconditioners in [2], [10] were presented in terms of functional operators. In this paper we shall show that their matrix representations can be written as follows:

$$B = \sum_{k=1}^{j} h_k^{2-d} T_k T_k^t, \tag{1.8}$$

where h_k is the meshsize of $\mathcal{T}_k$ and $T_k \in \mathbb{R}^{n\times n_k}$ is the representation matrix of the nodal basis $\{\phi_i^k\}$ for $\mathcal{M}_k$ in terms of the nodal basis $\{\phi_i\}$ for $\mathcal{M}$ (for $k \le j$ since $\mathcal{M}_k \subset \mathcal{M}$). More precisely,

$$\Phi_k = \Phi T_k,$$

where

$$\Phi_k = (\phi_1^k, \ldots, \phi_{n_k}^k), \qquad \Phi = (\phi_1, \ldots, \phi_n). \tag{1.9}$$

The preconditioner B given by (1.8) depends entirely on the transformation matrices between the nodal bases in different levels. For this reason we shall call this preconditioner *the multilevel nodal basis* preconditioner.

[1] Throughout this paper, if a subscript or superscript is j, it will often be dropped from the notation without further explanation.

The simple algebraic structure of (1.8) makes it easy to implement the preconditioner that is the basis for a very simple and efficient implementation technique for the preconditioner presented in this paper.

Another purpose of this paper is to introduce some modified preconditioners. The first example will be a Gauss–Seidel version of (1.8) as follows:

$$B = \sum_{k=1}^{j} T_k G_k T_k^t, \tag{1.10}$$

where G_k is the symmetric Gauss–Seidel iterative matrix obtained from the stiffness matrix on $\mathcal{M}_k$, namely, $A_k = ((\nabla\phi_i^k, \nabla\phi_l^k))_{n_k \times n_k}$. Numerical experiments show that this preconditioner results in a much smaller condition number than (1.8).

Another modified preconditioner we propose is related to an algorithm developed by Yserentant [12] that is often called the hierarchical basis preconditioner. The hierarchical basis preconditioner is quite similar to (1.8) and, in fact, can be derived from (1.8) by eliminating certain n_{k-1} columns from each T_k. Based on such an observation, the hierarchical basis preconditioner can be slightly modified to improve its efficiency. The implementation techniques for (1.8) can also be applied directly to hierarchical basis preconditioner and its modified version.

For convenience of exposition, we shall use MNB (*multilevel nodal basis*) or GS-MNB for the preconditioner (1.8) or (1.10), and HB for the hierarchical basis preconditioner.

The outline of the remainder of the paper is as follows. In the next section, the preconditioner proposed in [2], [10] is reviewed; §3 contains the derivation of expression (1.8); §4 is devoted to the implementation of the algorithms; and §5 discusses the hierarchical basis preconditioner, some variants of (1.8), and a few numerical examples that compare the performances of different algorithms.

2. Multilevel preconditioners and optimal conditioning estimates. In this section, we shall give a brief description of the multilevel preconditioners to be studied in this paper and then present a proof on their conditioning. Even though the theory holds in more general cases, the focus in this paper is only on its applications to finite element equations with quasi-uniform triangulations.

We first need to introduce some linear operators on the finite element functional spaces. For distinction, we will use boldface letters to denote these operators. In contrast, notation for matrices is not boldface unless otherwise specified.

With respect to the L^2 inner product $(\cdot, \cdot)$ (given by (1.3)), for each k we define operators $\boldsymbol{A}_k : \mathcal{M}_k \mapsto \mathcal{M}_k$ and $\boldsymbol{Q}_k : \mathcal{M} \mapsto \mathcal{M}_k$ by

$$(\boldsymbol{A}_k u, v) = a(u, v), \quad (\boldsymbol{Q}_k u, v) = (u, v), \quad u \in \mathcal{M}, v \in \mathcal{M}_k. \tag{2.1}$$

The operator $\boldsymbol{A}_k$(which is symmetric positive definite) may be regarded as a discretization of the elliptic operator $-\Delta$ on $\mathcal{M}_k$ and $\boldsymbol{Q}_k$ is the standard orthogonal L^2 projection.

Again we denote $\boldsymbol{A} = \boldsymbol{A}_j$. By definition, problem (1.4) is equivalent to

$$\boldsymbol{A}u = f$$

for some appropriate $f \in \mathcal{M}$. One of the preconditioners for $\boldsymbol{A}$ proposed in [2], [10] is given by

$$\boldsymbol{B}v = \sum_{k=1}^{j} h_k^{2-d} \sum_{i=1}^{n_k} (v, \phi_i^k)\phi_i^k, \qquad v \in \mathcal{M}. \tag{2.2}$$

It is easy to see that $\boldsymbol{B}$ is symmetric positive definite with respect to L^2 inner product.

The effectiveness of the preconditioner $\boldsymbol{B}$ is measured by the condition number of $\boldsymbol{BA}$, denoted by $\kappa(\boldsymbol{BA})$, which is the ratio of its maximum eigenvalue to the minimum one. We have the following theorem.

THEOREM 2.1. *There exists a constant C independent of k or j such that*

$$\kappa(\boldsymbol{BA}) \leq C.$$

The constant C appearing in the estimate in Theorem 2.1 behaves like $O(h_1^{-2})$, where h_1 is the size of triangulation $\mathcal{T}_1$. This is usually not a problem since h_1 is often of unit size in practical computations. Nevertheless, if h_1 is considerably small, the preconditioner can be modified by solving the problem on $\mathcal{M}_1$ exactly:

$$\boldsymbol{B}v = \boldsymbol{A}_1^{-1}\boldsymbol{Q}_1 + \sum_{k=2}^{j} h_k^{2-d} \sum_{i=1}^{n_k} (v, \phi_i^k)\phi_i^k \tag{2.3}$$

for which the relative condition number is independent of h_1 (cf. [10]).

A slightly more general preconditioner is also given in [2]:

$$\boldsymbol{B}_R = \sum_{k=1}^{j} \boldsymbol{R}_k \boldsymbol{Q}_k, \tag{2.4}$$

where $\boldsymbol{R}_k : \mathcal{M}_k \mapsto \mathcal{M}_k$ are symmetric positive definite operators satisfying

$$c_0 \lambda_k^{-1} \|v\|^2 \leq (\boldsymbol{R}_k v, v) \leq c_1 \lambda_k^{-1} \|v\|^2 \quad \forall\, v \in \mathcal{M}_k,$$

where $\|\cdot\|^2 = (\cdot, \cdot)$. It can be shown that Theorem 2.1 also holds for (2.4).

Using the techniques in [2], [10], the proof of Theorem 2.1 follows directly from the following.[2]

PROPOSITION 2.2. *There exist positive constants c_0 and c_1 such that*

$$c_0 \|v\|^2_{H^1(\Omega)} \leq \sum_{k=1}^{\infty} \|(\boldsymbol{Q}_k - \boldsymbol{Q}_{k-1})v\|^2_{H^1(\Omega)} \leq c_1 \|v\|^2_{H^1(\Omega)} \quad \forall\, v \in H_0^1(\Omega).$$

The above proposition can be derived from Oswald [9] and we shall present a new proof of this result in the rest of this section.

The proof to be presented can also be found in Xu [11] and the idea is similar to a proof in Bramble and Pasciak [1]. The main ingredient in our analysis is the fractional Sobolev space.

The fractional Sobolev space $H_0^{m+\sigma}(\Omega)$ ($m = 0, 1$ and $0 < \sigma < 1$) is defined by the completion of $C_0^\infty(\Omega)$ in the following norm:

$$\|v\|_{H^{m+\sigma}(\Omega)} = \left(\|v\|^2_{H^m(\Omega)} + |v|^2_{H^{m+\sigma}(\Omega)}\right)^{1/2},$$

where

$$|v|^2_{H^{m+\sigma}(\Omega)} = \sum_{|\alpha|=m} \int_\Omega \int_\Omega \frac{|D^\alpha v(x) - D^\alpha v(y)|^2}{|x - y|^{d+2\sigma}} dx\, dy.$$

[2]Here we assume there are infinitely many number of levels of nested triangulations. The result is more general than the case of finite number of levels.

For $\alpha \in (0, 1]$, we define $H^{-\alpha}(\Omega) = (H_0^{\alpha}(\Omega))^*$, the dual of $H_0^{\alpha}(\Omega)$.

The following inverse inequalities (cf. [4], [10]) hold:

$$\|v\|_{H^{1+\sigma}(\Omega)} \lesssim h_k^{-\sigma}\|v\|_{H^1(\Omega)}, \qquad \|v\|_{H^s(\Omega)} \lesssim h_k^{-s}\|v\| \quad \forall\, v \in \mathcal{M}_k, \tag{2.5}$$

where $\sigma \in (0, \frac{1}{2})$ and $s \in [0, 1]$.

Let $\boldsymbol{P}_k : H_0^1(\Omega) \mapsto \mathcal{M}_k$ be the H^1 projection defined by

$$(\nabla \boldsymbol{P}_k u, \nabla v_k) = (\nabla u, \nabla v_k) \quad \forall\, u \in H_0^1(\Omega),\, v_k \in \mathcal{M}_k.$$

It follows from the standard finite element approximation theory that

$$\|v - \boldsymbol{P}_k v\|_{H^{1-\alpha}(\Omega)} \lesssim h_k^{\alpha}\|v\|_{H^1(\Omega)} \quad \forall\, v \in H_0^1(\Omega) \tag{2.6}$$

for some constant $\alpha \in (0, 1]$ (depending on the domain Ω).

It is well known that (cf. Bramble and Xu [5])

$$\|v - \boldsymbol{Q}_k v\| + h_k\|\boldsymbol{Q}_k v\|_{H^1(\Omega)} \lesssim h_k\|v\|_{H^1(\Omega)} \quad \forall\, v \in H_0^1(\Omega). \tag{2.7}$$

By interpolation, we have (for $\sigma \in (0, 1)$, $\sigma \neq \frac{1}{2}$)

$$\|\boldsymbol{Q}_k v\|_{H^{\sigma}(\Omega)} \lesssim \|v\|_{H^{\sigma}(\Omega)} \quad \forall\, v \in H_0^1(\Omega). \tag{2.8}$$

The first ingredient in our proof is the so-called strengthened Cauchy inequality. This result is similar to the corresponding result by Yserentant [12] for his hierarchical basis preconditioner. A constructive proof of this inequality may be found in Xu [11], and the proof presented here is a modification of a proof of Bramble and Pasciak [1].

LEMMA 2.3. *There is a constant $\gamma \in (0, 1)$ and a positive constant C such that*

$$(\nabla u_i, \nabla v_j) \le C\gamma^{|i-j|}\|\nabla u_i\|\,\|\nabla v_j\| \quad \forall\, u, v \in H_0^1(\Omega),\, i, j \ge 1,$$

where $u_i = (\boldsymbol{Q}_i - \boldsymbol{Q}_{i-1})u$ and $v_j = (\boldsymbol{Q}_j - \boldsymbol{Q}_{j-1})v$.

Proof. Let $D_i = \partial/\partial x_i$. Obviously $D_i : H_0^1(\Omega) \mapsto L^2(\Omega)$ is continuous, and by duality, so is $D_i : L^2(\Omega) \mapsto H^{-1}(\Omega)$. By interpolation, for $\alpha \in (0, \frac{1}{2})$, $D_i : H^{1-\alpha}(\Omega) \mapsto H^{-\alpha}(\Omega)$ is also continuous. Consequently, $\|\nabla v\|_{H^{-\alpha}(\Omega)} \lesssim \|v\|_{H^{1-\alpha}(\Omega)}$, for $v \in H_0^1(\Omega)$. Therefore (without loss of generality, we assume that $j \ge i$),

$$(\nabla u_i, \nabla v_j) \lesssim \|\nabla u_i\|_{H^{\alpha}(\Omega)}\|\nabla v_j\|_{H^{-\alpha}(\Omega)} \lesssim \|u_i\|_{H^{1+\alpha}(\Omega)}\|v_j\|_{H^{1-\alpha}(\Omega)}.$$

It follows from the inverse inequality (2.5) that

$$(\nabla u_i, \nabla v_j) \lesssim h_i^{-\alpha}\|u_i\|_{H^1(\Omega)}h_j^{-(1-\alpha)}\|v_j\| \lesssim (\eta^{\alpha})^{j-i}h_j^{-1}\|u_i\|_{H^1(\Omega)}\|v_j\|.$$

An application of (2.7) then completes the proof with $\gamma = \eta^{\alpha}$. □

We are now in a position to present the proof for Proposition 2.2.

Denote $\tilde{\boldsymbol{Q}}_k = \boldsymbol{Q}_k - \boldsymbol{Q}_{k-1}$ and $v_i = (\boldsymbol{P}_i - \boldsymbol{P}_{i-1})v$. It follows from (2.5), (2.8), and (2.6) that

$$\|\tilde{\boldsymbol{Q}}_k v_i\|_{H^1(\Omega)}^2 \lesssim \lambda_k^{\alpha}\|\tilde{\boldsymbol{Q}}_k v_i\|_{H^{1-\alpha}(\Omega)}^2 \lesssim \lambda_k^{\alpha}\|v_i\|_{H^{1-\alpha}(\Omega)}^2 \lesssim \lambda_k^{\alpha}h_i^{2\alpha}\|v_i\|_{H^1(\Omega)}^2.$$

Let $i \wedge j = \min(i, j)$; we have

$$\begin{aligned}
&\sum_{k=1}^{\infty} \|(\boldsymbol{Q}_k - \boldsymbol{Q}_{k-1})v\|^2_{H^1(\Omega)} \\
&\quad \le c_1 \|v\|^2_{H^1(\Omega)} \\
&\quad = \sum_{k=1}^{\infty} \sum_{i,j=k}^{\infty} (\nabla \tilde{\boldsymbol{Q}}_k v_i, \nabla \tilde{\boldsymbol{Q}}_k v_j) \\
&\quad \lesssim \sum_{i,j=1}^{\infty} \sum_{k=1}^{i\wedge j} (\nabla \tilde{\boldsymbol{Q}}_k v_i, \nabla \tilde{\boldsymbol{Q}}_k v_j) \\
&\quad \lesssim \sum_{i,j=1}^{\infty} \sum_{k=1}^{i\wedge j} \lambda_k^{\alpha} h_i^{\alpha} h_j^{\alpha} \|v_i\|_{H^1(\Omega)} \|v_j\|_{H^1(\Omega)} \\
&\quad \lesssim \sum_{i,j=1}^{\infty} \lambda_{i\wedge j}^{\alpha} h_i^{\alpha} h_j^{\alpha} \|v_i\|_{H^1(\Omega)} \|v_j\|_{H^1(\Omega)} \\
&\quad \lesssim \sum_{i,j=1}^{\infty} \gamma^{\alpha|i-j|} \|v_i\|_{H^1(\Omega)} \|v_j\|_{H^1(\Omega)} \\
&\quad \lesssim \sum_{i=1}^{\infty} \|v_i\|^2_{H^1(\Omega)}.
\end{aligned}$$

On the other hand, by the strengthened Cauchy inequality

$$\begin{aligned}
\|v\|^2_{H^1(\Omega)} &= \sum_{i,j=1}^{\infty} (\nabla \tilde{\boldsymbol{Q}}_i v, \nabla \tilde{\boldsymbol{Q}}_j v) \\
&\lesssim \sum_{i,j=1}^{\infty} \gamma^{|i-j|} \|\tilde{\boldsymbol{Q}}_i v\|_{H^1(\Omega)} \|\tilde{\boldsymbol{Q}}_j v\|_{H^1(\Omega)} \\
&\lesssim \sum_{i=1}^{\infty} \|\tilde{\boldsymbol{Q}}_i v\|^2_{H^1(\Omega)}.
\end{aligned}$$

This completes the proof of Proposition 2.2.

3. Preconditioners in terms of matrices. The preconditioners in the form of (2.2) are convenient for theoretical analysis. But for numerical implementation, a matrix form is more desirable. In this section, we shall derive the corresponding preconditioning matrix that can be applied directly to the system (1.6). For a more systematic approach, we refer to Xu [11].

In this section, we shall use $(\cdot, \cdot)$ for the usual L^2 product and $\langle \cdot, \cdot \rangle$ for the usual Euclidean inner product of vectors in any Euclidean spaces.

To relate the functional operator to matrix, for a given operator $\boldsymbol{R} : \mathcal{M} \mapsto \mathcal{M}$, let $\boldsymbol{R}_m$ be the $n \times n$ matrix [3] representation of $\boldsymbol{R}$ satisfying

$$\langle \boldsymbol{R}_m \mu, \nu \rangle = (\boldsymbol{R} u, v), \qquad u, v \in \mathcal{M},$$

[3] Please note an abuse of notation here; a boldface with subscript m, e.g., $\boldsymbol{R}_m$, denotes a matrix.

where $\mu = (\mu_i) = (u(x_i^j)) \in \mathbb{R}^n$ and $v = (v_i) = (v(x_i^j)) \in \mathbb{R}^n$. $\boldsymbol{R}_m$ may be called the matrix representation of the operator $\boldsymbol{R}$.

The above definition of the matrix representation is a little different from the usual definition for a matrix corresponding to a linear operator on a linear vector space. The advantage of this definition can be seen from the following lemma.

LEMMA 3.1. *The matrix representation of A (defined by (2.1) for $k = j$) is just the usual stiffness matrix A given by* (1.7).

The proof of the above lemma is straightforward by definition. The unusual property of our matrix representation of operators is that the product of two operators does not quite correspond to the product of their matrices.

LEMMA 3.2. *For any two operators $\boldsymbol{R}, \boldsymbol{S}$ on $\mathcal{M}$,*

$$(\boldsymbol{R} + \boldsymbol{S})_m = \boldsymbol{R}_m + \boldsymbol{S}_m, \qquad (\boldsymbol{RS})_m = \boldsymbol{R}_m M^{-1} \boldsymbol{S}_m$$

with M being the mass matrix (which is symmetric positive definite) given by

$$M = ((\phi_i, \phi_l))_{n\times n}. \tag{3.1}$$

Proof. The first identity is obvious by definition. To see the second one, for any $u, v \in \mathcal{M}$, we introduce a mapping $E : \mathbb{R}^n \mapsto \mathcal{M}$ defined by

$$E\mu = \sum \mu_i \phi_i.$$

Thus $u = E\mu$ if $\mu_i = u(x_i)$ and $v = Ev$ if $v_i = v(x_i)$. Let $E^* : \mathcal{M} \mapsto \mathbb{R}^n$ be the adjoint of E, namely,

$$\langle E^* u, v\rangle = (u, Ev).$$

Obviously, both E^* and E are isomorphisms. By definition, one can easily check that

$$\boldsymbol{R}_m = E^* \boldsymbol{R} E, \qquad E^* E = M.$$

Consequently,

$$(\boldsymbol{RS})_m = E^* \boldsymbol{R} S E = E^* \boldsymbol{R} E (E^* E)^{-1} E^* S E = \boldsymbol{R}_m M^{-1} \boldsymbol{S}_m.$$

This proves the second identity. □

In order to see what $\boldsymbol{B}_m$ is, by Lemma 3.2, it suffices to obtain $(\boldsymbol{B}_k)_m$ for each k, where

$$\boldsymbol{B}_k v = \sum_{i=1}^{n_k} (v, \phi_i^k)\phi_i^k, \qquad v \in \mathcal{M}. \tag{3.2}$$

To do this, taking $u, v \in \mathcal{M}$, we have

$$(\boldsymbol{B}_k u, v) = \sum_{i=1}^{n_k} (u, \phi_i^k)(v, \phi_i^k) = \langle \alpha, \beta\rangle, \tag{3.3}$$

where $\alpha = ((u, \phi_1^k), \ldots, (u, \phi_{n_k}^k))^t$ and $\beta = ((v, \phi_1^k), \ldots, (v, \phi_{n_k}^k))^t$.

Note that, with Φ_k and Φ given by (1.9), $u = \Phi\mu$ and $\Phi_k = \Phi T_k$. Hence

$$\begin{aligned}\alpha &= \int_\Omega u \Phi_k^t dx = \int_\Omega \Phi \mu T_k^t \Phi^t dx \\ &= \int_\Omega T_k^t (\Phi^t \Phi)\mu dx = T_k^t \left(\int_\Omega \Phi^t \Phi dx\right)\mu.\end{aligned}$$

Observing that $\int_\Omega \Phi^t \Phi dx$ is exactly the mass matrix M defined by (3.1), we then get

$$\alpha = T_k^t M \mu$$

and, similarly, $\beta = T_k^t M \nu$. Combining these identities with (3.3) yields

$$\begin{aligned}(\boldsymbol{B}_k u, v) &= \langle T_k^t M\mu, T_k^t M\nu\rangle \\ &= \langle M T_k T_k^t M\mu, \nu\rangle,\end{aligned}$$

which means that

$$(\boldsymbol{B}_k)_m = M T_k T_k^t M.$$

Therefore, by (2.2), (3.2), and Lemma 3.2,

$$\boldsymbol{B}_m = \sum_{k=1}^{j} h_k^{2-d} (\boldsymbol{B}_k)_m = MBM, \tag{3.4}$$

where

$$B = \sum_{k=1}^{j} h_k^{2-d} T_k T_k^t. \tag{3.5}$$

Let λ_0 and λ_1 be the minimum and maximum eigenvalue of $\boldsymbol{BA}$, respectively,

$$\lambda_0(\boldsymbol{A}v, v) \le (\boldsymbol{BA}v, v) \le \lambda_1(\boldsymbol{A}v, v) \quad \forall\, v \in \mathcal{M}.$$

By the definition of the matrix representation

$$\lambda_0\langle \boldsymbol{A}_m \nu, \nu\rangle \le \langle (\boldsymbol{BA})_m \nu, \nu\rangle \le \lambda_1 \langle \boldsymbol{A}_m \nu, \nu\rangle \quad \forall\, \nu \in \mathbb{R}^n.$$

The above expression involves the matrix representation of a product of operators. By Lemma 3.2,

$$(\boldsymbol{BA})_m = \boldsymbol{A}_m M^{-1} \boldsymbol{B}_m M^{-1} \boldsymbol{A}_m.$$

Applying Lemma 3.1 then yields

$$(\boldsymbol{BA})_m = ABA,$$

where B is given by (3.5). Hence

$$\lambda_0\langle A\nu, \nu\rangle \le \langle BA\nu, \nu\rangle \le \lambda_1 \langle A\nu, \nu\rangle \quad \forall\, \nu \in \mathbb{R}^n.$$

Consequently,

$$\kappa(BA) = \kappa(\boldsymbol{BA}).$$

In summary, we have the following theorem.

THEOREM 3.3. *The matrix B given by* (3.5) *is a preconditioner of the stiffness matrix A such that*

$$\kappa(BA) \le C.$$

Corresponding to (2.3), we can have the following more refined preconditioner:

$$B = T_1 A_1^{-1} T_1^t + \sum_{k=2}^{j} h_k^{2-d} T_k T_k^t.$$

More generally, A_1^{-1} may be replaced by a preconditioner of A_1.

4. On the implementation. In this section, we shall discuss some techniques for the implementation of the preconditioner B given by (3.5).

As the preconditioner is applied with the preconditioned conjugate gradient method, it suffices to consider the computation of, for $\alpha \in \mathbb{R}^n$, $B\alpha$ and $\langle B\alpha, \alpha\rangle$ as follows:

$$B\alpha = \sum_{k=1}^{j} h_k^{2-d} T_k T_k^t \alpha, \qquad \langle B\alpha, \alpha\rangle = \sum_{k=1}^{j} h_k^{2-d} \langle T_k^t \alpha, T_k^t \alpha\rangle. \tag{4.1}$$

The implementation of the above computations was also discussed in the earlier paper [2]. But the current version of the algorithm allows us to discuss this issue in a more straightforward way. If, for $k \le l$, T_k^l denotes the matrix that transforms the nodal basis of $\mathcal{M}_k$ to the nodal basis of $\mathcal{M}_l$, the following relation is then obvious by definition:

$$T_l^l = I, \qquad T_k^l = T_{l-1}^l T_k^{l-1}, k < l.$$

Let, for $1 \le l \le j$,

$$B_l = \sum_{k=1}^{l} h_k^{2-d} T_k^l (T_k^l)^t.$$

By definition $B = B_j$ and

$$B_l = h_l^{2-d} I + T_{l-1}^l B_{l-1} (T_{l-1}^l)^t.$$

We shall use the above recurrence relation to compute the action of B. Assume that m_l is the number of operations that are needed to compute the action $B_l\alpha_l$ for $\alpha_l \in \mathbb{R}^{n_l}$. By the identity

$$B_l\alpha_l = h_l^{2-d}\alpha_l + T_{l-1}^l [B_{l-1}(T_{l-1}^l)^t \alpha_l],$$

we get, for some constant $c_0 > 0$,

$$m_l \le m_{l-1} + c_0 n_l$$

from which we conclude that

$$m_j \le m_1 + c_0 \sum_{l=2}^{j} n_l \le c_1 n$$

for some positive constant c_1. This means that the action of B_j can be carried out within $O(n)$ operations.

In summary, we have the following algorithm.

AN ALGORITHM FOR COMPUTING $B\alpha$ AND $\langle B\alpha, \alpha\rangle$

$\alpha_j = \alpha$; $c = h_j^{2-d}\langle \alpha, \alpha\rangle$;
for $l = j - 1 : 1$,
  $\alpha_l = T_l^{l+1}\alpha_{l+1}$; $c = c + h_l^{2-d}\langle \alpha_l, \alpha_l\rangle$;
end
$\langle B\alpha, \alpha\rangle = c$;
$\beta_1 = \alpha_1$;
for $l = 2 : j$,
  $\beta_l = h_l^{2-d}\alpha_l + T_{l-1}^l \beta_{l-1}$;
end
$B\alpha = \beta_j$.

As discussed above, the number of operations needed in the above algorithm is $O(n)$. We also note that all the vectors α_l for $1 \le l \le j$ need to be stored, but the whole storage space for these vectors is also only $O(n)$.

5. Other variants and numerical examples. In this section, we shall discuss several other methods that are related to our main algorithm. We shall first present a preconditioner by means of Gauss–Seidel iteration. We shall then have a look at the relationship between the preconditioner (1.8) and the hierarchical basis preconditioner. By comparing these two methods, a modified preconditioner will be proposed. Finally, some numerical examples will be presented.

5.1. Gauss–Seidel-type algorithms. By the more general version of the multigrid preconditioner (2.4), $\boldsymbol{R}_k$ are operators that are free to choose. Choosing $\boldsymbol{R}_k = h_k^2 \boldsymbol{I}$ that is related to the damped Jacobi iteration, we get the preconditioner and hence (3.5). It is well known that the Gauss–Seidel iteration often has better performance than the Jacobi method. In our case, a Gauss–Seidel algorithm can be obtained by choosing $\boldsymbol{R}_k$ from the symmetric Gauss–Seidel iteration as follows:

$$B_g = \sum_{k=1}^{j} T_k G_k T_k^t,$$

where $G_k = (D_k + U_k)^{-1} D_k (D_k + L_k)^{-1}$ from the decomposition of the stiffness matrix $A_k = D_k + L_k + U_k$ with D_k being the diagonal, L_k the lower triangle, and U_k the upper triangle of A_k.

By comparing their asymptotical theoretical estimates, B_g is spectrally equivalent to the B in (1.8). However, our numerical experiments seem to indicate that B_g results in much smaller condition numbers.

5.2. Hierarchical basis preconditioner. Since the hierarchical basis preconditioner only works well for one- or two-dimensional problems (cf. [8], [11]), the following discussion is then confined in a two-dimensional problem. The so-called hierarchical functions are defined in $\mathcal{M}_1$ by the nodal basis of $\mathcal{M}_1$ and in $\mathcal{M}_k$, for $k = 2, \ldots, j$, by the hierarchical basis functions of $\mathcal{M}_{k-1}$, together with those nodal basis functions $\{\phi_i^k\}$ of $\mathcal{M}_k$ that correspond to the nodes in $\mathcal{N}_k \setminus \mathcal{N}_{k-1}$. It is easy to see that the set of all the hierarchical basis functions on $\mathcal{M}$ indeed form a basis for $\mathcal{M}$.

Let $S \in \mathbb{R}^{n \times n}$ be the matrix that transforms the nodal basis into hierarchical basis, namely,

$$\Psi = \Phi S,$$

where Ψ is an n row vector whose components are hierarchical basis functions and Φ is an n row vector whose components are nodal basis functions. In terms of S, the hierarchical basis preconditioner [12] can be represented by

$$H = SS^t.$$

Following [12] and [10], we let $S_k \in \mathbb{R}^{n \times (n_k - n_{k-1})}$ be the matrix that represents the part of hierarchical basis on $\mathcal{M}_k$ in terms of nodal basis in $\mathcal{M}$, then we have $S = (S_1, \ldots, S_j)$ and

$$H = \sum_{k=1}^{j} S_k S_k^t. \tag{5.1}$$

Notice that S_k is a submatrix of T_k; hence we have

$$\langle H\alpha, \alpha \rangle \leq \langle B\alpha, \alpha \rangle \quad \forall\, \alpha \in \mathbb{R}^n. \tag{5.2}$$

A more detailed discussion of the relationship between the MNB preconditioner and the HB preconditioner can be found in Xu [10], [11] and in Yserentant [13]. We note that the

action of H is slightly easier to compute than that of B, but, from the theoretical estimates as well as numerical experiments, B is much better than H as far as the resulting condition numbers are concerned. We would also like to mention that in three-dimensional or higher cases, the HB preconditioner deteriorates (cf. Ong [8]), but the MNB preconditioner does not.

5.3. A modified preconditioner. In view of the inequality (5.2), we introduce the following modified preconditioner:

$$\hat{H} = \sum_{k=1}^{j-1} S_k S_k^t + I. \tag{5.3}$$

It is trivial to see that

$$\langle H\alpha, \alpha\rangle \leq \langle \hat{H}\alpha, \alpha\rangle \leq \langle B\alpha, \alpha\rangle \quad \forall\, \alpha \in \mathbb{R}^n.$$

The preconditioner in (5.3) is obtained by a very slight change from the HB preconditioner, but as we shall see from the numerical experiments, it is much better than the HB preconditioner and sometimes even better than the MNB preconditioner. Notice that, in terms of computational complexity, $\hat{H}$ is quite close to H. For convenience, we use MHB to stand for the modified hierarchical basis preconditioner.

5.4. Some numerical examples. In this subsection, we shall report some numerical experiments that compare the performances of the various preconditioners discussed above.

Our numerical examples will be for the problem (1.1) with piecewise linear finite element discretizations as described in §1. The equation we need to solve is (1.6).

For a given preconditioner B of A, to solve (1.6), we consider the equivalent system of (1.6):

$$BAx = Bb.$$

The conjugate gradient method may be applied to the above system and the resultant algorithm is a preconditioned conjugate gradient method. The effectiveness of the preconditioned conjugate gradient algorithm is measured by the condition number of the preconditioned matrix:

$$\kappa(BA) = \frac{\lambda_{\max}(BA)}{\lambda_{\min}(BA)},$$

where B is any preconditioner mentioned above and $\lambda_{\max}(BA)$ and $\lambda_{\min}(BA)$ are the maximum and minimum eigenvalue of BA, respectively. These eigenvalues are calculated by the standard power method and inverse power method. In the inverse power method, the corresponding systems can be solved effectively by the conjugate gradient method since BA is a well-conditioned matrix.

We have tested the MNB, HB, and MHB preconditioners on three different domains: (a) unit square; (b) slit domain; (c) irregular polygon. All the multilevel triangulations will be determined by the first triangulation $\mathcal{T}_1$ as all the other triangulations will be obtained by connecting the midpoints of the edges of the triangles from the previous coarse mesh. The first triangulations for different domains are shown in Fig. 1 (with broken lines).

The numerical results are shown in Figs. 2–4 for square, slit, and irregular polygon domains, respectively. The correspondences between pictures with different preconditioners are as follows:

$$\kappa(HA) : -\cdot-\cdot-, \quad \kappa(\hat{H}A) : \text{———}, \quad \kappa(BA) : - - --, \quad \kappa(B_gA) : \cdots\cdots.$$

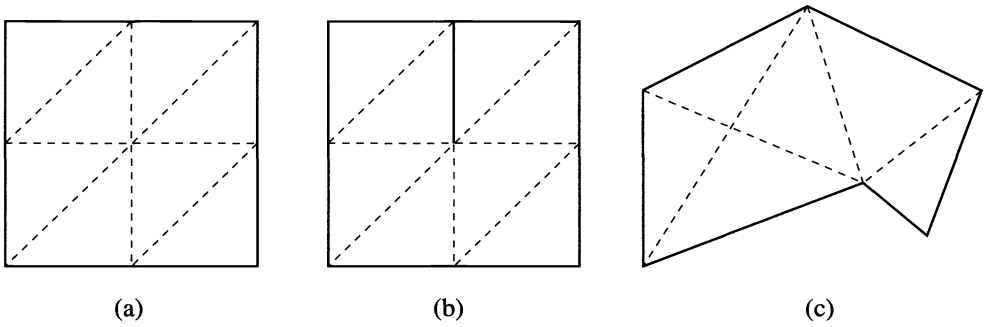


FIG. 1. *Three different test regions.*

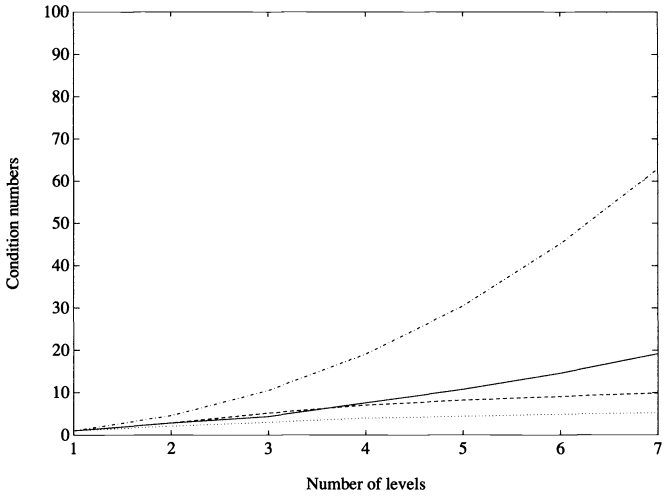


FIG. 2. *Square domain.*

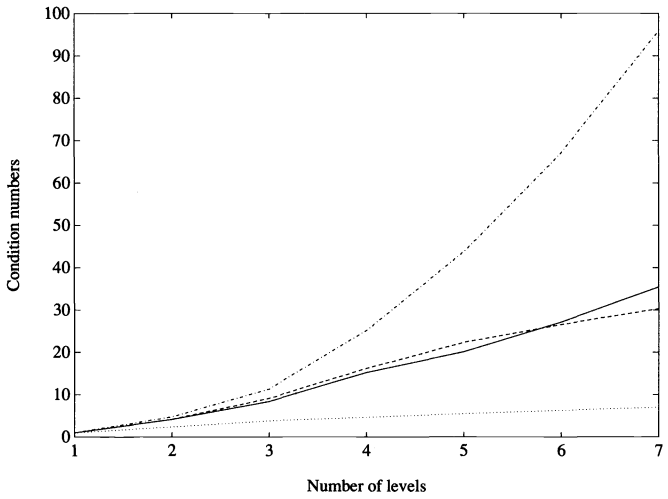


FIG. 3. *Slit domain.*

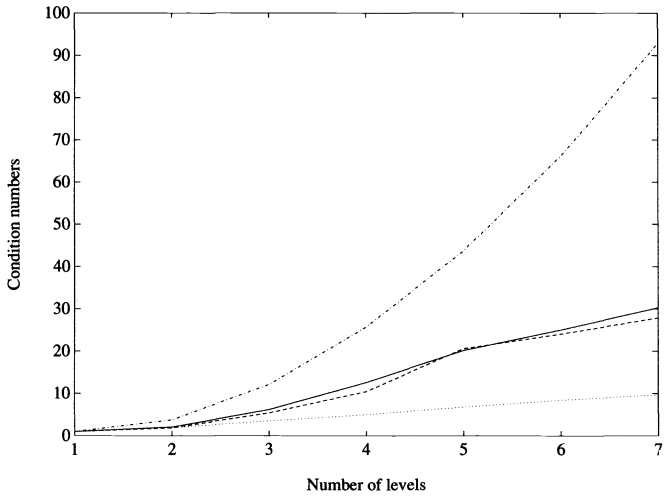


FIG. 4. *Irregular polygon.*

As shown by the pictures, MNB, GS-MNB, and MHB preconditioners result in much smaller condition numbers than the HB preconditioner, and GS-MNB gives the smallest condition number. It is most interesting to compare MHB with HB and MNB. In terms of computational complexity, the difference between MHB and HB is negligible, but the resultant condition numbers differ considerably. Note also that the behaviors of $\kappa(\hat{H}A)$ and $\kappa(BA)$ are quite close.

REFERENCES

[1] J. BRAMBLE AND J. PASCIAK, *New estimates for multilevel algorithms including the V-cycle*, preprint, 1991.
[2] J. BRAMBLE, J. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 55, 191 (1990), pp. 1–22.
[3] F. BORNEMANN AND H. YSERENTANT, *A basic norm equivalence for the theory of multilevel methods*, preprint, 1992.
[4] J. BRAMBLE, J. PASCIAK, AND J. XU, *The analysis of multigrid algorithms with non-imbedded spaces or non-inherited Quadratic forms*, Math. Comp., 56, 193 (1991), pp. 1–34.
[5] J. BRAMBLE AND J. XU, *Some estimates for a weighted L^2 projection*, Math. Comp., 56, 194 (1991), pp. 463–476.
[6] P. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, New York, 1978.
[7] W. HACKBUSCH, *Multi–Grid Methods and Applications*, Springer-Verlag, Berlin, Heidelberg, New York, 1985.
[8] E. ONG, *Hierarchical basis preconditioners for second order elliptic problems in three dimensions*, Ph.D. thesis; CAM Report 89–31, Dept. of Mathematics, UCLA, preprint, 1990.
[9] P. OSWALD, *On discrete norm estimates related to multilevel preconditioners in the finite element method*, preprint.
[10] J. XU, *Theory of Multilevel Methods*, Ph.D. thesis, Cornell University, Ithaca, NY, 1989. AM Report 48, Dept. of Mathematics, Pennsylvania State University, University Park, PA, July 1989.
[11] ———, *Iterative methods by space decomposition and subspace corrections*, SIAM Rev., 34 (1992) pp. 581–613.
[12] H. YSERENTANT, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.
[13] ———, *Two preconditioners based on the multilevel splitting of finite element spaces*, Numer. Math., 58 (1990), pp. 163–184.
[14] X. ZHANG, *Multilevel additive Schwarz methods*, Numer. Math., 63 (1992), pp. 521–539.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 185–206, January 1994

013

FAST ADAPTIVE METHODS FOR THE FREE-SPACE HEAT EQUATION*

JOHN STRAIN†

Abstract. Standard numerical methods for the heat equation in two or more space dimensions are excellent if it is necessary to follow the evolution in great detail through many small timesteps. This paper presents efficient and accurate new adaptive methods that solve the free-space heat equation with *large* timesteps. These methods combine the fast Gauss transform with an adaptive refinement scheme that represents the solution as a continuous piecewise polynomial, to a user-specified degree of accuracy. The same approach is extended to solve inhomogeneous problems and to solve the heat equation in moving domains with boundaries. In problems with boundaries, it allows the use of accurate boundary representations without requiring difficult product integration formulas or precluding fast evaluation schemes. Numerical experiments in two space dimensions show these methods to be accurate and efficient, especially for highly nonuniform or discontinuous initial data or when substantial accuracy is required.

Key words. heat equation, heat potentials, fast algorithms, adaptive methods, crystal growth

AMS subject classifications. 65M50, 33A65, 35K05, 80A20, 65V05, 65R10

1. Introduction.

It is often necessary to solve the free-space heat equation

$$\begin{aligned} u_t &= \Delta u \quad \text{in} \quad \mathbf{R}^2 \\ u &= f \quad \text{at} \quad t = 0 \end{aligned} \tag{1}$$

with a difficult initial temperature field f: We suppose that f vanishes at infinity, is globally Lipschitz but not necessarily C^1, and varies rapidly over only a small portion of its support. One needs values of u at arbitrary points, not on a regular grid.

In this paper, we present efficient and accurate new numerical methods for solving this problem. These methods are *accurate* in the sense that the user can specify a precision ϵ, and the solution is then produced within error ϵ (relative to $|u|_\infty$). The work required increases as ϵ decreases, as it must, but these methods are *efficient* in the sense that they attempt to represent u with as few degrees of freedom as possible, adapting those degrees of freedom to fit u. These methods are efficient in another sense as well; the work required to go from $u(x, t)$ to $u(x, t + \Delta t)$ is proportional to the number of degrees of freedom required and *decreases* as Δt increases. (For explicit numerical methods, the work required *increases* as Δt increases, and for implicit methods, it is superlinear in the number of degrees of freedom.)

Our methods are based on the exact evolution formula

$$u(x, t + \Delta t) = \frac{1}{\pi \delta} \int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y, t)\, dy, \qquad \delta = 4\Delta t. \tag{2}$$

We begin by projecting the initial temperature field f onto a finite-dimensional space of piecewise polynomial interpolants, chosen adaptively to resolve f within error ϵ. Then we use Hermite expansions to evaluate $u(x, \Delta t)$ via (2) efficiently, again within error ϵ, and project $u(x, \Delta t)$ onto another adaptively chosen finite-dimensional space that resolves it within error ϵ. This process can be repeated at each timestep, or the Hermite expansion coefficients can be updated to simply evaluate u at any desired time $t \geq \Delta t$. The first approach is more applicable to inhomogeneous problems, the second approach more efficient.

*Received by the editors August 19, 1991; accepted for publication (in revised form) March 17, 1993. This work was supported by Air Force Office of Scientific Research grant AFOSR-91-0165.

†Department of Mathematics, Princeton University, Princeton, New Jersey 08544. Current address: Department of Mathematics, University of California, Berkeley, California 94720 (strain@math.berkeley.edu).

The techniques used here can also be used to solve the more general problem

$$u_t = \Delta u + F(x,t) \quad \text{in } \Omega \subset \mathbf{R}^2$$

$$\alpha u + \beta \frac{\partial u}{\partial n} = g \quad \text{on } \partial\Omega \tag{3}$$

$$u = f \quad \text{at } t = 0,$$

with Ω a possibly time-dependent subset of the plane having a bounded C^2 boundary $\partial\Omega$, n the outward unit normal to $\partial\Omega$, and α, β, g, f, and F C^2 functions on their domains.

We first discuss the inhomogeneous free-space problem

$$u_t = \Delta u + F(x,t) \quad \text{in } \mathbf{R}^2$$

$$u = f \quad \text{at } t = 0 \tag{4}$$

as an example of the technique employed on (3). We solve (4) by Duhamel's principle and evaluate the resulting integral by the trapezoidal rule; this is second-order accurate in Δt. Simpson's rule gives a fourth-order method, and higher-order methods are similarly easy to construct.

The organization of the paper is as follows. In §2, we derive the Hermite expansion of the heat kernel and truncation error estimates. Next, in §3, we describe how we project a given function f onto a finite-dimensional subspace, which resolves it within error ϵ relative to $|f|_\infty = \max|f|$. In §4, we describe how to combine these two techniques to solve the homogeneous free-space heat equation (1), and in §5 we extend the method to the inhomogeneous equation (4). In §6 we sketch how to solve (3), using our method and heat potential theory; details will be given in a later publication. Numerical experiments are presented in §7 and conclusions in §8.

2. Hermite expansions. The purpose of this section is to describe how certain moments of $u(x,t)$ suffice to evaluate $u(x,t+\Delta t)$ at any point x within an error tolerance ϵ. We use the explicit evolution formula [5]

$$u(x,t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y,t)\,dy, \qquad \delta = 4\Delta t \tag{5}$$

for the bounded solution u of (1).

The heat kernel (with $\delta = 1$ for simplicity) is a real-analytic function of $y \in \mathbf{R}^2$, so we can expand it in a two-dimensional Taylor series:

$$h(x-y) = e^{-|x-y|^2} = \sum_{\alpha \geq 0} \frac{1}{\alpha!} h_\alpha(x) y^\alpha. \tag{6}$$

Here $\alpha = (\alpha_1, \alpha_2)$ is a multi-index with integer components, $\alpha! = \alpha_1!\alpha_2!$; we say that $\alpha \geq 0$ if $\alpha_i \geq 0$ for each $i = 1, 2$, and $y^\alpha = y_1^{\alpha_1} y_2^{\alpha_2}$. The Taylor coefficients are given by

$$h_\alpha(x) = D_y^\alpha h(x-y)\big|_{y=0} = (-1)^{|\alpha|} D_x^\alpha h(x), \tag{7}$$

where $|\alpha| = \alpha_1 + \alpha_2$ and

$$D_y^\alpha = \left(\frac{\partial}{\partial y_1}\right)^{\alpha_1} \left(\frac{\partial}{\partial y_2}\right)^{\alpha_2}.$$

Since the variables x_1 and x_2 are separated in $h(x)$, we have

$$h_\alpha(x) = h_{\alpha_1}(x_1)h_{\alpha_2}(x_2),$$

where each one-variable function $h_n(x)$ is given by

$$\begin{aligned} h_n(x) &= (-1)^n \left(\frac{d}{dx}\right)^n e^{-x^2} \\ &= H_n(x)e^{-x^2}, \end{aligned} \tag{8}$$

and H_n is the usual Hermite polynomial. We will need two facts from [11]. First, the two-term recursion

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x),$$

with $H_0(x) = 1$ and $H_1(x) = 2x$. Second, Cramer's inequality

$$|H_n(x)| \le K2^{n/2}\sqrt{n!}e^{x^2/2},$$

where K is less than 1.09. Hence

$$\frac{1}{\alpha!}|h_\alpha(x)| \le K^2\frac{2^{|\alpha|/2}}{\sqrt{\alpha!}}. \tag{9}$$

The "tail" left after truncating the series (6) after terms with every $\alpha_i \le p$ is bounded uniformly in x by

$$\sum_{\text{some } \alpha_i > p} \frac{1}{\alpha!}|h_\alpha(x)||y^\alpha| \le K^2 \sum_{\text{some } \alpha_i > p} \frac{2^{|\alpha|/2}}{\sqrt{\alpha!}} r^{|\alpha|} = O\left(\frac{2er^2}{p+2}\right)^{(p+1)/2} \tag{10}$$

if $|y_i| \le r$ for $i = 1, 2$, by Stirling's formula. This decreases very rapidly as p increases. Thus the kernel $h(x - y)$ can be approximated by a $(p + 1)^2$-term truncated Taylor series when $|y_i| \le r$, with an error that decreases rapidly as p, increases uniformly in x.

This expansion was used in [6] and [10] to evaluate the two-dimensional discrete Gauss transform

$$G_\delta f(t_i) = \sum_{j=1}^{N} f_j e^{-|t_i - s_j|^2/\delta}, \qquad i = 1, 2, 3, \ldots, M, \tag{11}$$

with $t_i, s_j \in \mathbf{R}^2$, $\delta > 0$, $|t|^2 = t_1^2 + t_2^2$, in $O(N + M)$ time. (In [6], δ was a constant; [10] extended the algorithm to cover the situation when $\delta = \delta_i$ or $\delta = \delta_j$.)

Now rescale and shift this formula with an arbitrary δ and a center of expansion c; we get

$$h\left(\frac{x-y}{\sqrt{\delta}}\right) = h\left(\frac{x-c-(y-c)}{\sqrt{\delta}}\right) = \sum_{\alpha \ge 0} \frac{1}{\alpha!} h_\alpha\left(\frac{x-c}{\sqrt{\delta}}\right)\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha, \tag{12}$$

and the error in truncating after terms with $\alpha_i \le p$ is given by (10) if $|y_i - c_i| \le r\sqrt{\delta}$. Hence $e^{-|x-y|^2/\delta}$ is well represented by a truncated Taylor series in y when y lies in a square with center c and side $2r\sqrt{\delta}$.

This suggests a natural way to evaluate the integral

$$G_\delta f(x) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} f(y)\, dy.$$

Suppose f has compact support. Cover the support of f with nonoverlapping square cells C of side $2r\sqrt{\delta}$ and centers c. In each cell, expand the heat kernel about the cell center using (12);

$$G_\delta f(x) = \frac{1}{\pi} \sum_C \sum_{\alpha \geq 0} C_\alpha h_\alpha \left(\frac{x-c}{\sqrt{\delta}} \right), \tag{13}$$

where

$$C_\alpha = \frac{1}{\delta \alpha!} \int_C \left(\frac{y-c}{\sqrt{\delta}} \right)^\alpha f(y)\, dy. \tag{14}$$

The variables x and y are now separated, so we have decomposed the calculation into two simpler pieces; first, we evaluate the moments C_α for each cell C and for $\alpha_i = 0, 1, \ldots, p$, and then we sum the Hermite series (13) for each point where $G_\delta f(x)$ is to be evaluated. The numerical error in the approximation of $G_\delta f$ by this procedure comes from two sources: first, truncation of the Hermite series, and second, quadrature error in evaluating C_α numerically. We have already discussed the truncation error; the quadrature error will be discussed in the next section.

We can speed up the evaluation considerably, when δ is small, by dropping negligible contributions from cells distant from the point of evaluation x. The contribution from a distant cell C' is bounded by

$$\frac{1}{\pi\delta} \int_{C'} e^{-|x-y|^2/\delta} |f(y)|\, dy \leq \frac{1}{\pi\delta} |f|_\infty |C'| e^{-|x-x_0|^2/\delta},$$

where x_0 is the closest point in C' to x and $|C'|$ is the area of C'. Thus only a fixed number of cells within a distance $O(\sqrt{\delta})$ need to be kept, for any fixed precision. Our next step is to evaluate C_α.

3. Adaptive refinement. The purpose of adaptive refinement is to know a function within ϵ as efficiently as possible. Numerically, we know a function f within ϵ if we can evaluate it at any point x with an error that is less than $\epsilon |f|_\infty$ and a cost that depends only on ϵ.

The Hermite expansion of §2 enables us to evaluate $u(x, t + \Delta t)$ at any point x, not just on a fixed set of grid points. The cost per evaluation is fixed, because each evaluation is independent of the others. Thus we are in the following situation: We have a function, f say, and we can evaluate it at any point for fixed cost per evaluation. How can we best approximate f by piecewise polynomials within error $\epsilon |f|_\infty$? For our applications, f is not very smooth initially, so refinement in space seems a reasonable approach. Thus we use a fixed low degree d of polynomial interpolation and refine the interpolation grid wherever f is not accurately represented. We use a triangular grid and approximate f by degree-d interpolation over each triangle; the degree is selected by the user. When we evaluate the moments C_α, we have to integrate f times powers over cells C. It is convenient if each triangle of our triangulation lies completely within a single cell C. We begin with the simplest such triangulation, which is the one formed by cutting each cell C into two isosceles right triangles and subdividing until f is accurate within ϵ.

Thus we use the following method to construct the adaptive approximation of f; we begin by dividing each cell C into two right triangles and constructing the degree-d interpolant to f at the $(d+1)(d+2)/2$ nodes shown in Fig. 1.

Now we stack all the triangles and sweep through, testing whether each triangle requires subdivision. To test a triangle, we evaluate f at each node that would be produced by bisecting

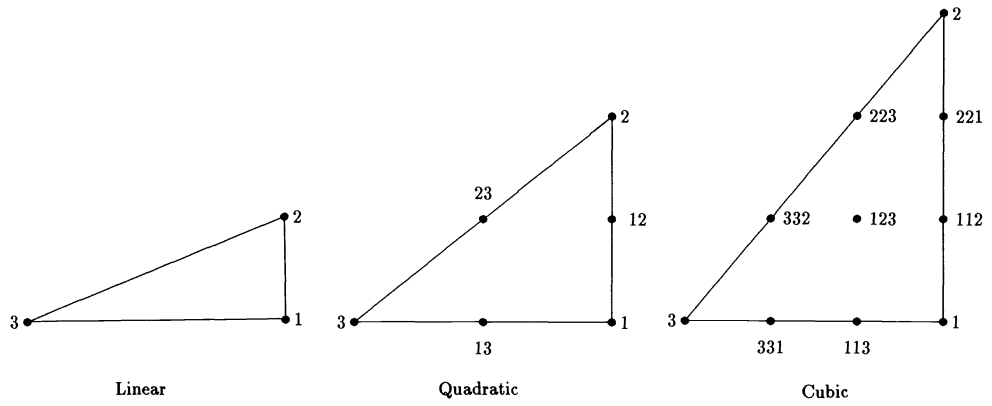


FIG. 1. *Nodes for degree-d interpolation on triangles for* $d = 1, 2,$ *and* 3.

the longest side, as shown in Fig. 2. This requires one evaluation for $d = 1$, three for $d = 2$, and six for $d = 3$; all these values are used if the triangle is subdivided. We also evaluate the degree-d interpolant at the same nodes and compute the maximum difference between the two sets of values. If f is within ϵ (relative to the maximum value of $|f|$ so far encountered) of the interpolant at the new nodes, the triangle is accepted. Otherwise, the triangle is subdivided by Mitchell's newest-node bisection method [8], maintaining compatibility by subdividing neighbors as necessary, and the new triangles are stacked. We then repeat the procedure until the stack is finished.

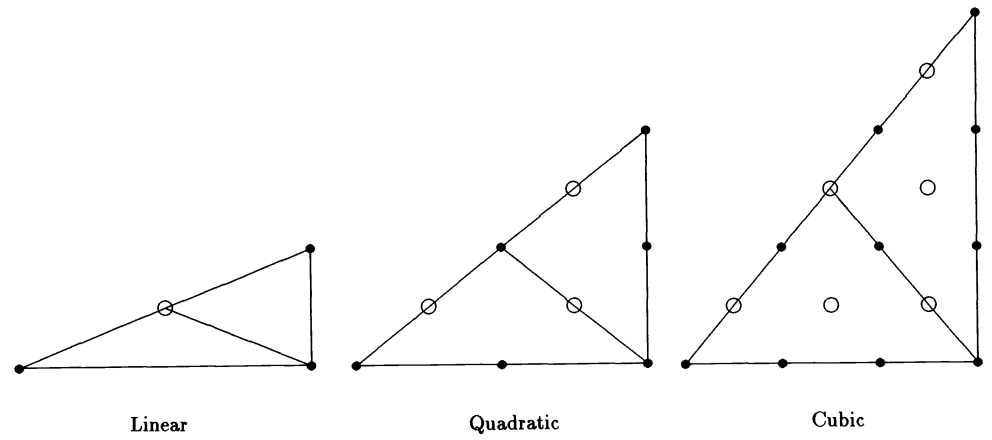


FIG. 2. *New nodes produced when the base is bisected are circled.*

Mitchell's subdivision procedure assigns one vertex of each triangle in the initial triangulation as a "peak" and the side opposite the peak as the base. (In our case, the initial triangulation consists of isosceles right triangles and the peak is the vertex at the right angle, opposite the hypotenuse.) Then it subdivides triangles by dividing the base and the neighboring triangle opposite the peak, with the new vertex being assigned as the peak of each of the four new triangles created by the subdivision. Compatibility is maintained by always subdividing compatible pairs of triangles; if the neighbor opposite the peak is not compatibly subdivisible, it is itself divided recursively until compatibility is maintained. An example is shown in Fig. 3.

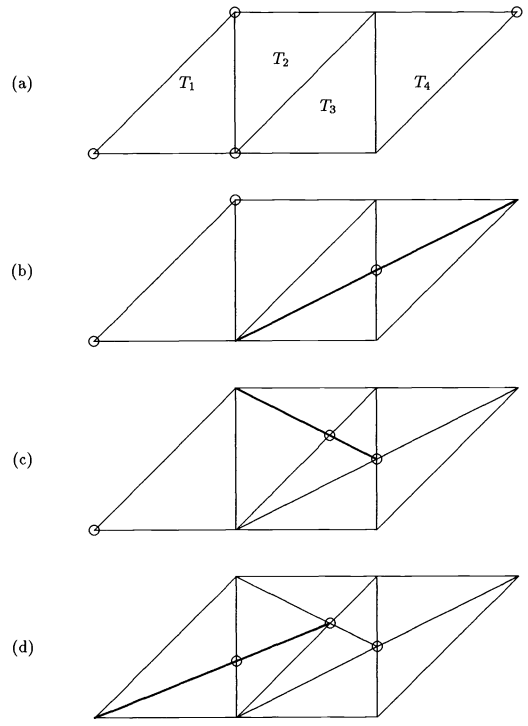


FIG. 3. *An example of Mitchell's recursive newest-node bisection* [8]. *Triangle T_1 is flagged for subdivision, but the peak of its neighbor T_2 (indicated by circle) does not lie opposite T_1. Hence we must refine T_2 and its neighbor T_3. Similarly, the peak of T_3 does not lie opposite T_2, so we must refine T_4 and T_3. The peak of T_4 lies opposite T_3, so the recursion stops here. We then divide triangles backwards in pairs as shown in* (b) *through* (d), *until we have divided the triangle T_1 we originally wanted to divide. The subdivided triangulation is shown in* (d).

Once a triangulation is constructed, we have enough information about f to solve the heat equation to accuracy ϵ after time Δt has elapsed. We need only to use the interpolant of f on the adapted triangulation to evaluate $(p+1)^2$ of the moments C_α within relative error ϵ. This requires integrating the interpolant times $((x-c)/\sqrt{\delta})^\alpha$ over each triangle lying in cell C with center c.

This task is simpler if we put the interpolant, on each triangle, in the shifted and scaled Cartesian form

$$\tilde{f}(x) = \sum_{|\beta| \le d} F_\beta \left(\frac{x-c}{\sqrt{\delta}} \right)^\beta .$$

This requires considerable hand calculation, because $\tilde{f}$ is most naturally expressed in barycentric coordinates as in [3], and the expressions for the Cartesian coefficients F_β can become quite complicated for quadratic or cubic interpolation. The expressions for F_β are given in the Appendix.

Next we need to calculate the approximate moments

$$\begin{aligned} \tilde{C}_\alpha &= \frac{1}{\alpha!\delta} \int_C \left(\frac{y-c}{\sqrt{\delta}} \right)^\alpha \tilde{f}(y)\, dy \\ &= \sum_{T \subset C} \frac{1}{\alpha!\delta} \int_T \left(\frac{y-c}{\sqrt{\delta}} \right)^\alpha \tilde{f}(y)\, dy \end{aligned} \tag{15}$$

using the piecewise polynomial interpolant. Here we have split up the integral over C into a sum of integrals $\tilde{T}_\alpha$, say, over the triangles T contained in C.

Now consider the evaluation of

$$\tilde{T}_\alpha = \frac{1}{\alpha!\delta} \int_T \left(\frac{y-c}{\sqrt{\delta}} \right)^\alpha \tilde{f}(y)\, dy \tag{16}$$

$$= \sum_{|\beta| \le d} F_\beta \frac{1}{\alpha!\delta} \int_T \left(\frac{y-c}{\sqrt{\delta}} \right)^{\beta+\alpha} dy. \tag{17}$$

We need only evaluate triangle moments

$$T_\gamma = \frac{1}{\gamma!\delta} \int_T \left(\frac{x-c}{\sqrt{\delta}} \right)^\gamma dx$$

for $\gamma_i = 0, 1, 2, \ldots, p+d$. Note that while c may not lie in T and δ may be small, we nevertheless have $T \subset C$ and thus $|x_i - c_i| \le r\sqrt{\delta}$ for $x \in T$, where r is a fixed constant. Hence evaluating T_γ and adding up values with coefficients is not a numerically unstable process.

To evaluate T_γ, we first translate and scale T to simplify the notation, then use the Divergence Theorem and recursion. Write $\gamma = (i, j)$ and $x = (x, y)$. Then

$$T_\gamma = T_{ij} = \frac{1}{i!j!} \int_{T'} x^i y^j \, dxdy,$$

where $T' = (T - c)/\sqrt{\delta}$ is a shifted and scaled version of T.

Let v_1, v_2, and v_3 be the vertices of T', arranged in counterclockwise order, and let $v_4 = v_1$. Then by the Divergence Theorem,

$$T_{ij} = \sum_{k=1}^{3} T_{ijk},$$

where explicit parametrization of the line segment $\overline{v_k v_{k+1}}$ gives

$$T_{ijk} = \int_0^1 (y_{k+1} - y_k) \frac{x^{i+1}}{(i+1)!} \frac{y^j}{j!} \, d\theta.$$

Here $(x, y) = v = \theta v_{k+1} + (1-\theta) v_k$ and $v_k = (x_k, y_k)$. Integrating by parts gives a recurrence relation

$$T_{ijk} = \Lambda \left[\frac{x_k^{i+1}}{(i+1)!} \frac{y_k^{j+1}}{(j+1)!} \right] - \frac{\Lambda x_k}{\Lambda y_k} T_{i-1,j+1,k}.$$

Here Λ is the forward difference operator $\Lambda x_k = x_{k+1} - x_k$. We can use this recurrence to compute T_{ijk} for $i, j = 0, 1, 2, \ldots, q = p + d$ if we start with values for T_{0jk} for $j = 0, 1, 2, \ldots, 2q$. Integrating by parts again produces initial values for the recurrence:

$$T_{0jk} = \Lambda \left[x_k \frac{y_k^{j+1}}{(j+1)!} \right] - \frac{\Lambda x_k}{\Lambda y_k} \Lambda \left[\frac{y_k^{j+2}}{(j+2)!} \right].$$

Note that (a) it is efficient to precompute and store the values $x_k^i/i!$ and $y_k^j/j!$, which are used repeatedly, and (b) $T_{ijk} = 0$ when $\Lambda y_k = 0$ (so there is no difficulty with dividing by zero). This concludes the evaluation of T_{ijk} and hence of C_α.

4. The homogeneous problem. We now combine the tools described in §§2 and 3 to construct methods for solving the homogeneous free-space heat equation

$$u_t = \Delta u \quad \text{in } \mathbf{R}^2, \tag{18}$$

$$u(x, 0) = f(x), \tag{19}$$

with f a bounded function that vanishes at infinity. The unique bounded solution u satisfies the semigroup property [5]

$$u(x, t + \Delta t) = \frac{1}{\pi\delta} \int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y, t)\, dy, \tag{20}$$

with $\delta = 4\Delta t$.

Method 1. Taking $t = 0$ and $\Delta t = t$ gives

$$u(x, t) = \frac{1}{\pi\delta} \int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} f(y)\, dy, \qquad \delta = 4t, \tag{21}$$

which represents u at any time t in terms of the convolution of a Gaussian with scale $\delta = 4t$ with the initial temperature field f. Let us suppose that we want to know $u(x, t)$ within relative error ϵ at a sequence of times $t = 0, \Delta t, 2\Delta t, \ldots, N\Delta t = T$, and Δt is not too small. (If Δt is very small and we wish to know u at every step, explicit adaptive finite difference or finite element methods are excellent.)

Let $\delta = 4\Delta t$ and let us first evaluate $u(x, \Delta t)$. We replace f by a piecewise polynomial $\tilde{f}$ constructed, as in §3, to approximate f within $\epsilon|f|_\infty/2$. Then we use $\tilde{f}$ to evaluate the moments $\tilde{C}_\alpha$ defined in (15) for $0 \le \alpha_i \le p$ for each cell. We can now evaluate $u(x, \Delta t)$ by (21) within error ϵ relative to $|f|_\infty$, by a truncated Hermite series

$$u(x, t + \Delta t) = \frac{1}{\pi} \sum_C \sum_{\alpha \ge 0}^{p} C_\alpha h_\alpha \left(\frac{x - c}{\sqrt{\delta}} \right) + E,$$

where the error E comes from several sources; first, the error due to replacing f by $\tilde{f}$, which is bounded by $\epsilon|f|_\infty/2$ by the maximum principle [5]; second, the error due to truncating each Hermite series at $\alpha_i = p$, and third, the error due to ignoring distant cells. These add up to

$$\frac{\epsilon|f|_\infty}{2} + O\left(\frac{2er^2}{p+2} \right)^{(p+1)/2} |f|_\infty + O\left(e^{-4r^2 I^2} \right) |f|_\infty, \tag{22}$$

which can be easily made less than ϵ by choice of p and the cutoff distance once r and N are fixed.

Thus we can evaluate $u(x, \Delta t)$ at any point x. Method 1 is distinguished by the way in which we evaluate u at later times. In principle, a separate computation should be required for each time at which we wish to evaluate u, because the moments C_α depend on t through δ. However, C_α is homogeneous of degree $-1 - |\alpha|/2$ in δ;

$$C_\alpha = \frac{1}{\alpha!\delta} \int_C \left(\frac{y - c}{\sqrt{\delta}} \right)^\alpha f(y)\, dy.$$

Thus $C_\alpha(2\delta) = 2^{-(2+|\alpha|)/2} C_\alpha(\delta)$ if we make the dependence on δ explicit. Once we have $C_\alpha(\delta)$, therefore, we can evaluate $u(x, t)$ with relative error less than ϵ at any time $t \ge \Delta t$,

simply by scaling the coefficients and evaluating. More precisely,

$$u(x,t) = \frac{1}{\pi} \sum_{C} \sum_{\alpha \geq 0}^{p} \left(\frac{t}{\Delta t} \right)^{-(2+|\alpha|)/2} C_\alpha h_\alpha \left(\frac{x-c}{\sqrt{4t}} \right) + E,$$

where $E \leq \epsilon |f|_\infty$. The need to include more cells C in the sum as t increases, due to spreading of the Gaussians, is more than counterbalanced by the fewer coefficients required: C_α is effectively reduced exponentially as $\delta = 4t$ increases.

The scaling of coefficients used in Method 1 was developed in [10] for evaluating the discrete Gauss transform with target-dependent scales.

Method 1 is highly efficient and accurate. It requires a substantial investment of effort at the beginning of the calculation to construct the triangulation and approximate f (especially if f is not very smooth), but evaluation of u gets less and less expensive as time goes by. Moreover, the space required for the triangulation can be reused for other purposes as soon as the coefficients are evaluated and stored. Method 1 is useful whenever the homogeneous free-space heat equation is to be solved for rough data and not too small times.

Two fundamental properties of the heat equation influence this method. First, the increasing smoothness and decreasing variation of $u(x,t)$ as t increases means that it takes fewer and fewer degrees of freedom to represent u within a fixed relative precision ϵ. A typical example is shown in Figs. 4 and 5. In Fig. 4, we display the triangulation needed to resolve

$$f(x) = \sum_{k_1=1}^{10} \sum_{k_2=1}^{10} \cos(k_1(x_1 - \bar{x}_1)) \cos(k_2(x_2 - \bar{x}_2)) e^{-(k_1(x_1-\bar{x}_1))^2 - (k_2(x_2-\bar{x}_2))^2}$$

(where $\bar{x}_1$ and $\bar{x}_2$ are randomly chosen, for each k_1 and k_2, on $[-1, 1]$) within error $\epsilon = 10^{-2}$ relative to $|f|_\infty$, using linear interpolation. Figure 5 displays the triangulation needed to resolve the solution $u(x,t)$ with initial data $f(x)$, at $t = 0.1$, to the same relative precision. The triangulation began with a 6×6 grid of square cells on $[-3, 3]^2$ in both cases; f required 2910 triangles while u required only 608, a reduction in the number of degrees of freedom by almost 5. The smallest triangle for f is 2^{-6} on a side, while at $t = 0.1$, the smallest triangle is 2^{-3}. Thus smoothing by heat flow speeds up later stages of the calculation.

The other property of the heat equation, the spreading support of its solutions, is less benign though hardly malignant. Even though f has compact support, $u(x,t)$ will not have compact support for any $t > 0$, because heat flows instantaneously (though in small amounts) to infinity. Thus if we begin by representing f by a piecewise polynomial on $B = [-R, R]^2$, say, where $|f| \leq \epsilon$ outside B, the support of u will spread and $|u|$ will eventually be greater than ϵ outside B. No fixed region can be used for computing u for arbitrarily long times, because eventually significant heat flows to infinity.

Method 2. We now describe another method for solving (18), with slightly different aims. Method 2 is better suited to solving inhomogeneous problems and extends more easily to solving problems on domains with boundaries and variable-coefficient and nonlinear problems.

The basic idea of Method 2 is simply to restart Method 1 at each step. (More generally, we could restart every few steps.) The advantage in this is that data like inhomogeneous terms and boundaries can more easily enter the evolution.

More precisely, we carry out Method 1 to construct the coefficients C_α of $u(x, \Delta t)$. Then we apply the adaptive refinement strategy again, with $u(x, \Delta t)$ in place of $f(x)$. Thus we construct a new triangulation on which $u(x, \Delta t)$ is equal to a piecewise polynomial of degree d, within error $\epsilon |u(\cdot, \Delta t)|_\infty$. We then repeat the construction of the C_α's with $u(x, \Delta t)$ in place of f, and we then can evaluate $u(x, 2\Delta t)$. We repeat this process, going from real-space

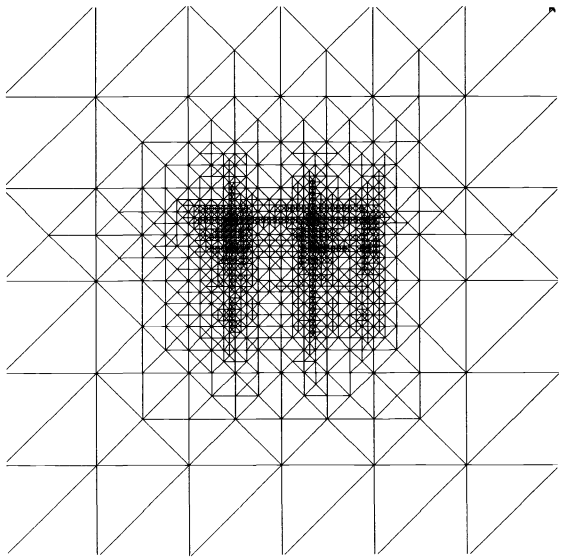


FIG. 4. *Initial triangulation for sum of* 100 *scaled and randomly shifted terms of*

$$u_1(x,t) = \frac{e^{1/4T-1/4}}{\sqrt{T}} \cos\left(\frac{x}{T}\right) e^{-x^2/T},$$

where $T = 1 + 4t$, *resolved to error* $\epsilon = 10^{-2}$ *relative to* $|u_1|_\infty$ *with linear interpolation. The triangulation began with a* 6×6 *grid of square cells on* $[-3, 3]$. *This required* 2910 *triangles and* 1468 *nodes, with minimum side length* $2^{-6.5}$ *times the maximum side length.*

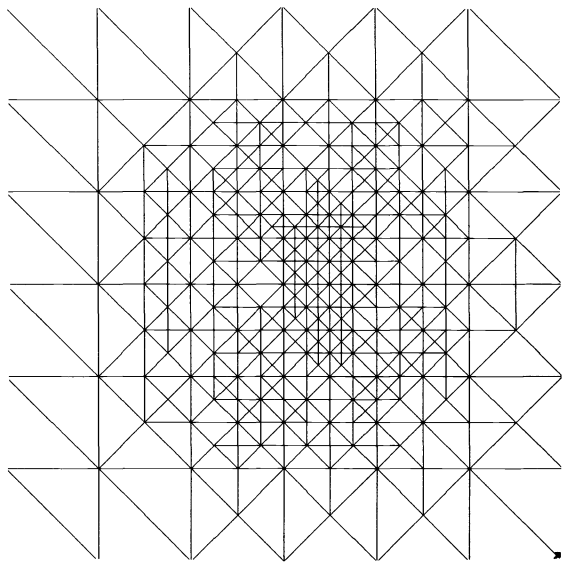


FIG. 5. *Triangulation for* 100 *terms of* u_1 *at* $t = 0.1$, *with* $\epsilon = 10^{-2}$ *and linear interpolation. The increasing smoothness of the solution is reflected in the larger triangles used to represent it; only* 608 *triangles are needed here, with* 317 *nodes and minimum side length* $2^{-3.5}$ *times the maximum side length.*

values sufficient to resolve $u(x, n\Delta t)$ to the coefficients for $u(x, (n+1)\Delta t)$ and back again, at each timestep.

A disadvantage of Method 2 is that in the worst case we commit error ϵ *at each step*; thus if we want to go N steps to time T, and *guarantee* a final error $\leq \epsilon$, we have to commit error $\epsilon' = \frac{\epsilon}{N}$ at each step. Fortunately, this estimate is quite pessimistic, because the error committed at one time is high frequency and therefore decays rapidly under the heat flow. Hence errors do not accumulate to this extent in practice.

There are several ways to deal with the problem of spreading support and they lead to a subdivision of Method 2.

Method 2.1 is the simplest. Here we plan to compute up to some final time T, determine R so that $|u(x,t)| \leq \epsilon$ outside $[-R, R]^2$ for $0 \leq t \leq T$, then compute on the fixed domain $[-R, R]^2$. Either R can be determined experimentally or the exact representation (18) for u can be used to bound R a priori. Suppose $|f| \leq \epsilon |f|_\infty$ when $|x| \geq \rho$. Then within error $\epsilon |f|_\infty$,

$$u(x,t) = \frac{1}{\pi\delta} \int_{|y|\leq\rho} e^{-|x-y|^2/\delta} f(y)\, dy, \qquad \delta = 4\Delta t.$$

If $|x| \geq R$ where $R \geq \rho$, then

$$|u(x,t)| \leq \frac{\rho^2}{\delta} e^{-(R-\rho)^2/\delta} |f|_\infty.$$

We can make this $\leq \epsilon$ for $0 \leq t \leq T$ with $R = \rho + O(\sqrt{T \log(\epsilon T)})$, so R grows rather slowly with T. This a priori bound is rather crude, however, and it is usually far cheaper to run the code once with $\Delta t = T$ and large ϵ (which costs very little), measure the maximum of $|u|$ near the boundary, and increase R until a suitable value is found. Note that increasing R adds to the outer regions of the computational domain, where u is small and smooth and therefore needs little refinement, hence adds very little to the cost of the calculation.

Method 2.2 is almost as simple as Method 2.1; here, we compute on a changing computational box $B(t) = [a_1(t), b_1(t)] \times [a_2(t), b_2(t)]$, outside of which $|f| \leq \epsilon$ initially. At each step, we expand $B(t)$ adaptively to include all regions in which $|u| > \epsilon$. This is done by computing the max of $|u|$ over each side and adding one cell at a time until satisfied.

This method is adaptive in a way that fits well with our general approach. It is not foolproof (as Method 2.1 with an a priori estimate would be), but in practice works very well. It achieves a small savings over Method 2.1 at a small additional cost in algorithmic complexity.

Method 2.3, like 2.2, expands the domain as necessary. But Method 2.2 adds cells, so the cost goes up; in Method 2.3, we simply scale up the size of each cell by increasing δ until cells of size $2r\sqrt{\delta}$ cover the region on which $|u| > \epsilon$. Thus we look at the solution on a larger spatial scale to take advantage of its increased smoothness and take larger timesteps accordingly. The spatial and temporal scalings are related by the natural scaling law of the heat equation; time = space squared.

This method achieves roughly constant cost, but at the price of varying the timestep. Thus we have to use interpolation in time if we want to know u at a specific time. Another disadvantage is that Method 2.3 does not easily extend to inhomogeneous problems, where the solution does not necessarily become smoother as time goes by.

Method 3 speeds up the evaluation of u by transforming the sum of $(2I+1)^2$ Hermite series into a single Taylor series for each cell, using a technique developed in [6]. We have

$$u(x,t) = \sum_C \sum_{\alpha\geq 0} C_\alpha h_\alpha \left(\frac{x-c}{\sqrt{\delta}}\right).$$

We can transform this to a single Taylor series in the cell B in which x lies, say,

$$u(x,t) = \sum_{\beta \geq 0} B_\beta \left(\frac{x-b}{\sqrt{\delta}} \right)^\beta,$$

where b is the center of B. To do this, we simply calculate the Taylor coefficients

$$\begin{aligned} B_\beta &= \frac{(\sqrt{\delta})^{|\beta|}}{\beta!} D_b^\beta u(b,t) \\ &= \frac{1}{\beta!} \sum_C \sum_{\alpha \geq 0} C_\alpha (\sqrt{\delta})^{|\beta|} D_b^\beta h_\alpha \left(\frac{b-c}{\sqrt{\delta}} \right). \end{aligned}$$

But $h_\alpha = (-D)^\alpha e^{-|x|^2}$, by definition, so

$$B_\beta = \frac{(-1)^{|\beta|}}{\beta!} \sum_C \sum_{\alpha \geq 0} C_\alpha h_{\alpha+\beta} \left(\frac{b-c}{\sqrt{\delta}} \right). \tag{23}$$

This sum can be truncated with an error bound similar to that for the original series. Thus u can be expressed as a local Taylor series in each cell, with coefficients depending on the Hermite coefficients C_α for nearby cells and the Hermite functions of the center-to-center vectors.

The advantage of this is that each evaluation of u costs only $O(p+1)^2$ arithmetic operations, rather than $O((2I+1)(p+1))^2$, which can be one or two orders of magnitude larger. Since most of the computational effort is spent on evaluating u, this is a considerable savings.

As in [6], there is a break-even point, a number of evaluations per cell below which it is not efficient to transform u to a Taylor series. This point depends on the degree of refinement necessary in the given cell, hence is not known a priori. However, it can be estimated from the behavior of u in the cell in the previous timestep or directly from the rate of decay of the Hermite coefficients.

However, another advantage is that when u is given by a truncated Taylor series, the coefficents C_α for the next time level are trivial to compute. For cell C, we have

$$\begin{aligned} C_\alpha &= \int_C \left(\frac{x-c}{\sqrt{\delta}} \right)^\alpha \sum_{\beta \leq p} B_\beta \left(\frac{x-c}{\sqrt{\delta}} \right)^\beta dx \\ &= \sum_{\beta \leq p} B_\beta \frac{r^{|\alpha+\beta|+2}(1-(-1)^{\alpha_1+\beta_1+1})(1-(-1)^{\alpha_2+\beta_2+1})}{(\alpha_1+\beta_1+1)(\alpha_2+\beta_2+1)} \end{aligned} \tag{24}$$

since the sides of C are $2r\sqrt{\delta}$ long. Thus it is unnecessary, as far as u is concerned, to retriangulate. Of course, inhomogeneous terms and layer potentials will still need to be approximated on a triangulation.

By substituting (23) into (24), we also get an expression for the new C_α's in terms of the old C_α's, which allows us to evolve u completely in transform space and makes it unnecessary to return to real space at every step.

5. The inhomogeneous case. We now generalize one of the methods, Method 2.1, to solve the inhomogeneous free-space heat equation

$$\begin{aligned} u_t &= \Delta u + F(x,t) \quad \text{in } \mathbf{R}^2, \\ u(x,0) &= f(x), \end{aligned} \tag{25}$$

where F has compact support in x for each t. The evolution formula corresponding to (20) is Duhamel's principle [5, p. 196]

$$u(x,t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta} u(y,t)\,dy + \int_t^{t+\Delta t} \frac{1}{\pi\sigma}\int_{\mathbf{R}^2} e^{-|x-y|^2/\sigma} F(y,s)\,dy\,ds \tag{26}$$

with $\delta = 4\Delta t$ as usual and $\sigma = 4(t+\Delta t - s)$.

Let us begin by discretizing the time integral. The integrand is a smooth function of s, so the trapezoidal rule

$$\int_t^{t+\Delta t} g(s)\,ds = \frac{\Delta t}{2}(g(t+\Delta t)+g(t)) + O(\Delta t^3)$$

is globally second-order accurate. Moreover, when $s \to t+\Delta t$, the integrand approaches $F(x,t+\Delta t)$. Thus we have

$$u(x,t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta}\left[u(y,t)+\frac{\Delta t}{2}F(y,t)\right]dy + \frac{\Delta t}{2}F(x,t+\Delta t) + O(\Delta t^3).$$

Thus $u(x,t+\Delta t)$ can be found by Method 2.1, if we simply replace $u(y,t)$ by $u(y,t) + \frac{1}{2}\Delta t F(y,t)$ in the construction of the triangulation and the evaluation of the moments, and afterward add $\frac{1}{2}\Delta t F(x,t+\Delta t)$. Since we commit $O(\Delta t^3)$ error at each of $N = \frac{T}{\Delta t}$ steps, the final error in u up to time T is $O(\Delta t^2)$.

Higher-order methods are easy to construct, but slightly more expensive per timestep. Several sets of coefficients must be stored, and updated (by the technique of Method 1) after each step. These higher-order methods are important, however, because our approach is efficient when Δt is *large*; thus we need a higher order of accuracy to capture variations in F. Hence this approach is suited to F varying rapidly in x but not in t.

A fourth-order method, for example, can be based on Simpson's rule:

$$\int_{t-\Delta t}^{t+\Delta t} \frac{\Delta t}{3}\left[g(t-\Delta t)+4g(t)+g(t+\Delta t)\right] + O(\Delta t^5).$$

This leads to two possible methods, depending on whether or not it is convenient to evaluate F at half timesteps. If not, for example, we get

$$u(x,t+\Delta t) = \frac{1}{2\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/2\delta}\left[u(y,t-\Delta t)+\frac{\Delta t}{3}F(y,t-\Delta t)\right]dy + \frac{4\Delta t}{3\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta}F(y,t)\,dy + \frac{\Delta t}{3}F(x,t+\Delta t) + O(\Delta t^5).$$

The final error after N steps of this method is $O(\Delta t^4)$.

6. Domains with boundaries. The methods we are developing in this paper find perhaps their best application in the solution of the heat equation on domains with (possibly time-dependent) boundaries. To do this, we use the classical theory of layer potentials for the heat equation described in [9] and more recently in [2]. The key operators in this theory are the layer potentials

$$Sg(x,t) = \int_0^t \int_{\Gamma(t')} G(x-x', t-t')g(x,t')\,dx'\,dt',$$

$$Dg(x,t) = \int_0^t \int_{\Gamma(t')} \frac{\partial G}{\partial n'}(x-x', t-t')g(x,t')\,dx'\,dt',$$

$$D^*g(x,t) = \int_0^t \int_{\Gamma(t')} \frac{\partial G}{\partial n}(x-x', t-t')g(x,t')\,dx'\,dt',$$

taken over a time-dependent boundary $\Gamma(t)$. Here G is the heat kernel $G(x,t) = (4\pi t)^{-1}e^{-|x|^2/4t}$ and g is a function defined on $\Gamma(t)$ for each t. Various boundary conditions for the heat equation can be transformed into Volterra integral equations on $\Gamma(t)$, with operators like S, D, or D^*, by representing the solution of the heat equation as a sum of layer potentials.

In this section, we describe briefly how our methods can be used to evaluate Sg. The basic idea is that layer potentials are solutions of inhomogeneous problems like the one treated in the previous section, but with distributions rather than smooth functions on the right-hand side as forcing terms. Sg, for example, solves

$$\begin{aligned} u_t &= \Delta u + \mu, \\ u(x,0) &= 0, \end{aligned} \tag{27}$$

where μ is the *measure* that assigns density g concentrated on $\Gamma(t)$. The double layer potential Dg has the normal derivative of a measure on the right-hand side.

Clearly this observation does not allow one to evaluate layer potentials with standard numerical methods that mostly require point values. Our methods, however, require only the ability to integrate the inhomogeneous term against a piecewise polynomial. The singularity in time of the kernel G turns out to complicate the integration slightly, requiring a straightforward asymptotic calculation.

Let us consider the single layer potential $u = Sg$. Since u satisfies (27), the evolution formula (26) gives

$$\begin{aligned} u(x,t) = {} & \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta}u(y, t-\Delta t)\,dy \\ & + \int_{t-\Delta t}^t \int_{\Gamma(t')} \frac{e^{-|x-y|^2/4(t-t')}}{4\pi(t-t')}g(y,t')\,dy\,dt'. \end{aligned} \tag{28}$$

The integrand in the time integral is singular; it becomes infinite as t' approaches t. Thus a standard integration formula like the trapezoidal rule will not achieve its usual order of accuracy. A standard technique for eliminating this difficulty is *product integration*; to apply product integration, one extracts an analytic form for the singularity of

$$\tilde{g}(x,t') = \int_{\Gamma(t')} \frac{e^{-|x-y|^2/4(t-t')}}{4\pi(t-t')}g(y,t')\,dy$$

as t' approaches t from below. Say, $\tilde{g}(x,t') \sim \phi(x,t')$ as $t' \uparrow t$. Then we construct an integration rule by interpolating $\tilde{g}(x,t')/\phi(x,t')$ and integrating the resulting polynomial

times $\phi(x, t')$ exactly. Such a rule will achieve the order of accuracy of the interpolation used. Of course, ϕ must be simple enough that ϕ times a polynomial can be integrated exactly or very accurately, by some trick.

Thus we need the asymptotic behavior of $\tilde{g}$ as $t' \uparrow t$. When x lies on $\Gamma(t)$, a simple calculation shows that

$$\tilde{g}(x, t') \sim \frac{1}{\sqrt{4\pi(t - t')}} g(x, t) \quad \text{as } t' \uparrow t.$$

This square-root singularity is consistent with the usual parabolic scaling of the heat equation. It is derived by approximating $\Gamma(t')$ by its tangent line, estimating the error, and Taylor expanding. The technique is fairly standard [1].

When x does not lie on $\Gamma(t)$, however, approximation by the tangent line gives the wrong answer, because it neglects the curvature of $\Gamma(t)$. A better approach is to approximate $\Gamma(t)$ by the osculating circle or parabola; the result depends only on the curvature of $\Gamma(t)$, so either will do. We omit the details and state the result.

Assume x_0 is the closest point on $\Gamma(t)$ to x and let $D = |x - x_0|$. Let R be the radius of curvature of $\Gamma(t)$ at x_0 and ρ be the distance from x to the center of curvature. Then, if $\rho > 0$,

$$\tilde{g}(x, t') \sim \sqrt{\frac{R}{\rho}} \frac{1}{\sqrt{4\pi(t - t')}} e^{-D^2/4(t-t')} g(x_0, t)$$

as $t' \uparrow t$.

We now have the asymptotic behavior of the integrand. The next step is to write the time integral as

$$\int_{t-\Delta t}^{t} \frac{e^{-D^2/4(t-t')}}{\sqrt{4\pi(t - t')}} h(x, t')\, dt'$$

$$h(x, t') = \int_{\Gamma(t')} \frac{e^{(D^2 - |x-y|^2)/4(t-t')}}{\sqrt{4\pi(t - t')}} g(y, t')\, dy$$

so that $h(x, t') \to \sqrt{R/\rho}\, g(x_0, t)$ as $t' \uparrow t$. Now h is a nice smooth function, so we replace it by its linear interpolant

$$\tilde{h}(x, t') = \frac{t - t'}{\Delta t} h(x, t - \Delta t) - \frac{t - \Delta t - t'}{\Delta t} h(x, t)$$

and carry out the time integral exactly. This gives

$$\int_{t-\Delta t}^{t} \frac{e^{-D^2/4(t-t')}}{\sqrt{4\pi(t - t')}} \tilde{h}(x, t')\, dt'$$

$$= \sqrt{\frac{\Delta t}{4\pi}} e^{-D^2/\delta} \left[\Gamma_0\left(-\frac{1}{2}, \frac{D^2}{\delta}\right) - \Gamma_0\left(-\frac{3}{2}, \frac{D^2}{\delta}\right) \right] \sqrt{\frac{R}{\rho}} g(x_0, t) \tag{29}$$

$$+ \sqrt{\frac{\Delta t}{4\pi}} \Gamma_0\left(-\frac{3}{2}, \frac{D^2}{\delta}\right) \int_{\Gamma(t-\Delta t)} \frac{e^{-|x-y|^2/\delta}}{\sqrt{\pi\delta}} g(y, t - \Delta t)\, dy + O(\Delta t^{5/2}).$$

Here Γ_0 is the normalized incomplete gamma function defined by

$$\Gamma_0(a, \xi) = e^{\xi} \xi^{-a} \int_{\xi}^{\infty} e^{-z} z^{a-1}\, dz \qquad a < 0.$$

We now have an approximate evolution formula

$$u(x,t+\Delta t) = \frac{1}{\pi\delta}\int_{\mathbf{R}^2} e^{-|x-y|^2/\delta}u(y,t)\,dy + W_1(x,t)g(x_0,t+\Delta t) + W_2(x,t)\int_{\Gamma(t)} \frac{e^{-|x-y|^2/\delta}}{\sqrt{\pi\delta}}g(y,t)\,dy,$$

accurate to $O(\Delta t^{5/2})$. Here W_0 and W_1 are given in (29) as functions of x, Δt, and $\Gamma(t)$. To evaluate u by this formula, we need to find the nearest point x_0 on $\Gamma(t+\Delta t)$ to x and estimate the curvature and normal of $\Gamma(t+\Delta t)$ at x_0. Finding the nearest point can be done efficiently by binning pieces of $\Gamma(t+\Delta t)$ and searching bins.

To evaluate u, we also need to evaluate the "Gauss transform on a curve" defined by

$$\Gamma_\delta g(x) = \int_\Gamma \frac{e^{-|x-y|^2/\delta}}{\sqrt{\pi\delta}}g(y)\,dy.$$

This can be approximated adaptively by the Hermite expansion technique used for the homogeneous case. We begin by representing Γ and g as piecewise polynomials of degree d, within accuracy ϵ. To do this, we first lay down a coarse grid on Γ (parametrized by $s \in [0,1]$, say), and construct piecewise polynomial interpolants to the coordinate functions of Γ and to g at meshpoints s_j equispaced in $[0,1]$. In the style of §3, we now refine our representation of Γ adaptively whenever the polynomial interpolation fails to represent either Γ or g accurately. The result is a representation

$$\Gamma = \cup_j \Gamma_j + O(\epsilon),$$
$$g = \tilde{g} + O(\epsilon),$$

where Γ_j is an element; if we use linear interpolation, for example, Γ_j is a line segment.

Now we apply the expansion technique of §2. Lay down cells C of size $2r\sqrt{\delta}$ to cover Γ, and expand the contribution of $\Gamma \cap C$ as a series about the cell center c. We find

$$\Gamma_\delta g(x) = \sum_C \sum_{\alpha\geq 0} C_\alpha h_\alpha\left(\frac{x-c}{\sqrt{\delta}}\right) + E$$

with $E = O(\epsilon)$ and

$$C_\alpha = \frac{1}{\sqrt{\pi\delta}}\int_{\Gamma\cap C}\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha \tilde{g}(y)\,dy$$
$$= \frac{1}{\sqrt{\pi\delta}}\sum_{\Gamma_j\subset C}\int_{\Gamma_j}\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha \tilde{g}(y)\,dy,$$

where $\Gamma = \cup\Gamma_j$. (In order to have Γ divided into elements each of which lies in a single cell C, we perform a further subdivision of Γ by cutting any element crossing more than one cell.) This Hermite series can be truncated with the usual bounds.

We now have to evaluate integrals of the form

$$\int_{\Gamma_j}\left(\frac{y-c}{\sqrt{\delta}}\right)^\alpha \tilde{g}(y)\,dy,$$

where $\tilde{g}$ is a polynomial. If we use linear interpolation, the recurrence relation for doing this was developed in §3. Otherwise, similar but more complicated recurrence relations can be developed, or Gauss–Legendre integration of sufficiently high order can be used.

A major advantage of the present approach is that we are able to use high-order approximations of Γ without having to integrate Gaussians over such approximations. The difficulty of carrying out these calculations has been a major stumbling block in the construction of high-order product integration schemes, both for the heat equation and the Laplace equation. With the current approach, one needs only to integrate powers over polynomial curve elements, a technique that can always be carried out. This advantage was observed but not developed in [6]. The fast multipole method [7] can be used in a similar way to eliminate the necessity of product integration in potential theory for the Laplace equation. This approach seems likely to be particularly useful in three-dimensional problems, where product integration is more difficult.

The remainder of the calculation is straightforward; we present some preliminary numerical results in §7. We have described only the single layer potential, but the analysis of the double layer potential is quite similar. The jump in the double layer potential across the boundary complicates matters surprisingly little.

7. Numerical experiments. In this section, we describe the results of codes written for the homogeneous problem and the single layer potential. The codes were written in standard FORTRAN 77 and run on a SUN SPARCstation 1+ with the optimizer flag `-O`, using double-precision arithmetic. The timings reported are usually reproducible within 1 or 2%, which is sufficient for our purposes.

First, we implemented Method 2.1 for the homogeneous problem, checking the numerical results against exact solutions for three sets of exact temperature fields; each is produced by shifting, scaling, and summing a basic solution. We put

$$u(x, y, t) = \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} u_j(k_1(x - x_{k_1}), k_1^2 t) u_j(k_2(y - y_{k_2}), k_2^2 t),$$

where x_{k_1} and y_{k_2} are random uniformly distributed on $[-1, 1]$. The three basic solutions are given by

$$u_1(x, t) = \frac{e^{1/4T} e^{-1/4}}{\sqrt{T}} \cos\left(\frac{x}{T}\right) e^{-x^2/T}$$

where $T = 1 + 4t$,

$$u_2(x, t) = \begin{cases} \max(1 - |x|, 0) & \text{if } t = 0, \\ \Lambda^2[\frac{1}{2}(x-1)\operatorname{erf}((x-1)/\sqrt{4t}) + \sqrt{t/\pi} e^{-(x-1)^2/4t}] & \text{if } t > 0, \end{cases}$$

where $\Lambda f(x) = f(x + 1) - f(x)$, and

$$u_3(x, t) = \begin{cases} \chi_{[-1,1]}(x) & \text{if } t = 0, \\ \frac{1}{2}[\operatorname{erf}((x+1)/\sqrt{4t}) - \operatorname{erf}((x-1)/\sqrt{4t})] & \text{if } t > 0, \end{cases}$$

where $\chi_{[-1,1]}(x)$ is zero for $|x| > 1$ and 1 otherwise. The first solution is smooth with K^2 sharp peaks of scales from 1 to $\frac{1}{K}$, and decays exponentially at infinity. The second is piecewise linear, continuous, Lipschitz, and piecewise smooth but not C^1 at $t = 0$. The third is discontinuous in x at $t = 0$ but smooth and sharply varying for $t > 0$. We used our method to compute u for ten timesteps equispaced from 0 to 1, beginning each triangulation with a 14×14 grid of square cells on the domain $[-7, 7]^2$ and using linear, quadratic, and cubic

interpolation with various error tolerances ϵ. We took $K = 10$, so each solution u_j varies over scales from 1 to 0.1. We used $p = 6$ and $I = 2$ to achieve error $\epsilon = 10^{-2}$ relative to $|u|_\infty$ at each step.

Table 1 reports the errors and times produced by Method 2.1 on these three temperature fields. Several conclusions can be drawn from Table 1. The method achieves the requested accuracy in every case. The time required for accuracy ϵ scales roughly like ϵ^{-1} for linear, $\epsilon^{-2/3}$ for quadratic, and $\epsilon^{-1/2}$ for cubic interpolation, as it should.

For purposes of comparison, the standard explicit nonadaptive finite difference method

$$u_{ij}^{n+1} = u_{ij}^n + \frac{\Delta t}{h^2}(\Lambda_i^2 + \Lambda_j^2)u_{i-1,j-1}^n$$

on a square $N \times N$ grid requires time $O(\epsilon^{-2})$ to attain accuracy ϵ. Indeed, this method has error $O(\Delta t + h^2)$ and is stable if $\Delta t \leq h^2/4$, so decreasing the error by a factor of 4 requires halving h and reducing Δt by 4. Since this method requires $O(h^{-2})$ work per step, we see that decreasing the error by a factor of 4 requires 16 times as much work, corresponding to $O(\epsilon^{-2})$. A fourth-order explicit method would need $O(\epsilon^{-1})$ work to achieve error less than ϵ.

Our method is relatively unfazed by nonsmooth or discontinuous initial temperature fields as long as they are bounded. It resolves the discontinuity as well as it can, and ends up with a very good approximation except on a very small area. The small area where the interpolant is inaccurate does not affect the total error, essentially because the heat equation is stable in L^1 as well as L^∞.

TABLE 1

Times and errors for the homogeneous free-space heat equation. Column N_0/N_1 displays the number of triangles in the triangulation at $t = 0$ and at $t = 1$, T is the total computing time for ten steps from $t = 0$ to $t = 1$, and E is the maximum error, divided by the maximum of the solution. Results for lower-degree interpolation with small ϵ, shown with dashes, were too time consuming and were omitted.

u_1	Linear			Quadratic			Cubic		
ϵ	N_0/N_1	T	E	N_0/N_1	T	E	N_0/N_1	T	E
10^{-1}	3198/784	52	.96−1	1076/394	66	.81−1	728/392	115	.76−1
10^{-2}	27634/5260	752	.43−2	3382/590	276	.18−2	1550/396	317	.17−2
10^{-3}	264454/49028	8812	.44−3	14460/1888	1079	.20−3	3948/532	628	.21 − 3
10^{-4}	–	–	–	64056/8126	5619	.20−4	11096/1236	1898	.14−4
10^{-5}	–	–	–	–	–	–	33496/3604	11178	.22−5
u_2	Linear			Quadratic			Cubic		
ϵ	N_0/N_1	T	E	N_0/N_1	T	E	N_0/N_1	T	E
10^{-1}	4548/784	26	.98−1	2188/394	32	.82−1	1410/392	55	.77−1
10^{-2}	54664/5242	568	.46−2	27340/588	318	.19−2	21746/396	446	.18−2
10^{-3}	399988/48982	7834	.48−3	316904/1884	2973	.22−3	249236/534	3833	.22−3
10^{-4}	–	–	–	399988/8152	7350	.21−4	399988/1242	7764	.14−4
10^{-5}	–	–	–	–	–	–	399988/3654	16443	.23−5
u_3	Linear			Quadratic			Cubic		
ϵ	N_0/N_1	T	E	N_0/N_1	T	E	N_0/N_1	T	E
10^{-1}	5106/782	25	.10−0	5290/394	41	.84−1	5344/392	80	.75−1
10^{-2}	14632/5266	457	.86−2	15032/590	213	.52−2	15060/394	309	.42−2
10^{-3}	115852/49016	6476	.95−3	115852/1878	1430	.18−2	115852/530	1779	.60−3

One feature that is not apparent from the tables is that evaluation of u is more costly than evaluation of the coefficients. Typically the code spends 80% of its time evaluating u and only 20% evaluating coefficients. This is partly due to inefficiency; the refinement test we use wastes many evaluations of u when the grid is almost completed. We plan to address this admittedly minor point in future improvements.

Finally, we present some preliminary numerical results for the evaluation of the single layer potential

$$Sg(x,t) = \int_0^t \int_{\Gamma(t')} \frac{e^{-|x-y|^2/4(t-s)}}{4\pi(t-s)} g(y,s)\,dy\,ds.$$

In order to compute the error, we took a very simple case with $g = 1$ and $\Gamma(t)$ a stationary circle with center (0, 0) and radius 1.1. (Of course, we coded the method for a general curve and density.) We computed Sg on an adaptive grid using N steps until $t = 1$, setting $\epsilon = 10^{-1}$ initially and reducing ϵ by a factor of 4 for each successive calculation. The triangulation with $\epsilon = 10^{-1}$ is shown in Fig. 6 at $t = 1$; the triangulation for this problem changes little over time.

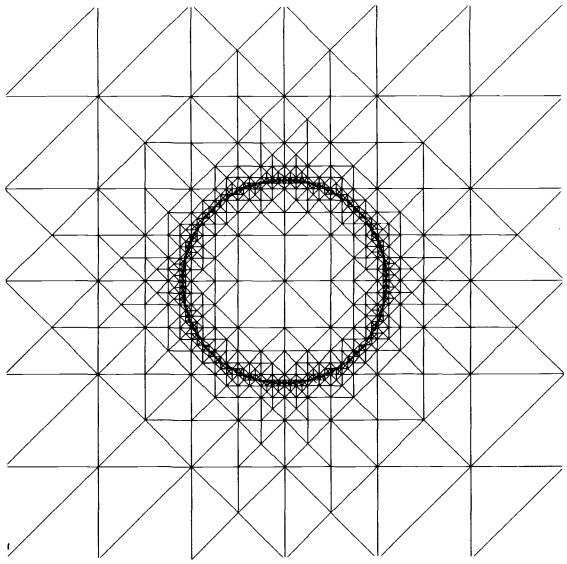


FIG. 6. *Triangulation for the single layer potential of unit density on a circle with center* (0, 0) *and radius* 1.1. *Here* $\epsilon = 10^{-2}$ *and* $t = 0.1$. *The smallest triangle has sides of length* 2^{-5} *times the maximum side length and there are* 1232 *triangles with* 5581 *nodes in the box* $[-3, 3]$. *The maximum level of subdivision was* 10. *The dotted circle is the curve* $\Gamma(t)$; *it is resolved to accuracy* $\epsilon' = 10^{-3}$ *with* 120 *line segments, subdivided further to have each line segment lie in a single cell.*

Table 2 presents numerical results for this calculation with $N = 10, 20, 40$ steps and cubic interpolation. T is the total computing time, E is the maximum error, divided by the maximum of the solution, and N_T is the number of triangles at $t = 1$.

TABLE 2

Times and errors for the single-layer heat potential; T is the total computing time for N steps from $t = 0$ *to* $t = 1$, *and E is the maximum error, divided by the maximum of the solution.*

Δt	h	p	I	N_T	T	E
0.1	1.33	8	1	218	79	.58−1
0.05	0.923	10	2	520	1104	.15−1
0.025	0.632	11	2	1220	6694	.37−2

We conclude that the method is expensive but accurate; perhaps its best feature is the elimination of product integration.

8. Conclusions. We have described an efficient adaptive approach to several heat flow problems that arise in physics. The simplest is the homogeneous free-space heat equation, for which we constructed several methods. All are based on the Hermite expansion of the heat kernel, combined with an adaptive triangulation scheme that represents a given function f as a piecewise polynomial within an error $\epsilon|f|_\infty$. Another key ingredient is the idea of writing the solution as a *function* that can be evaluated at any point, rather than a set of values to be interpolated.

We then demonstrated how to construct methods for the inhomogeneous case, using Duhamel's principle, and how to evaluate layer potentials. We treated layer potentials as inhomogeneous problems with distributional right-hand sides and treated the time singularities by product integration. The Hermite expansion is highly useful here, because it makes it unnecessary to carry out product integration in space. Thus even for high-order curve representations such as cubic splines, we still need only to evaluate integrals of *monomials* over the curve; we do not have to integrate Gaussians over a cubic spline curve, which is very difficult or impossible to do exactly. The Hermite series approach also combines the accuracy of product integration with the speed of the fast algorithm; usually product integration and fast algorithms do not marry well, because fast algorithms depend on processing the source independently of the location of the target then evaluating. In this case, the product integration simply *postmultiplies* the output of the fast summation technique.

Numerical results show these methods to be efficient and accurate. They perform particularly well in spatially rough problems where considerable accuracy is required.

The present techniques are formulated for the heat equation. It is shown in [4] how to extend fast techniques for the heat equation to nonlinear parabolic problems, and a similar technique works for variable-coefficient linear problems. Thus the techniques presented in this paper seem likely to be broadly applicable.

Appendix A. Cartesian coefficients. The polynomial of degree d that interpolates given values of a function f at the nodes shown in Fig. 1 (for $d = 1, 2, 3$) is usually expressed in terms of the barycentric coordinates $\lambda_1(x, y), \lambda_2(x, y), \lambda_3(x, y)$ defined by

$$\begin{aligned} \lambda_1 + \lambda_2 + \lambda_3 &= 1, \\ x_1\lambda_1 + x_2\lambda_2 + x_3\lambda_3 &= x, \\ y_1\lambda_1 + y_2\lambda_2 + y_3\lambda_3 &= y, \end{aligned} \tag{30}$$

with $(x_i, y_i) = v_i$ the vertices of the triangle. The barycentric coordinates of a point v can be computed in terms of the components x and y of v, by

$$\lambda_i = a_i x + b_i y + c_i, \tag{31}$$

by solving (30).

The $(d + 1)(d + 2)/2$ nodes required for interpolation of degree d are shown in Fig. 1. The interpolants of degrees $d = 1, 2, 3$ at these nodes are given by [3]

$$f(x) = \sum_{i=1}^{3} \lambda_i f(v_i) \qquad (d = 1),$$

$$f(x) = \sum_{i=1}^{3} \lambda_i(2\lambda_i - 1) f(v_i) + \sum_{i<j} 4\lambda_i\lambda_j f(v_{ij}) \qquad (d = 2),$$

$$f(x) = \frac{1}{2}\sum_{i=1}^{3} \lambda_i(3\lambda_i - 1)(3\lambda_i - 2) f(v_i) + \frac{9}{2}\sum_{i\neq j} \lambda_i\lambda_j(3\lambda_i - 1) f(v_{iij}) + 27 \sum_{i<j<k} \lambda_i\lambda_j\lambda_k f(v_{ijk}) \qquad (d = 3).$$

The numbering of the nodes is shown in Fig. 1 and follows [3].

In the linear case, it is straightforward to transform from barycentric to Cartesian coordinates:

$$f(x, y) = (a \cdot f)x + (b \cdot f)y + (c \cdot f),$$

where $a \cdot f = \sum_{i=1}^{3} a_i f(v_i)$, and so forth.

For the quadratic case, we find after some tedious calculations that

$$f(x, y) = \sum_{|\alpha| \le 2} F_\alpha x^\alpha = F_{20}x^2 + F_{11}xy + F_{02}y^2 + F_{10}x + F_{01}y + F_{00},$$

where the Cartesian coefficients F_{ij} are given by

$$F_{20} = \sum_{i=1}^{3} 2a_i^2 f_i + \sum_{i<j} 4a_i a_j f_{ij},$$

$$F_{11} = \sum_{i=1}^{3} 4a_i b_i f_i + \sum_{i<j} 4(a_i b_j + a_j b_i) f_{ij},$$

$$F_{02} = \sum_{i=1}^{3} 2b_i^2 f_i + \sum_{i<j} 4b_i b_j f_{ij},$$

$$F_{10} = \sum_{i=1}^{3} 2a_i(4c_i - 1) f_i + \sum_{i<j} 4(a_i c_j + a_j c_i) f_{ij},$$

$$F_{01} = \sum_{i=1}^{3} 2b_i(4c_i - 1) f_i + \sum_{i<j} 4(b_i c_j + b_j c_i) f_{ij},$$

$$F_{00} = \sum_{i=1}^{3} 2c_i(2c_i - 1) f_i + \sum_{i<j} 4c_i c_j f_{ij}.$$

Here we have abbreviated $f_i = f(v_i)$, $f_{ij} = f(v_{ij})$, and so forth.

The calculations for the cubic case are even more tedious and lead to

$$\begin{aligned} f(x, y) &= \sum_{|\alpha| \le 3} F_\alpha x^\alpha \\ &= F_{30}x^3 + F_{21}x^2 y + F_{12}xy^2 + F_{03}y^3 \\ &\quad + F_{20}x^2 + F_{11}xy + F_{02}y^2 + F_{10}x + F_{01}y + F_{00}, \end{aligned}$$

with

$$F_{30} = \frac{9}{2}\sum_{i=1}^{3} a_i^3 f_i + \frac{27}{2}\sum_{i \ne j} a_i^2 a_j f_{iij} + 27a_1a_2a_3 f_{123},$$

$$\begin{aligned} F_{21} &= \frac{27}{2}\sum_{i=1}^{3} a_i^2 b_i f_i + \frac{27}{2}\sum_{i \ne j} a_i(a_i b_j + 2b_i a_j) f_{iij} \\ &\quad + 27(a_1a_2b_3 + a_1b_2a_3 + b_1a_2a_3) f_{123}, \end{aligned}$$

$$\begin{aligned} F_{12} &= \frac{27}{2}\sum_{i=1}^{3} b_i^2 a_i f_i + \frac{27}{2}\sum_{i \ne j} b_i(b_i a_j + 2a_i b_j) f_{iij} \\ &\quad + 27(b_1b_2a_3 + b_1a_2b_3 + a_1b_2b_3) f_{123}, \end{aligned}$$

$$F_{03} = \frac{9}{2}\sum_{i=1}^{3} b_i^3 f_i + \frac{27}{2}\sum_{i\neq j} b_i^2 b_j f_{iij} + 27 b_1 b_2 b_3 f_{123},$$

$$F_{20} = \frac{9}{2}\sum_{i=1}^{3} a_i^2(3c_i - 1) f_i + \frac{9}{2}\sum_{i\neq j} a_i(3a_i c_j + (6c_i - 1)a_j) f_{iij}$$

$$+ 27(a_1 a_2 c_3 + a_1 c_2 a_3 + c_1 a_2 a_3) f_{123},$$

$$F_{11} = 9\sum_{i=1}^{3} a_i b_i(3c_i - 1) f_i + \frac{9}{2}\sum_{i\neq j}(6a_i b_i c_j + a_i(6c_i - 1)b_j + b_i(6c_i - 1)a_j) f_{iij}$$

$$+ 27(c_1(a_2 b_3 + a_3 b_2) + c_2(a_1 b_3 + a_3 b_1) + c_3(a_1 b_2 + a_2 b_1)) f_{123},$$

$$F_{02} = \frac{9}{2}\sum_{i=1}^{3} b_i^2(3c_i - 1) f_i + \frac{9}{2}\sum_{i\neq j} b_i(3b_i c_j + (6c_i - 1)b_j) f_{iij}$$

$$+ 27(b_1 b_2 c_3 + b_1 c_2 b_3 + c_1 b_2 b_3) f_{123},$$

$$F_{10} = \frac{1}{2}\sum_{i=1}^{3} a_i(27c_i^2 - 18c_i + 2) f_i + \frac{9}{2}\sum_{i\neq j}(a_i(6c_i - 1)c_j + c_i(3c_i - 1)a_j) f_{iij}$$

$$+ 27(a_1 c_2 c_3 + c_1 a_2 c_3 + c_1 c_2 a_3) f_{123},$$

$$F_{01} = \frac{1}{2}\sum_{i=1}^{3} b_i(27c_i^2 - 18c_i + 2) f_i + \frac{9}{2}\sum_{i\neq j}(b_i(6c_i - 1)c_j + c_i(3c_i - 1)b_j) f_{iij}$$

$$+27(b_1 c_2 c_3 + c_1 b_2 c_3 + c_1 c_2 b_3) f_{123},$$

$$F_{00} = \frac{1}{2}\sum_{i=1}^{3} c_i(3c_i - 1)(3c_i - 2) f_i + \frac{9}{2}\sum_{i\neq j} c_i(3c_i - 1)c_j f_{iij} + 27 c_1 c_2 c_3 f_{123}.$$

REFERENCES

[1] N. Bleistein and R. A. Handelsman, *Asymptotic Expansions of Integrals*, Dover, New York, 1986.

[2] R. Brown, *Layer Potentials and Boundary Value Problems for the Heat Equation on Lipschitz Cylinders*, Ph.D. thesis, Dept. of Mathematics, University of Minnesota, Duluth, 1987.

[3] P. G. Ciarlet, *Numerical Analysis of the Finite Element Method*, Seminaire de Mathématique Supérieures Été 1975, Université de Montréal, Montréal, 1976.

[4] J. Epperson, *Semi-group linearization for nonlinear parabolic equations*, preprint, 1991.

[5] G. B. Folland, *Introduction to Partial Differential Equations*, Mathematical Notes Number 17, Princeton University Press, Princeton, NJ, 1976.

[6] L. Greengard and J. Strain, *The fast Gauss transform*, SIAM J. Sci. Statist. Comput, 12 (1991), pp. 79–94.

[7] L. G. J. Carrier and V. Rokhlin, *A fast adaptive multipole algorithm for particle simulations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 669–686.

[8] W. F. Mitchell, *A comparison of adaptive refinement techniques for elliptic problems*, ACM Trans. Math. Software, 15 (1989), pp. 326–347.

[9] W. Pogorzelski, *Integral Equations and Their Applications*, Pergamon Press, Oxford, 1966.

[10] J. Strain, *The fast Gauss transform with variable scales*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1131–1139.

[11] G. Szegö, *Orthogonal Polynomials*, American Mathematical Society Colloquium Publications Vol. XXIII, American Mathematical Society, Providence, RI, 1939.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 207–224, January 1994

014

A NONLINEAR VARIATIONAL PROBLEM FOR IMAGE MATCHING*

YALI AMIT†

Abstract. Minimizing a nonlinear functional is presented as a way of obtaining a planar mapping that matches two similar images. A smoothing term is added to the nonlinear functional to penalize discontinuous and irregular solutions. One option for the smoothing term is a quadratic form generated by a linear differential operator. The functional is then minimized using the Fourier representation of the planar mapping. With this representation the quadratic form is diagonalized. Another option is a quadratic form generated via a basis of compactly supported wavelets. In both cases, a natural approximation scheme is described. Both quadratic forms are shown to impose the same smoothing. However, in terms of the finite dimensional approximations, it is easier to accommodate local deformations using the wavelet basis.

Key words. image matching, movement compensation, nonlinear variational problem, spectral methods, wavelets

AMS subject classifications. 68V10, 65M70, 49N60

1. Introduction.

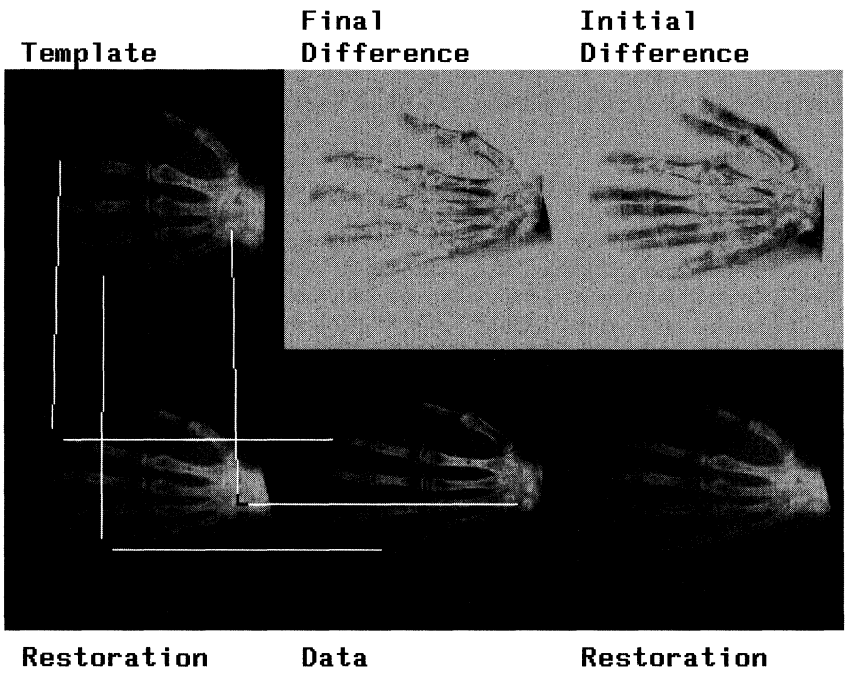
Let F and G be smooth functions on the two-dimensional unit square I^2, and let $\phi(x)$ be a mapping of the unit square into itself, such that $G(x) = F(\phi(x))$. It is clear that if F and G have the same range, many such mappings exist, most of which would be highly discontinuous and degenerate and not very interesting. However, if the mapping $\phi(x)$ is a smooth diffeomorphism of the unit square onto itself, then the extremal points of F would be mapped via ϕ^{-1} onto the extremal points of G, level curves would be mapped onto level curves, etc. Heuristically speaking, the graphs of F and G considered as surfaces would have similar topographies. Conversely, if F and G have similar topographies, then it should be possible to find a smooth and locally nondegenerate mapping ϕ such that $F(\phi(x))$ is close to G in some sense.

To illustrate this idea, consider the images in Fig. 1, which are x-rays of two different hands. If we consider the images as some smooth function sampled at the points of the pixel lattice, we obtain two functions that indeed have very similar topographies. This would be the case with any two images of some fixed organ of the body of two different patients, or of the same patient obtained at different times, provided that these images came from the same type of imaging device. Consequently, there should exist a smooth and locally nondegenerate mapping ϕ that transforms one image, called the *template*, into the other image, called the *data*, via composition. The mapping ϕ would automatically match between the corresponding parts of the two images. See, for example, Fig. 1 where the various parts of the hand such as the tips of the fingers or the joints are correctly matched.

One of the first attempts dealing with the issue of image matching can be traced to Horn and Schnuck [5] and Huang and Tsai [6] in the context of optical flow and movement compensation calculations for sequences of images. These ideas were further developed by Nagel [10] and Terzopoulos [11]. In Bajcy and Kovacic [2] these ideas were applied to the issue of matching medical images of similar organs, such as MRI images of the brain. Here the matching is not intended to calculate movement, but to automate the analysis of medical images. This second problem is also more difficult in that large deformations may occur, as opposed to relatively small deformations in image sequences.

*Received by the editors April 1, 1991; accepted for publication (in revised form) March 1, 1993. This research was supported in part by Office of Naval Research grant N00014-88-K-0289 and Army Research Office grant DAAL03-90-G-0033.

†Department of Statistics, University of Chicago, Chicago, Illinois 60637. This work was done while the author was at Brown University, Division of Applied Mathematics, Providence, Rhode Island 02912 (amit@galton.uchicago.edu).



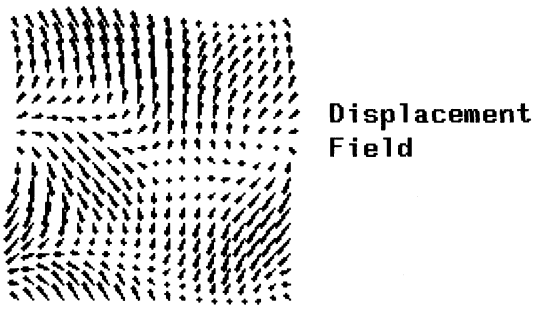


FIG. 1. *The template corresponds to the function F and the data to the function G. The restoration was done using the Fourier method. The white lines show the matching induced by the displacement field. A point x in the data is connected via the restoration to the point $x + \overline{U}(x)$ from which it obtained its grey level value. The final difference image shows $1 - |F(x + \overline{U}(x)) = G(x)|$ and the initial difference image shows $1 - |F(x) - G(x)|$.*

The question is how to find the mapping ϕ. We have addressed this problem by minimizing the following functional:

$$I(U) = \frac{1}{2}\int_{T^2} |F(x + U(x)) - G(x)|^2 dx, \tag{1}$$

where $U(x) = \phi(x) - x$ is the displacement field and T^2 is the unit torus. Minimizing I over some set of vector fields provides a mapping $\phi(x) = x + U(x)$ of the torus into itself, such that $F \circ \phi(x)$ is close in the mean square norm to G.

It should be noted that the periodic domain is chosen for the sake of notational and computational convenience. It takes care of the problem of what to do when $x + U(x)$ is not

in I^2. Another possibility is that F is defined on some large domain D that includes I^2. Then one would want to minimize over mappings from I^2 into D.

To rule out the discontinuous and irregular solutions to this minimization problem, it is possible to introduce a smoothing or regularizing term, thus obtaining a new functional

$$J(U) = \alpha \mathcal{E}(U, U) + \frac{1}{2} \int |F(x + U(x)) - G(x)|^2 dx, \tag{2}$$

where $\mathcal{E}$ is a bilinear form penalizing nonsmooth functions. In all our applications, $\mathcal{E}$ was taken to be a Hilbertian norm equivalent to one of the Sobolev norms. Since the domain under consideration is two dimensional, taking the Sobolev norm to be of order greater or equal to two will ensure that the solutions are continuous. Higher-order Sobolev norms will, of course, introduce additional smoothness. This approach was first described in a statistical setting by Amit, Grenander, and Piccioni [1].

The functional J is nonlinear and may have many global and local minima. In the sequel we will be interested mainly in finding local minima of J close to the initial point $U(x) \equiv 0$ *that corresponds to the identity map*. The nondegeneracy of the mapping $\hat{\phi}$ generated by a local minimizer $\hat{U}$ is then ensured by the fact that it is close to the identity map so that its Jacobian is nonzero at most points.

There are several major differences between the work mentioned above and the approach presented here. First, we do not use a data term derived from intensity conservation assumptions originally suggested by Horn and Schnuck [5], which is equivalent to linearizing the functional I as described at the end of §2. It appears that the linearized problem will not capture larger deformations (see Fig. 2).

Second, the solution of the variational problem is obtained by parametrizing the unknown function in terms of its coefficients with respect to either the Fourier basis or some wavelet basis, thus allowing for a coarse-to-fine or multiresolution approach. This was indeed suggested in [11] using multigrid techniques, which may be appropriate for the linearized equations that have a unique solution and are known to be efficiently soluble using multigrid techniques. However, given that the nonlinear functional is to be used, and that this nonlinear functional is not convex, it is not clear how well the classical multigrid approach will perform. The coarse level displacement is calculated using only the information of a smooothed version of the data on that same coarse grid, and there is some risk of information being lost. Moreover, when moving to finer grids, a bilinear interpolation is used that may not be smooth enough and that may introduce unnatural deformations.

Setting the problem in terms of an orthonormal basis directly incorporates interpolation through the basis functions. The smoothing operator is automatically written in diagonal form in terms of the basis chosen. Thus using the description in terms of a basis expansion, and solving first for low-frequency coefficients, gradually increasing the number can be thought of as a multigrid method translated onto the finest grid. Although some computational speed is lost, the advantage is that all the data is used to drive the algorithm.

The level of smoothness versus locality can be controlled by the choice of wavelet basis. Since the problem at hand is not really governed by physical fluid dynamical or elasticity laws, there is no special advantage in using the Laplacian as a smoothing operator. The existence of fast transforms for these bases makes the algorithm computationally feasible.

In §2 the smoothing term $\mathcal{E}$ is set to be a quadratic form generated by a linear differential operator. The approximations are then described together with minimization procedure. The basic idea is to diagonalize the differential operator using the Fourier basis and to solve the problem in the spectral domain.

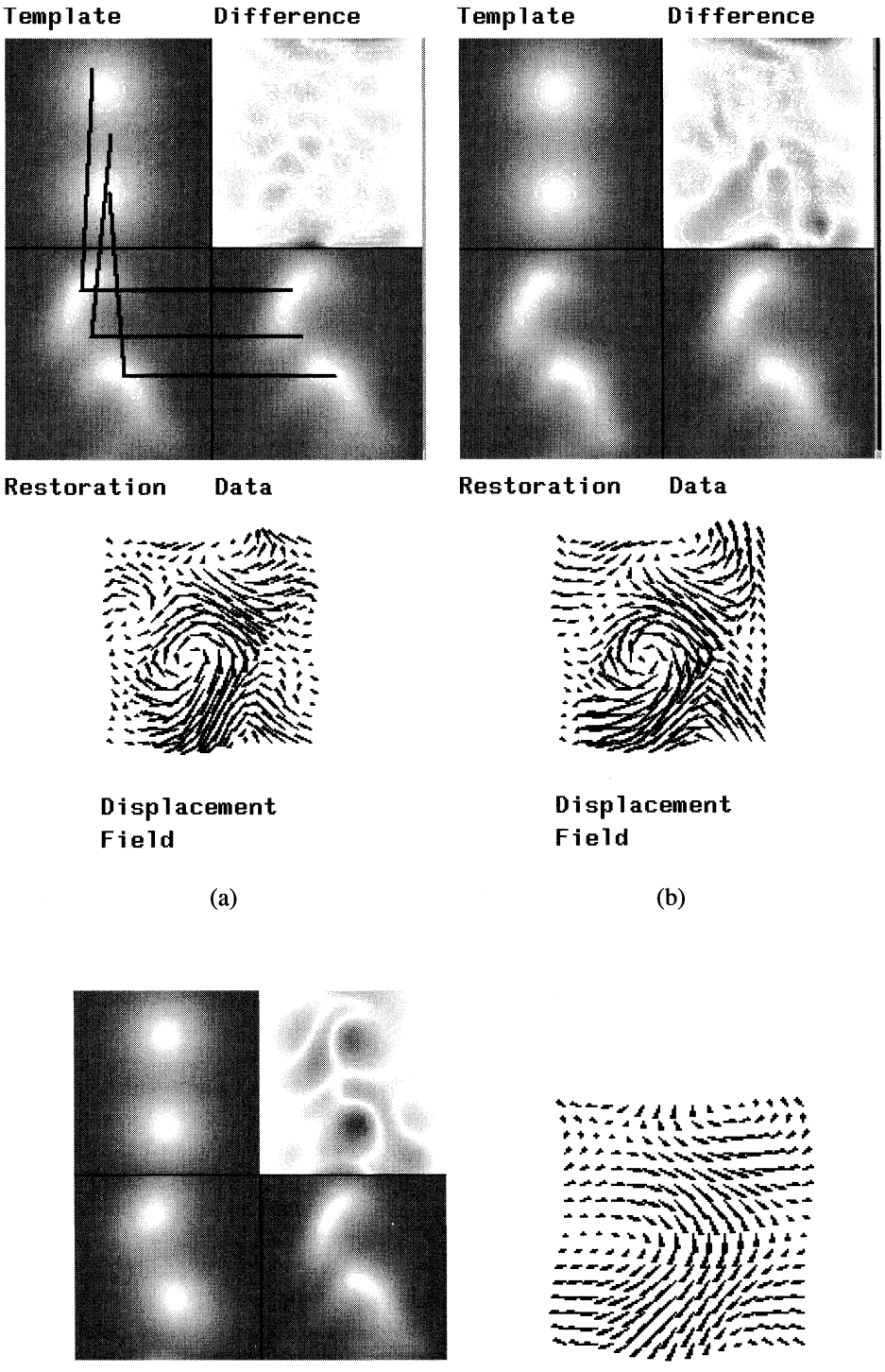


FIG. 2. *A transformation constructed with the Fourier basis was initially applied to the template to generate the data.* (a) *was done using the Fourier basis.* (b) *was done using the wavelet basis.* (c) *was done using the linearized equations.*

In §3 an alternative smoothing term is suggested. This time $\mathcal{E}$ is directly given in a diagonal form using a wavelet basis instead of the Fourier basis. The eigenvalues are set so as to ensure the same type of smoothing.

In §4 the experiments are described, and the performance of the two approaches is compared.

2. A nonlinear partial differential equation. Consider the bilinear form

$$B(f, g) = \frac{1}{2}\int_{T^2} (\Delta + \varepsilon)^2 f(x) \cdot g(x)dx,$$

where Δ denotes the Laplacian with periodic boundary conditions. This bilinear form defines a Hilbertian norm equivalent to the standard Sobolev norm on $H = H^2(T^2)$. Set $\varepsilon(U, U) = B(U^{(1)}, U^{(1)}) + B(U^{(2)}, U^{(2)})$ in (2). With this choice of the bilinear form in the functional J, the Euler equation for the minimizer is the following nonlinear partial differential equation (PDE):

$$\alpha(\Delta + \varepsilon)^2 U^{(1)} = (G(x) - F(x + U(x)))\frac{\partial F}{\partial x_1}(x + U(x)),$$
$$\alpha(\Delta + \varepsilon)^2 U^{(2)} = (G(x) - F(x + U(x)))\frac{\partial F}{\partial x_2}(x + U(x)).$$

The parameter α determines the relative weight of the regularizing term. Since the issue of the choice of α is not addressed here, we set $\alpha = 1$. This is the parameter used in the experiments as well.

For the purpose of numerical solutions, it is, of course, necessary to find finite-dimensional approximations to the functional J. Since the spectral decomposition of Δ is known, it is convenient to write J in the spectral domain and then approximate it coordinatewise. Let λ_{kl}, ϕ_{kl} denote the eigenvalues and eigenvectors of Δ, then

$$\lambda_{kl} = (2\pi)^2(k^2 + l^2) + \varepsilon \quad \text{and} \quad \phi_{kl}(x_1, x_2) = e^{2\pi i(kx_1 + lx_2)},$$

and the functional J can be rewritten as

$$J(U) = \sum_{k,l=-\infty}^{\infty} \lambda_{kl}^2((u_{kl}^{(1)})^2 + (u_{kl}^{(2)})^2)$$
$$+ \int \left[F\left(x_1 + \sum u_{kl}^{(1)}\phi_{kl}(x), x_2 + \sum u_{kl}^{(2)}\phi_{kl}(x)\right) - G(x)\right]^2 dx$$
$$= \Gamma(u^{(1)}, u^{(2)}) + q(u^{(1)}, u^{(2)}),$$

where $u_{kl}^{(i)} = \int U^{(i)}(x)\phi_{kl}(x)dx$. The vector $(u^{(1)}, u^{(2)}) \in \ell_2 \times \ell_2$, is simply the coordinate vector of $(U^{(1)}, U^{(2)})$ with respect to the basis ϕ_{kl} of $L_2(T^2)$. We write $(u^{(1)}, u^{(2)}) = \pi(U^{(1)}, U^{(2)})$. Note that $q(u^{(1)}, u^{(2)}) = I(U^{(1)}, U^{(2)})$ with I as in (1).

The finite dimensional approximations of the functional are obtained by taking the sums in the linear term and those in the integrand between $-(N-1)$ and N. The approximation is therefore obtained by restricting the argument of J to the space $H_N \times H_N$, where $H_N = \operatorname{span}\{\phi_{kl}\}_{k,l=-(N-1)}^{N}$. The dimension of the approximate space H_N is $(2N)^2$ and $\pi(H_N \times H_N) = R^{(2N)^2} \times R^{(2N)^2}$.

Let J_N denote the approximate functional on $H_N \times H_N$. Each of the finite dimensional functionals is positive and continuous. Moreover, $J_N(U) \to \infty$ as $U \to \infty$ and therefore has at least one global minimum. Let S_N be the set of global minima of J_N.

THEOREM. *Let $U_N \in S_N$ for $N = 1, 2, \ldots$. Then U_N has a convergent subsequence in $H \times H$, which converges to a global minimum of J.*

Proof. Consider $H \times H$ with the Hilbertian norm defined by $\mathcal{E}(\cdot, \cdot)$. Since $H \times H$ is compactly imbedded in $C(T^2) \times C(T^2)$, $|U_N - U|_{H\times H} \to 0$ implies uniform convergence that, in turn, implies that $G(x + U_N(x)) \to G(x + U(x))$, as $N \to \infty$. Since G and F are bounded, it follows from the dominated convergence theorem that $I(U_N) \to I(U)$. Clearly, $\mathcal{E}(U_N, U_N) \to \mathcal{E}(U, U)$, so that J is continuous in H.

Now $J_N(U_N)$ is a positive monotonically decreasing sequence so that it converges to some value L. Moreover, since U_N is the global minimum of J in $H_N \times H_N$, and since $\cup_{N=1}^{\infty}(H_N \times H_N)$ is dense in $H \times H$, it follows from the continuity of J that $L = \inf_{U\in H\times H} J(U)$.

Since $|U_N|_{H\times H} = \mathcal{E}(U_N, U_N) < J_N(U_N) < J_1(U_1)$ for all $N = 1, 2, \ldots$ the sequence U_N is weakly compact. Let U_{N_k} be a subsequence that converges weakly to some $U \in H \times H$. Again, since H is compactly embedded in $C(T^2)$, U_{N_k} converges in the uniform norm to U as $k \to \infty$. As above, this implies that $I(U_{N_k}) \to I(U)$. Together with the fact that $J(U_{N_k})$ converges, we conclude that $\mathcal{E}(U_{N_k}, U_{N_k})$ converges to $L - I(U)$ as $k \to \infty$.

Now since the norm function is lower semicontinuous in the weak topology, we have $\mathcal{E}(U, U) \le L - I(U)$, and since L is the infimum, we have $\mathcal{E}(U, U) + I(U) = L$ and $\mathcal{E}(U, U) = \lim_{k\to\infty} \mathcal{E}(U_{N_k}, U_{N_k})$. This, together with the weak convergence of U_{N_k} to U, implies strong convergence in H. Finally, since $J(U) = L$, it is a global minimum of J. □

Observe that by using the same arguments as above, it is possible to show that the set of global minima of J is compact in H.

Practical considerations. In practice, the template and the data are images and are given only on a discrete pixel lattice of equally spaced points $x_{\alpha\beta}$, $\alpha, \beta = 0, \ldots, L-1$ in I^2. The unknown displacement field U is given in terms of its array of values at the points of the lattice, which will also be denoted by U. We write $U_{\alpha\beta} = U(x_{\alpha\beta})$ for all $\alpha, \beta = 0, \ldots, L-1$. The approximate functional J_N now has the form

$$J_N(U) = \sum_{k,l=-(N-1)}^{N} \lambda_{kl}((u_{kl}^{(1)})^2 + (u_{kl}^{(2)})^2) + \frac{1}{L^2}\sum_{\alpha,\beta=0}^{L-1}\left[F(x_{\alpha\beta} + U_{\alpha\beta}) - G(x_{\alpha\beta})\right]^2. \tag{3}$$

The array $u_{k,l}$, $k, l = -(N-1), \ldots, N$ is the discrete Fourier transform of the array U, i.e.,

$$u_{kl}^{(i)} = \frac{1}{L^2}\sum_{\alpha,\beta=0}^{L-1} U_{\alpha\beta}^{(i)}\phi_{kl}(x_{\alpha\beta}),$$

and we write $u^{(i)} = \pi(U^{(i)})$. Since F is actually given only on the lattice, and $x_{\alpha\beta} + U_{\alpha\beta}$ may not lie on the lattice, it is possible either to truncate to the nearest point or to use a linear interpolation between the four nearest points. The resolution of the pixel lattice is the finest, so that these corrections are negligible.

We have not tried to find the global minimum of the approximating functional. Instead, we have done gradient descent starting at initial point zero. In other words, the following ordinary differential equation (ODE) in $R^{(2N)^2} \times R^{(2N)^2}$ was solved.

$$\begin{aligned}\frac{du_{kl}^{(i)}(t)}{dt} &= -\lambda_{kl}u_{kl}^{(i)} \\ &\quad -\frac{1}{L^2}\sum_{\alpha,\beta=0}^{L-1}\left[F(x_{\alpha\beta} + U_{\alpha\beta}(t)) - G(x_{\alpha\beta})\right]\frac{\partial F}{\partial x_i}(x_{\alpha\beta} + U_{\alpha\beta}(t))\phi_{kl}(x_{\alpha\beta}).\end{aligned} \tag{4}$$

Writing

$$Z_{\alpha\beta}^{(i)}(t) = \left[F(x_{\alpha\beta} + U_{\alpha\beta}(t)) - G(x_{\alpha\beta})\right] \frac{\partial F}{\partial x_i}(x_{\alpha\beta} + U_{\alpha\beta}(t)), \tag{5}$$

the second term in (4) is simply $(\pi(Z^{(i)}(t)))_{kl}$. The choice of initial point zero is motivated by the fact that the mapping generated by the solution is expected to be in some neighborhood of the identity map. The symmetric difference approximation of the derivatives of F was used.

It was found that minimizing all $2(2N)^2$ coefficients at once for large N is not the best approach. It might be preferable to start out in a low-dimensional space and gradually to increase the dimension until the desired maximum dimension is reached. The initial point in each space is then taken as the local minimum obtained in the previous space. This procedure is also faster. Heuristically, this can be interpreted as matching global features in low-dimensional spaces and moving on to the finer details as the dimension increases.

The algorithm is described as follows.

(i) Set $N = 0$, initial condition $u(0) = 0$.

(ii) Find local minimum $\hat{u}_N$ for J_N, starting from $u(0)$.

(iii) Set $N \to N + 1$ and $u_{kl}^{(i)}(0) = \begin{cases} (\hat{u}_N^{(i)})_{kl} & \text{for } -N < k, l \leq N, \\ 0 & \text{otherwise.} \end{cases}$

(iv) Go to (ii).

Step (ii) is carried out by a simple Euler method.

(a) Fix time step dt, set $t = 0$.

(b) Generate $U(t)$ from $(u^{(1)}(t), u^{(2)}(t))$-inverse DFT, $U^{(i)} = \pi^{-1}(u^{(i)})$ or

$$U_{\alpha\beta}^{(i)}(t) = \sum_{k,l=-N+1}^{N} u_{kl}^{(i)}(t)\phi_{kl}(x_{\alpha\beta}), \qquad \alpha, \beta = 0, \ldots, L-1.$$

(c) Carry out quadrature (in (4)) for each eigenvector ϕ_{kl}-DFT, $V^{(i)}(t) = \pi(Z^{(i)}(t))$ or

$$V_{kl}^{(i)}(t) = \frac{1}{L^2} \sum_{\alpha,\beta=0}^{L-1} F(x_{\alpha\beta} + U_{\alpha\beta}(t)) \frac{\partial F}{\partial x_i}(x_{\alpha\beta} + U_{\alpha\beta}(t))\phi_{kl}(x_{\alpha\beta}).$$

(d) Add the linear term and carry out Euler step.

$$u_{kl}^{(i)}(t+1) = u_{kl}^{(i)}(t) - dt \cdot \left[V_{kl}^{(i)}(t) + \lambda_{kl} u_{kl}^{(i)}(t)\right].$$

If difference is smaller than tolerance, set $\hat{u}_N = u(t+1)$ and go to (iii).

(e) Go to (b).

The time-consuming parts of the algorithm are steps (b) and (c). One option is to apply a fast Fourier transform (FFT) and an inverse FFT. The dimensions of these FFTs are determined by the size of the pixel lattice on which F and G are defined, i.e., 128×128 or 256×256. However, when N is very small, or when the dimensions of the pixel lattice are not powers of two, it may be faster to actually carry out the quadrature in step (c) for those frequencies that are being updated and to carry out the summation in step (b) for all points (α, β) on the lattice.

In some cases it is possible to carry out the quadrature in (c) on a coarser lattice than the original pixel lattice; then, of course, the summation in (b) is only done for the points on the coarse lattice. This is particularly true when F and G are smooth functions. This is related to the multigrid approach suggested in [11]. However, in [11] the number of unknowns is

equal to the number of points used and this seems to be insufficient. This is also the case for the pseudospectral methods described below. In the experiments described here the algorithm did not produce much improvement for $N > 10$, and at low frequencies the coarse lattice quadrature method is indeed faster than the FFT method.

In some experiments it was useful to actually smooth out the template and the data through some low-pass filter (see Figs. 3 and 4). This tends to single out global topological features and eliminate local ones. In this case the coarse lattice quadrature is very appropriate. For images 128×128, one could work with a 16×16 lattice for the first four or five frequencies and get good results.

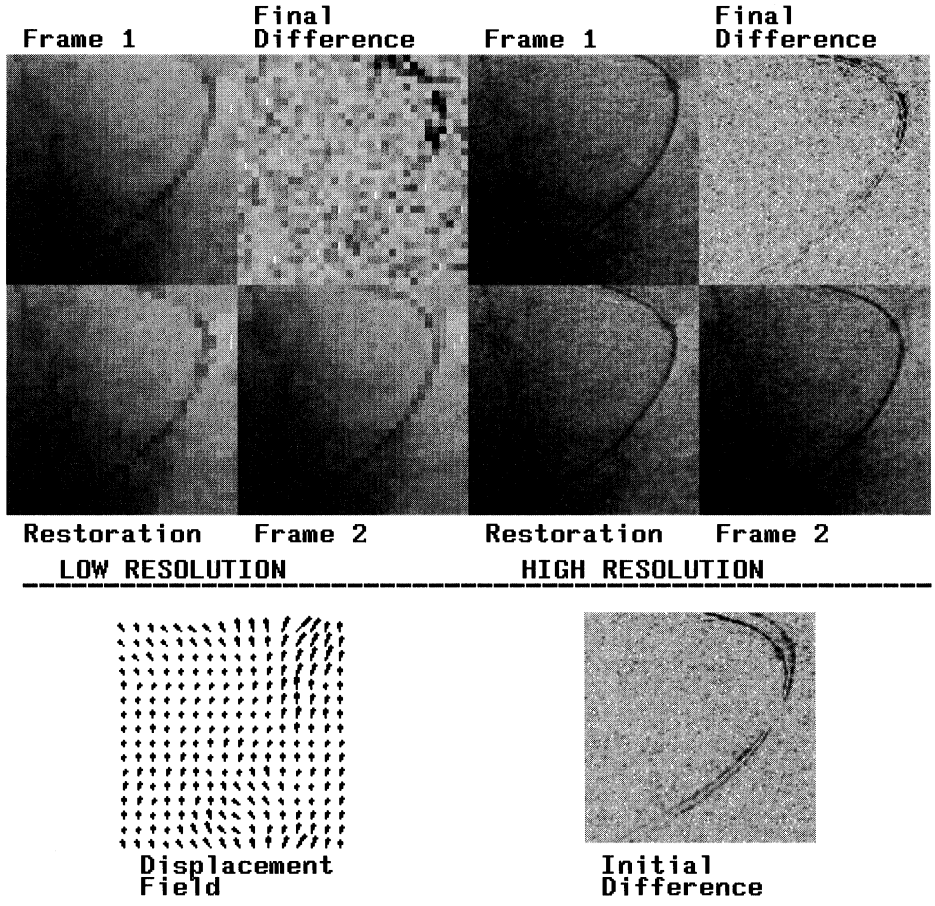


FIG. 3. *Frames* 1 *and* 2 *are two subsequent pictures from an x-ray of a catheter inserted into an artery. Frame* 1 *is used as the template and Frame* 2 *as the data. The displacement field is found by using the wavelet method at the low resolution, and then applied to the high-resolution template (Frame* 1*). The black line in the restoration is almost perfectly aligned with the black line in the second frame.*

If the full lattice has $L \times L$ pixels and the coarser lattice has $K \times K$ pixels, the number of floating-point multiplications used in this method is of order $K^2 \cdot N^2$, with N being the number of frequencies, whereas using the full FFT would be approximately $L^2 \cdot \log L^2$. Thus, depending on the number of frequencies to be used, it is possible to choose which is the most appropriate. As a rule, K, the number of quadrature points, should be greater than N, the

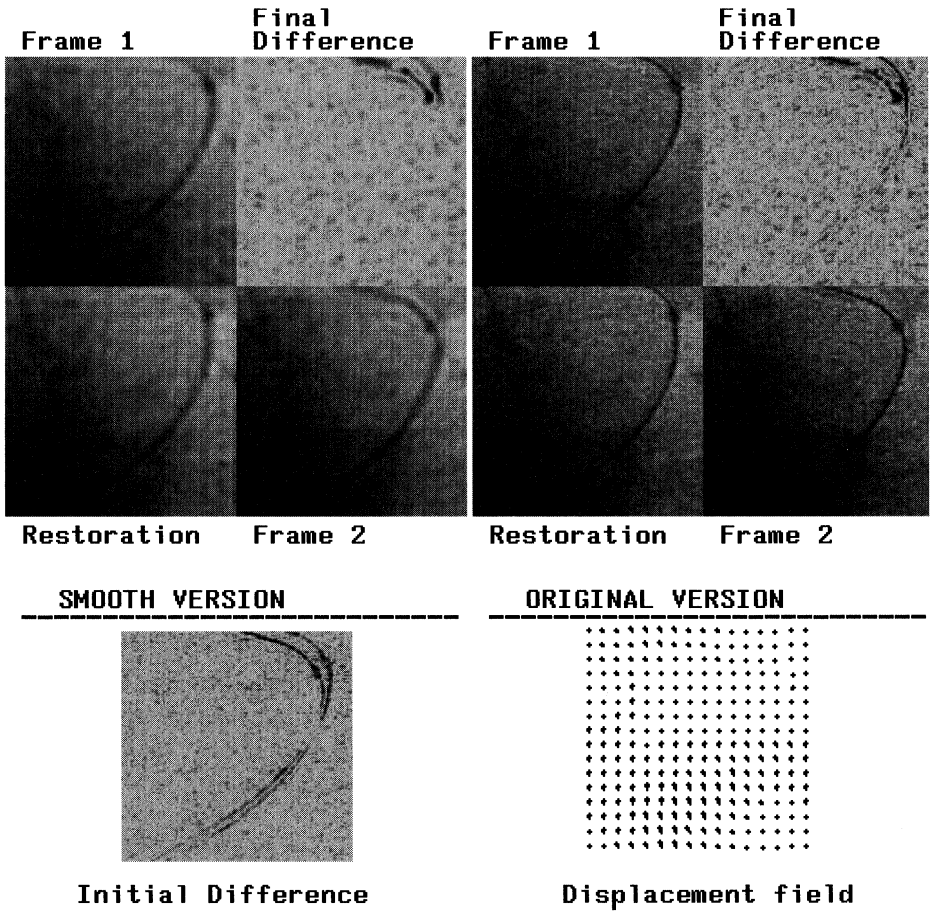


FIG. 4. *The same setting as in Fig.* 3. *The restoration is done using the Fourier basis. The line in the restoration is not aligned at the upper right-hand corner.*

number of frequencies being solved. For high frequencies the coarse lattice approximation is therefore not suitable, in which case it is clearly advantageous to use the FFT.

Another method is the pseudo-spectral method described by Gottlieb, Hussaini, and Orszag in [4]. Since one does not expect to solve, say, for $N > K$ (namely, the solution that is assumed to be in $H_K \times H_K$) using the periodic version of the sampling theorem, one can write

$$U^{(i)}(x) = \sum_{\alpha,\beta=0}^{K-1} U_{\alpha\beta}^{(i)} S_{\alpha\beta}(x),$$

where $S_{\alpha\beta}(x)$ are trigonometric polynomials and $x_{\alpha\beta}$, $\alpha, \beta = 0, \ldots, K-1$ are equally spaced points on the unit square called the collocation points. Thus

$$\Delta^2 U^{(i)}(x) = \sum_{\alpha,\beta=0}^{K-1} U_{\alpha\beta}^{(i)} \Delta^2 S_{\alpha\beta}(x).$$

If the quadrature in (3) is carried out on the $K \times K$ lattice of collocation points, then the

gradient descent equation has the form

$$\frac{dU_{(i)\gamma\delta}(t)}{dt} = -\sum_{\alpha\beta} U^{(i)}_{\alpha\beta}(t)(\Delta^2+\varepsilon)S_{\alpha\beta}(x_{\gamma\delta}) - \left[F(x_{\gamma\delta}+U_{\gamma\delta}(t)) - G(x_{\gamma\delta})\right]\frac{\partial F}{\partial x_i}(x_{\gamma\delta}+U_{\gamma\delta}(t)),$$

for $\gamma, \delta = 0, \ldots, K-1$. Now since the minimization is taking place in the space domain, the coupling between the equations occurs in the smoothing term and not in the data term. The derivatives of the functions S at all collocation points can be calculated off-line and stored as a matrix.

The discrete time iterations for this ODE require only one time-consuming step, the summation, as opposed to the previous schemes that had two time-consuming steps. If a $K \times K$ lattice of collocation points is used, the number of floating-point multiplications is approximately K^4.

The disadvantage of this method lies in the fact that the number of frequencies being updated is equivalent to the number of collocation points used. Thus if only very low frequencies are to be updated, very few collocation points are used and they are insufficient to obtain a good result. On the other hand, if more collocation points are used, the solution obtained involves higher frequencies at the start and this might be undesirable. There is also no natural way to gradually increase the dimension of approximation.

As mentioned in the Introduction, a further possibility would be to linearize the functional by substituting $F(x) + \nabla F(x) \cdot U(x)$ for $F(x + U(x))$ in $I(U)$. Then we obtain a quadratic form in $U(x)$, or in the coefficients u_{kl}, which has a unique minimum. The Euler equations for this functional are precisely those suggested by Horn and Schnuck [5]:

$$\alpha(\Delta+\varepsilon)^2 U^{(1)} + (\nabla F \cdot U)\frac{\partial F}{\partial x_1} = (G(x)-F(x))\frac{\partial F}{\partial x_1},$$
$$\alpha(\Delta+\varepsilon)^2 U^{(2)} + (\nabla F \cdot U)\frac{\partial F}{\partial x_2} = (G(x)-F(x))\frac{\partial F}{\partial x_2}.$$

Experiments with this option have not led to satisfactory results; see §4 and Fig. 2.

The disadvantage of the various spectral methods described above is the use of the Fourier basis whose functions have global support. This makes local changes in U or in the mapping ϕ difficult to achieve. A very attractive alternative is the wavelet basis.

3. The wavelet method. Using a periodic wavelet basis obtained from certain types of compactly supported wavelets, it is possible to rewrite the functional J in such a way that the same regularization is achieved with a different bilinear form. First, a brief survey of various results regarding these wavelets. For a detailed description of the discrete wavelet transform used in the experiments see the Appendix.

Periodic wavelets. Let ψ be a compactly supported wavelet as constructed by Daubechies in [3] with support in $[-R, R]$. Let ϕ be the corresponding function that generates the multiresolution analysis. Observe that ϕ has compact support on $[0, 2R]$. Define $\psi^P_{nk}(x) = 2^{n/2}\sum_{l=-\infty}^{\infty}\psi(2^n(x-l)-k)$, and $\phi^P_{nk}(x) = 2^{n/2}\sum_{l=-\infty}^{\infty}\phi(2^n(x-l)-k)$ for $0 \le x \le 1$. All the functions with superscript P are periodic with period 1, and if $k = 2^n$, then $\psi^P_{nk} = \psi^P_{n0}$, similarly for ϕ^P. The number of terms in these sums is determined by the support of ψ and is bounded by $2R$. For $n > r = \log_2 R + 1$ there are at most two terms in the sum.

For small n the functions ψ^P_{nk} cannot be expressed as scales and shifts of $\psi^P_{00}(x) \equiv \psi^P$. However, for $n > r$, we have $\psi^P_{(n+1)k}(x) = \sqrt{2}\psi^P_{n0}(2x-k)$, where the argument is considered modulo 1 or on the unit circle. Moreover, the family of functions ψ^P_{nk} for $n = 0, 1, \ldots$ and

$k = 0, 1, \ldots, 2^n - 1$ together with the function $\phi_{00}^P \equiv \mathbf{1}$ form an orthonormal basis. Let $V_0 = \text{span}\{\phi_{00}^P\}$ (i.e., the constant functions) and $W_n = \text{span}\{\psi_{nk}^P\}_{k=0}^{2^n-1}$. Using the fact that the ordinary wavelet basis spans $L_2(R)$, it is not hard to verify that

$$L_2(T^1) = V_0 \oplus \oplus_{n=1}^{\infty} W_n,$$

where T^1 is the one-dimensional torus. All that is necessary is to extend a periodic function f to a function $F \in L^2(R)$ by making $2R$ copies to the right and to the left of the unit interval and leaving the rest zero. Expanding F in the ordinary wavelet basis, we find that the part supported on [0, 1] can be given in terms of the functions $\mathbf{1}$ and ψ_{nk}^P; see [9, Chap. III, §11].

By [3] for sufficiently large R, the functions ψ and ϕ are twice differentiable and the first two moments of ψ are zero. Hence by [9, Chap. II, Thm. 8], $f \in H_2(T)$ if and only if

$$C(f, f) = f_{00}^2 + \sum_{n=1}^{\infty} \sum_{k=0}^{2^n-1} f_{nk}^2 (1 + 4^{2n}) < \infty,$$

where $f_{00} = \int f(x)dx$ and $f_{nk} = \int f(x)\psi_{nk}^P(x)dx$. This implies, via the closed graph theorem, that the Hilbert norm $C(f, f)$ defined above is equivalent to the Sobolev norm.

In two dimensions let

$$\Phi_{nkl}^P(x) = \phi_{nk}^P(x_1)\phi_{nl}^P(x_2), \qquad \Psi_{nkl}^{P,a}(x) = \phi_{nk}^P(x_1)\psi_{nl}^P(x_2),$$
$$\Psi_{nkl}^{P,b}(x) = \psi_{nk}^P(x_1)\phi_{nl}^P(x_2), \qquad \Psi_{nkl}^{P,c}(x) = \psi_{nk}^P(x_1)\psi_{nl}^P(x_2).$$

Let V_0 denote the constant functions and

$$W_n^{(a)} = \text{span}\left\{\Psi_{nkl}^{P,a}\right\}_{k,l=0}^{2^n-1}, \qquad n = 1, \ldots, \infty,$$

similarly for b and c. Setting $W_n = W_n^{P,a} \oplus W_n^{P,b} \oplus W_n^{P,c}$, we can show that

$$L_2(T^2) = V_0 \oplus \oplus_{n=1}^{\infty} W_n.$$

As in one dimension, $f \in H^2(T^2)$ if and only if

$$C(f, f) = f_{00}^2 + \sum_{n=1}^{\infty} \sum_{k,l=0}^{2^n-1} (1 + 4^{2n}) \left((f_{nkl}^{(a)})^2 + (f_{nkl}^{(b)})^2 + (f_{nkl}^{(c)})^2 \right) < \infty,$$

where $f_{nkl}^{(a)} = \int f(x)\psi_{nkl}^{P,a}(x)dx$, similarly for b and c. Consequently, it is possible to redefine $\varepsilon(U, U)$ in the functional J as $C(U^{(1)}, U^{(1)}) + C(U^{(2)}, U^{(2)})$ to obtain the same regularization as before. Now the finite dimensional spaces are given by

$$H_N = V_0 \oplus \oplus_{n=0}^{N-1} W_n.$$

Thus the regularization on the infinite dimensional space is the same as in the Fourier method; however, the sequence of approximations is different, and hence should lead to different types of solutions.

Observe that H_N is of dimension $2^N \times 2^N$. Moreover, for $f \in H_N$, the coefficients can be arranged in a $2^N \times 2^N$ array C so that $C_{00} = f_0$ and

$$C_{kl} = \begin{cases} f_{nkl}^{(a)} & \text{for } 2^n \le k < 2^{n+1} \quad \text{and } 0 \le l < 2^n, \\ f_{nkl}^{(b)} & \text{for } 0 \le k < 2^n \quad \text{and } 2^n \le l < 2^{n+1}, \\ f_{nkl}^{(c)} & \text{for } 2^n \le k < 2^{n+1} \quad \text{and } 2^n \le l < 2^{n+1}, \end{cases}$$

for $n = 1, \ldots, N - 1$.

Practical considerations. The algorithm itself proceeds very much like the one described in the previous section except that the Fourier transform is replaced by the two-dimensional wavelet transform. In practice, we are working with discrete data so that we use the discrete two-dimensional wavelet transform developed by Mallat in [8].

This transform takes the data D given on a discrete $2^N \times 2^N$ lattice as representing the coefficients of some function $f \in H_N$ on the unit square with respect to the functions Φ^P_{Nkl}, $k, l = 0, \ldots, 2^N - 1$. Think of f as an interpolation between the data values D_{ij} given on the discrete lattice using the compactly supported functions Φ^P_{nkl}. As above, the function f can be decomposed into its coefficients with respect to the wavelet basis. Let C denote this new $2^N \times 2^N$ array of coefficients, and write $C = W(D)$. Mallat [8] provides a very simple and fast algorithm for calculating $W(D)$, the forward wavelet transform, and $W^{-1}(C)$, the inverse wavelet transform. For the sake of completeness this is described in detail in the Appendix.

Now the algorithm described previously is modified as follows. Let $u^{(i)}$ describe the wavelet coefficients of $U^{(i)}$, i.e., $u^{(i)} = W(U^{(i)})$. In step (iii) we now have

$$u^{(i)}_{kl}(0) = \begin{cases} (\hat{u}_N)_{kl} & \text{for } 0 \le k, l < 2^N - 1, \\ 0 & \text{for } 2^N \le k < 2^{N+1} \quad \text{or} \quad 2^N \le l < 2^{N+1}. \end{cases}$$

In other words, when increasing the level of approximation $3 \cdot 2^N$ zeros are added for each component of the displacement field. In step (b) we have $U^{(i)}(t) = W^{-1}(u^{(i)}(t))$ and in step (c) $V^{(i)}(t) = W(Z^{(i)}(t))$ with $Z^{(i)}(t)$ as in (5). In step (d) the eigenvalues λ_{kl} are replaced by $\eta_{kl} = 1 + 4^{2n}$ for $2^n \le k < 2^{n+1}$, $0 \le l < 2^n$, or $0 \le k < 2^n$, $2^n \le l < 2^{n+1}$, or $2^n \le k < 2^{n+1}$, $2^n \le l < 2^{n+1}$.

In this approach the smoothing term is, of course, decoupled as before. In addition, since at higher frequencies ψ_{nk} have small support, the data parts of the equations are only *locally coupled.* This is in contrast to the Fourier approach in which *all* the equations are coupled through the data part.

This points to a clear advantage of the wavelet approach. It allows for local updates in problematic regions where the difference between $F(x + U(x))$ and $G(x)$ is still large, which has a relatively small effect on other regions where the difference is small.

The periodic setting we have chosen greatly facilitates the discrete algorithm by eliminating the need to store extra boundary terms at each level. Thus the size of the transformed data at each level of resolution is precisely that of the original data.

In practice we have used $R = 3$, which does not generate a wavelet smooth enough to apply the above-mentioned theorems; however, it performed well enough for our purposes.

The wavelet algorithm was tried using only the discrete transform. Due to its speed and ease of implementation there was no need to try the quadrature method for low-frequency coefficients. Moreover, instead of smoothing the data G and the template F as in the Fourier method, we take a lower resolution version of both, which consists of smoothing and sampling at the coarser grid (see the Appendix). The field U is found on the coarser grid. This field is then interpolated to the fine grid through the inversion formula (6) in the Appendix with the y coefficients set to zero (see Fig. 3). Again, this is similar to multigrid techniques; however, we did not apply it to a cascade of grids or continue the algorithm at the fine grid. This was merely a way to help the algorithm to "see" similar topological structures.

4. Description of experiments. In the first experiment (Fig. 1), the transformation between one x-ray of a hand, the template, to another, the data, is found and is used to automatically identify the location of important landmarks of the hand in the data image. The white lines connect points in the template to the points they are mapped to by the displacement field.

In this case, the tip of the index finger was successfully located in the data, whereas the tip of the little finger was slightly misplaced. The restoration has definitely succeeded in correcting the widths of the fingers and the palm. Observe that the fingers in the template are wider than those in the data, as is the palm of the hand. The absolute value of the initial difference between the images is shown as compared to the absolute value of the difference between the restoration image and the data. The black area indicates large differences.

In the second experiment (Fig. 5), the restoration on the right represents the result using a fixed dimension, whereas the restoration on the left used a sequence of increasing dimensions. The maximum dimension used was $N = 7$. Using the increasing dimension method not only produces a better result, but is also much faster when the Fourier transform is calculated via quadrature only for the frequencies being updated.

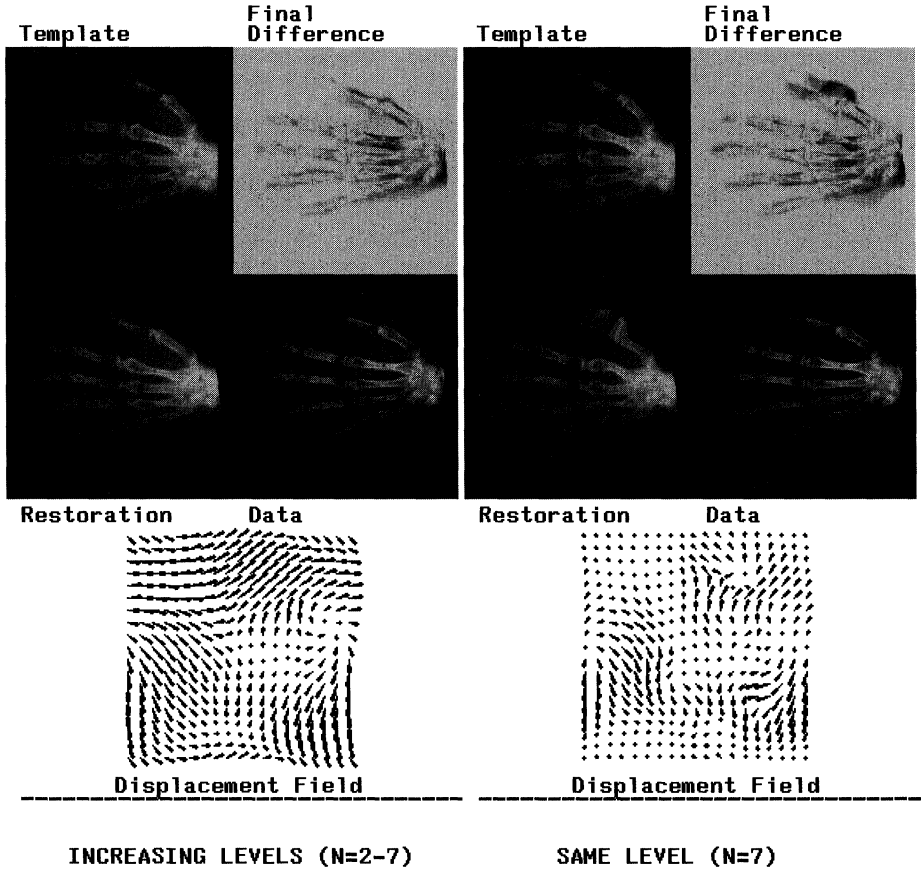


FIG. 5. *The restoration was done using the Fourier method. Using* (a) *increasing levels of approximation starting at* $N = 2$ *up to* $N = 7$, *and* (b) *using a fixed level* $N = 7$.

In the third experiment, we use a synthetic image consisting of two "humps" for the template F. The data G was created by composing F with some field U. The field U was generated by drawing the coefficients of the Fourier basis from independent Gaussian random variables with variances $1/\lambda_{nm}$, for $-9 \leq n,\ m \leq 10$. On the left in Fig. 2, we have the restoration process using the Fourier basis and on the right using the wavelet basis. Observe

that the final field obtained by both methods is very similar. Again, it is possible to see that the topological landmarks — the maxima, the saddle points, etc. — are mapped into each other by the field U. At the bottom we have the restoration using the linearized equations. These did not do nearly as well in spite of the fact that the mean square error in the equation was down to 10^{-5}.

The fourth experiment (Fig. 6) shows what happens when we attempt to use a template with the wrong topological features (Template 2) as compared to the correct template (Template 1). The data in this case was the same as in the previous two pictures with added independently and identically distributed Gaussian noise of variance 0.2, when the grey level values are scaled to the interval [0, 1]. Even though the restoration using the wrong template is fairly good, the displacement field is highly irregular when compared to the displacement field obtained by using the correct template. This might indicate a method of determining which of several possible templates corresponds to the objects in the noisy image.

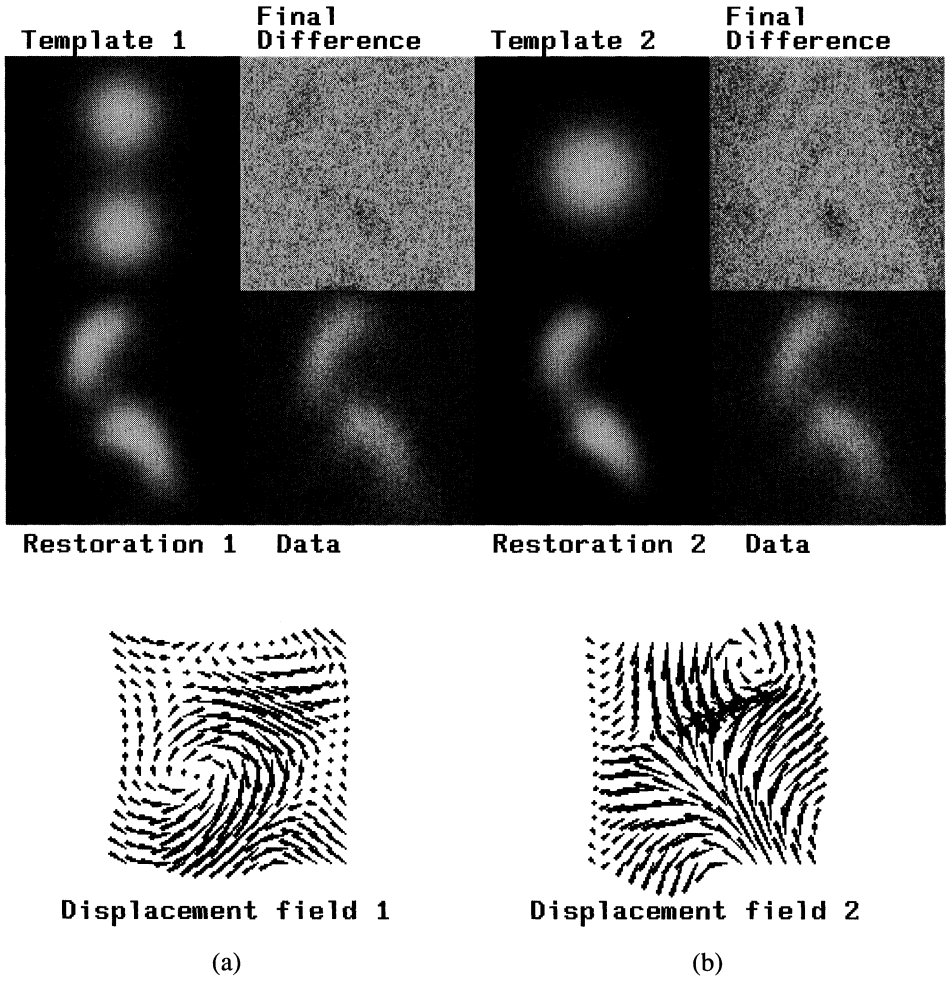


FIG. 6. *Gaussian independently and identically distributed noise was added to the data. In* (a) *the correct template was used, and in* (b) *a different template was used (one hump instead of two).*

The fifth experiment illustrates the possible advantages of the wavelet basis. The template F is a subimage from a sequence of x-rays of a catheter that has been inserted into a coronary

artery. The data G is the corresponding subimage of the x-ray that follows in the sequence. The idea is to use the algorithm to find the movement of the catheter between the two images. In this experiment the template F is larger than what is actually seen in the image. In other words, the function F is defined outside the unit square. When $x + U(x)$ lies outside the unit square, the value of F is taken from that point and not from the corresponding point on the torus. Thus we did not wrap around when composing the template with the mapping. This allows the algorithm to find possible movement *in and out* of the frame under consideration, and is, of course, much better adapted to the issue of movement compensation between consecutive movie frames.

Figure 3 shows the result using the wavelet basis. On the left side are the low resolution versions of the template, Frame 1, and of the data, Frame 2, together with the restoration obtained by using these lower resolution versions in the algorithm. On the right side are the original Frame 1 and Frame 2 together with the restoration obtained by using the displacement field from the low-resolution calculation. Most of the movement between these two consecutive frames has occurred in the upper right-hand corner and, indeed, the restoration has found that movement. Observe that in the low-resolution versions the black lines are widened and thus begin to overlap. This is what generates a gradient for the algorithm to proceed along.

Figure 4 shows the result of using the Fourier basis. The smooth version is on the right and the high resolution is on the left. Observe that this time the change in the upper right-hand corner was not found by the restoration procedure. Apparently, the wavelet basis performed better due to its ability to create a local change in the field without affecting other parts of the image.

In this experiment both the data *and* the template images are quite noisy. In the Fourier method the Fourier transforms of the template and the data are low-pass filtered and retransformed to the original space domain. They are, therefore, much smoother than in the wavelet case where a low-resolution version of the two is used, i.e., a low-pass filter that is not retransformed to the original resolution. This explains why the final low-resolution difference image in Fig. 3 is much noisier than the corresponding smooth version difference image in Fig. 4.

It should be noted that this movement analysis of the catheter is done without any preprocessing of the images to find the line or edge corresponding to the catheter. The algorithm was "helped" only by having it run on the low-resolution transform of the images. In such a way the "canyons" corresponding to the lines become wide valleys that overlap and enable the algorithm to draw them into one another. In other words, if the high resolution images are used, small changes of the field do not produce a better mean square error because the error along both canyons remains the same. Thus there is no clear gradient in any direction. On the other hand when the "valleys" overlap, the direction in which the field could reduce the error is clear.

In the sixth experiment (Fig. 7), we tried to find the movement between two subsequent frames of x-rays of the coronary arteries themselves. The direction of movement varies between different parts of the frame and the wavelet method managed to accommodate these local changes. In this experiment the actual difference between the two frames is compared to the actual difference between the restoration and Frame 2. The grey levels indicate values close to zero. Black or white values indicate large differences.

5. Conclusion. We have presented a nonlinear functional whose minimizers represent the mapping that transforms one image or function into another. The minimizers make sense only in so far as the two functions considered as surfaces have similar topographies, so that one function may be considered as a template for the other.

The functional is regularized using two different choices of a bilinear form. The first form is generated by a differential operator and has a spectral representation using the Fourier basis.

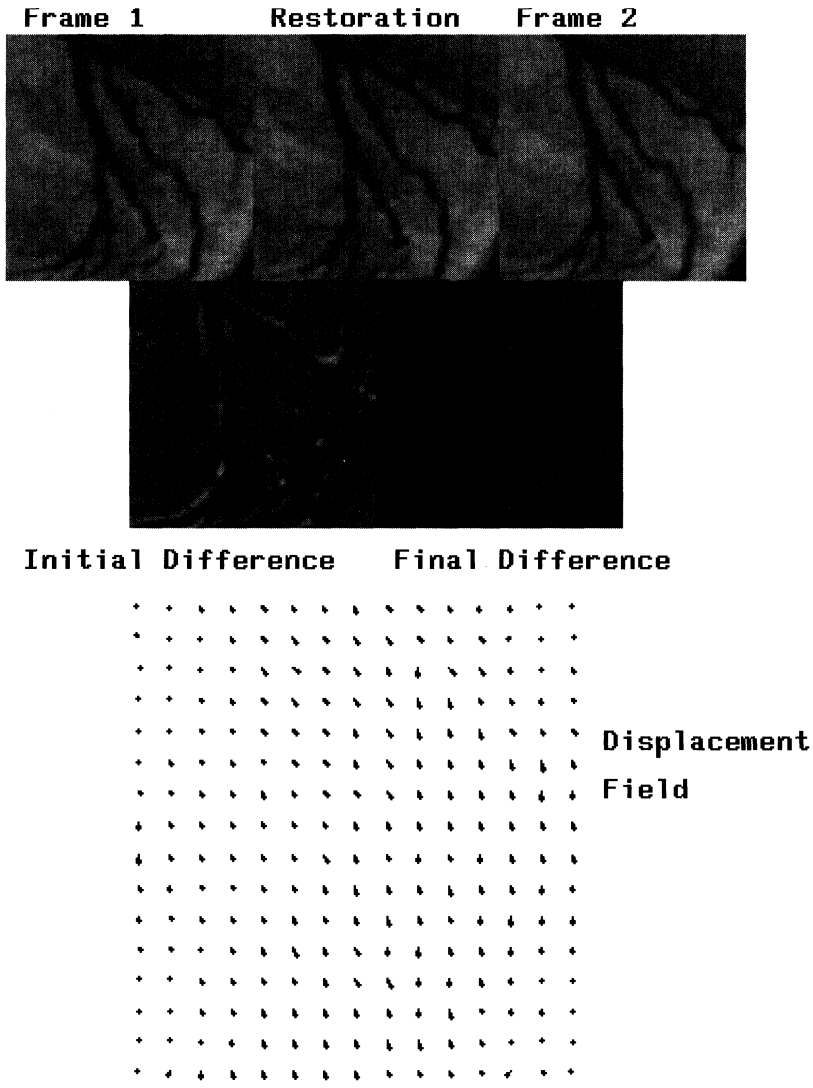


FIG. 7. *Frames* 1 *and* 2 *are two subsequent pictures from an x-ray movie of the coronary arteries. The displacement field corresponds to the movement between the two frames. The initial difference and the final difference pictures represent the actual difference. Grey values are close to zero and white and black values correspond to large differences. The restoration was done using the wavelet method.*

The second form is directly defined in the spectral domain using a wavelet basis of compact support. Both bilinear forms generate the same type of regularization, namely, they constrain the solutions to the same Sobolev space.

The variational problem is solved in terms of the expansion coefficients of the unknown map in terms of the chosen basis. The solution starts at low frequency and gradually moves up. The experimental results clearly indicate that the wavelet basis is more flexible and permits local changes in the mapping.

Appendix. For the sake of completeness we briefly describe the discrete wavelet transform so that the interested reader can code it without further reading. We start with the one-dimensional case.

Let $N = 2^n$ and let $x^{(0)} = (x_0, \ldots, x_{N-1})$ be the data. Given two vectors $g = (g_0, \ldots, g_{2R-1})$, and $h = (h_0, \ldots, h_{2R-1})$, let $\tilde{x} = h * x^{(0)}$ and $\tilde{y} = g * x^{(0)}$, i.e.,

$$\tilde{x}_k = \sum_{j=0}^{2R-1} h_j x_{[(k+j) \bmod N]}, \qquad \tilde{y}_k = \sum_{j=0}^{2R-1} g_j x_{[(k+j) \bmod N]},$$

for $k = 0, \ldots, N-1$. Now set $x_i^{(1)} = \tilde{x}_{2i}$ and $y_i^{(1)} = \tilde{y}_{2i}$ for $i = 0, \ldots, N/2-1$. Alternatively, we write $x^{(1)} = H_n x^{(0)}$ and $y^{(1)} = G_n x^{(0)}$, where H_n, G_n are linear mappings from R_N to $R^{N/2}$. As long as $N > 2R$, it is possible to write the $N/2 \times N$ matrix for G_n as follows:

$$G_n = \begin{pmatrix} g_0 & g_1 & g_2 & g_3 & \cdots & g_{2R-3} & g_{2R-2} & g_{2R-1} & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & g_0 & g_1 & \ldots & \ldots & \ldots & g_{2R-3} & g_{2R-2} & g_{2R-1} & \ldots & 0 & 0 \\ & & & \vdots & & & & & & & & & \\ g_2 & g_3 & \ldots & \ldots & \ldots & g_{2R-1} & 0 & 0 & 0 & 0 & \ldots & g_0 & g_1 \end{pmatrix}.$$

The $N/2 \times N$ matrix for H_n is given in the same way using the vector h. (Observe that we have indexed the matrices according to the log of the dimension.) This is the first level of the transform. Using the properties of multiresolution analysis and other techniques (see [3] and [8]) it is possible to find vectors g and h with the following properties.

(i) $\sum_{i=0}^{2R-1} g_i = 0$ and $\sum_{i=0}^{2R-1} h_i = \sqrt{2}$,
(ii) $g_i = (-1)^n h_{2R-1-i}$,
(iii) $H_n^t H_n + G_n^t G_n = I$,
(iv) $H_n G_n^t = 0$.

The first item implies that the H_n matrix is a smoothing operator and that the G_n matrix is a difference operator. The third item is the inversion formula. The fourth item is an orthogonality condition. Given two vectors x, y of length $N/2$, then the corresponding vectors in the full resolution space are orthogonal, i.e., $\langle H_n^t x, G_n^t y \rangle = 0$.

The second level of the transform is given by $x_i^{(2)} = h * x_{2i}^{(1)}$ and $y_i^{(2)} = g * x_{2i}^{(1)}$, for $i = 0, \ldots, N/4 - 1$. $y^{(1)}$ remains unchanged. Thus, at each level, the smoothed version $x^{(i)}$, which is of length 2^{n-i}, is decomposed into yet a smoother version $x^{(i+1)}$ and the difference component $y^{(i+1)}$ both of length 2^{n-i-1}. Clearly, this procedure comes to an end at step $i = n$, where $x^{(n)}$ and $y^{(n)}$ are scalars. The sequence of vectors $x^{(1)}, \ldots, x^{(n)}$ are called *lower resolution versions* of the original vector $x^{(0)}$.

From the inversion formula it follows that

$$x^{(i)} = H_{n-i}^t H_{n-i-1}^t \cdots H_1^t x^{(n)} + \sum_{l=i}^{n-1} H_{n-i}^t \cdots H_{n-l+1}^t G_{n-l}^t y^{(l)}.$$

In Daubechies [3], numbers for the vectors h and g are calculated so as to satisfy the four conditions listed above (see [3, p. 980]). Moreover, Daubechies shows that the wavelet corresponding to these numbers satisfies certain smoothness conditions. In terms of the inversion formula above this means that if we set $x^{(n)} = 1$ and $y^{(j)} = 0$ for $j = 1, \ldots, n$, then the vectors

$$x^{(i)} = H_{n-i}^t H_{n-i-1}^t \cdots H_1^t \mathbf{1}$$

for small values of i are smooth. It should be noted that since we are working here in the periodic setting, $x^{(0)}$ would be constant.

In two dimensions the wavelet transform is constructed using the one-dimensional operators described above. An initial $N \times N$ matrix $x^{(0)}$ is decomposed into four components. First, the one-dimensional transform is carried out on the columns, $A = H_n x^{(0)}$ and $B = G_n x^{(0)}$, where A, B are $N/2 \times N$ matrices. Then the one-dimensional transform is carried out on the rows of A and B or on the columns of A^t and B^t, and we obtain

$$x^{(1)} = H_n A^t = H_n (H_n x^{(0)})^t, \qquad y_a^{(1)} = G_n A^t = G_n (H_n x^{(0)})^t,$$
$$y_b^{(1)} = H_n B^t = H_n (G_n x^{(0)})^t, \qquad y_c^{(1)} = G_n B^t = G_n (G_n x^{(0)})^t.$$

The low-resolution version of $x^{(0)}$ is $x^{(1)}$. The transform continues by operating on $x^{(1)}$ with the matrices G_{n-1}, H_{n-1}. Finally, the inversion formula has the form

$$x^{(0)} = G_n^t (G_n^t y_c^{(1)} + H_n^t y_b^{(1)})^t + H_n^t (G_n^t y_a^{(1)} + H_n^t x^{(1)})^t. \tag{6}$$

Acknowledgments. I would like to express my gratitude to David Gottlieb for his suggestions and encouragement.

REFERENCES

[1] Y. Amit, U. Grenander, and M. Piccioni, *Structural image restoration through deformable templates*, J. Amer. Statist. Assoc., 86 (1991), pp. 376–387.

[2] R. Bajcy and S. Kovacic, *Multiresolution Elastic Matching*, Computer Vision, Graphics, and Image Processing, 46 (1989), pp. 1–21.

[3] I. Daubechies, *Orthonormal bases of compactly supported wavelets*, Comm. Pure Appl. Math., XLI (1988), pp. 909–996.

[4] D. Gottleib, M. Y. Hussaini, and A. O. Orszag, *Theory and applications of spectral methods*, in Spectral Methods for Partial Differential Equations, R. G. Voigt, D. Gottlieb, and M. Y. Hussaini, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1984.

[5] B. K. P. Horn and B. G. Schnuck, *Determining optical flow*, Artificial Intelligence, 17 (1981), pp. 185–203.

[6] T. S. Huang and R. Y. Tsai, *Image sequence analysis: Motion estimation*, in Image Sequence Analysis, T. S. Huang, ed., Springer-Verlag, New York, 1981.

[7] Z. Jin and P. Mowforth, *A discrete approach to signal mapping*, Tech. Report TIRM-88-036, The Turing Institute, Glasgow, Scotland, 1988.

[8] S. Mallat, *A theory for multiresolution signal decomposition: the wavelet representation*, in IEEE Trans. on Pattern Analysis and Machine Intelligence, 11 (1989), pp. 674–693.

[9] Y. Meyer, *Ondelettes et Operateurs*, Hermann, Paris, 1990.

[10] H. H. Nagel, *Displacement vectors derived from second-order intensity variations in image sequences*, Computer Vision, Graphics, and Image Processing, 21 (1983), pp. 85–117.

[11] D. Terzopoulos, *Image analysis using multigrid relaxation methods*, in IEEE Trans. on Pattern Analysis and Machine Intelligence, 8 (1986), pp. 129–139.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, pp. 225–230, January 1994

015

A REFORMULATION OF THE PARTIAL LEAST SQUARES REGRESSION ALGORITHM*

P. J. YOUNG†

Abstract. A new algorithm for partial least squares regression is introduced that is shown to be equivalent to, and more efficient than, the current algorithms. In addition, the new algorithm works explicitly in terms of the factor loadings and is therefore particularly useful when interpretation of the factors is required.

Key words. partial least squares, latent variables, factor loadings

AMS subject classifications. primary 62J99; secondary 62H20, 62H25, 62J05

1. Introduction. In the notation of Helland [2], we assume that we have k variables recorded on n subjects, $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$, where $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ are n-dimensional vectors, often referred to as the independent variables. Also available is an associated n-dimensional vector, $\mathbf{y}$, which we shall call the dependent variable. Until mentioned again it is assumed that the means $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_k$ and $\bar{y}$ have been subtracted from the data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$, and $\mathbf{y}$.

In linear regression we attempt to predict the values of the dependent variable using a linear combination of the independent variables, i.e., $\hat{\mathbf{y}} = \mathbf{X}\beta$ for some k-dimensional vector β. If we then define $\mathbf{S} = \mathbf{X}'\mathbf{X}$ and $\mathbf{s} = \mathbf{X}'\mathbf{y}$, then the ordinary least squares estimate of β is $\mathbf{S}^{-1}\mathbf{s}$.

Unfortunately, problems arise with $\mathbf{S}$ being singular if there is collinearity between the independent variables. This is frequently the case when large numbers of independent variables are available on relatively few subjects. Typically, examples of this are found in the social sciences [4] and the use of infrared spectroscopy data [1]. Use of the Moore–Penrose generalised inverses does not often lead to a unique solution [8], and selecting the appropriate variables from a large pool of variables is not an easy task.

There are three methods of overcoming the problem of collinearity which are currently used [6]. The first, partial least squares (PLS) regression [9] is the focus of this paper and is considered at length later. The second method, ridge regression, overcomes singularity of $\mathbf{S}$ by replacing it by $(\mathbf{S} + k\mathbf{I})$, where k is a scalar known as the ridge parameter. Ridge regression is known to lead to biased parameter estimates. However, it is also known to have better mean squared error properties if the ridge parameter is chosen correctly. Unfortunately, to date, no satisfactory method of selection of the ridge parameter has been proposed and the selection of the ridge parameter is carried out using subjective methods. A more detailed explanation of ridge regression is given by Smith and Campbell [7].

The final method is principal component (PC) regression. In PC regression linear combinations of the original variables are formed and used instead of the original variables. The point to note here is that because the new factors, or latent variables, are only linear combinations of the original variables, PC regression is still a method of linear regression.

The advantage of PC regression lies in the way in which the factor loadings are selected. These are selected sequentially, by maximising the variance of each factor such that it is mutually orthogonal to all the previous factors. These conditions are equivalent to using the eigenvectors $\mathbf{S}$ as factor loadings, with the eigenvalues being the corresponding variances.

*Received by the editors July 27, 1992; accepted for publication (in revised form) March 12, 1993. This research was supported by a Science and Engineering Research Council research studentship held at the University of Kent, United Kingdom.

†Department of Applied Statistics, University of Reading, PO box 238, Earley Gate 3, Whiteknights road, Reading RG6 2AL, United Kingdom (SNSYOUPH@uk.ac.reading).

The advantage of using PC regression is that if there is collinearity in the data, then the last few PC will have zero variance and can be discarded.

The idea of using latent variables is not just restricted to PCs. Two points to note are that, first, the importance of any of the PCs in the subsequent linear regression depends on more than just their variance. Second, the PCs are formed without reference to **s**. In PLS the factors are formed by sequentially maximising the canonical correlation between the latent variables and **y**, whilst keeping the different latent variables uncorrelated. PLS regression therefore has two advantages over PC regression. First, the order of the latent variables is known, and second, it usually requires fewer latent variables in any stepwise inclusion [2]. One criticism of using latent variable methods is that they use all the original variables. For this reason it is often desirable to see the factor loading explicitly so that the factors can be interpreted [3]. Alternatively, having access to the factor loadings may be useful in the detection of outliers [5].

2. PLS regression. PLS revolves around the central decomposition of **X**.

$$\mathbf{X} = \mathbf{t}_1\mathbf{p}_1' + \mathbf{t}_2\mathbf{p}_2' + \cdots + \mathbf{t}_A\mathbf{p}_A' + \mathbf{E}_A. \tag{1}$$

The $\mathbf{t}_a$'s are n-dimensional vectors, which are referred to as the "score vectors," while the $\mathbf{p}_a$'s are k-dimensional vectors, known as the "loading vectors." The $n \times k$ matrix $\mathbf{E}_A$ is a residual matrix, which is intended to be small in some sense. The basis of the PLS algorithm is that the relationship between **X** and **y** is carried through the score vectors, so that

$$\mathbf{y} = \mathbf{t}_1 q_1 + \mathbf{t}_2 q_2 + \cdots + \mathbf{t}_A q_A + \mathbf{f}_A, \tag{2}$$

where the q_a's are scalars and $\mathbf{f}_A$ is a residual vector.

There are a multitude of ways in which **X** could be decomposed, as in (1). There are two restrictions that are of interest here. First, the $\mathbf{t}_a$'s are forced to be mutually orthogonal, and second, the $\mathbf{p}_a$'s are forced to be mutually orthogonal. Using both restrictions has a unique solution [2]. Interestingly, the algorithms derived from assuming either the loadings or the scores to be orthogonal have been shown to be equivalent by Helland [2]. Here we will only consider the so-called original PLS algorithm, derived from assuming the $\mathbf{t}_a$'s to be orthogonal, which is by far the simpler of the two algorithms to use.

2.1. The original PLS algorithm. Initially, we define $\mathbf{E}_0 = \mathbf{X}$ and $\mathbf{f}_0 = \mathbf{y}$. The following items are then iteratively defined in the following sequence, starting with the so-called weight vector

$$\mathbf{w}_a = \mathbf{E}_{a-1}'\mathbf{f}_{a-1}, \tag{3.1}$$

$$\mathbf{t}_a = \mathbf{E}_{a-1}\mathbf{w}_{a-1}, \tag{3.2}$$

$$\mathbf{p}_a = (\mathbf{E}_{a-1}'\mathbf{t}_a)/(\mathbf{t}_a'\mathbf{t}_a), \tag{3.3}$$

$$q_a = \frac{\mathbf{f}_{a-1}'\mathbf{t}_a}{\mathbf{t}_a'\mathbf{t}_a} = \frac{\mathbf{w}_a'\mathbf{w}_a}{\mathbf{t}_a'\mathbf{t}_a}, \tag{3.4}$$

$$\mathbf{E}_a = \mathbf{E}_{a-1} - \mathbf{t}_a\mathbf{p}_a', \tag{3.5}$$

$$\mathbf{f}_a = \mathbf{f}_{a-1} - \mathbf{t}_a q_a. \tag{3.6}$$

The above algorithm does not explicitly give the factor loadings and the following procedure is needed for the prediction of a new observation. Because cross-validation is usually used to determine A, the number of factors required [2], this procedure is often used even if there are

no additional observations to classify. For additional observations with an outcome y_0 and covariates $(x_1, x_2, \ldots, x_k)$, we define $\mathbf{e}_0 = (x_1, x_2, \ldots, x_k)'$ and the predicted value of y_0 is

$$\hat{y}_0 = \bar{y} + \sum_{a=1}^{A} t_{a,0} q_a, \tag{4}$$

where $t_{a,0} = \mathbf{e}_{a-1}\mathbf{w}_a$ and $\mathbf{e}_a = \mathbf{e}_{a-1} - \mathbf{t}_{a,0}\mathbf{p}_a$.

Alternatively, the factor loadings can be explicitly calculated. Let $\mathbf{b}_i$ be the vector of factor loadings for the ith factor, then the predicted values of the dependent variable, $\hat{\mathbf{y}}$, is given by

$$\hat{\mathbf{y}} = \bar{\mathbf{y}} + \beta_1 \mathbf{X}\mathbf{b}_1 + \beta_2 \mathbf{X}\mathbf{b}_2 + \cdots + \beta_A \mathbf{X}\mathbf{b}_A = \bar{\mathbf{y}} + \mathbf{X}\mathbf{c}_A,$$

where β_i's are calculated by ordinary least squares regression [8] and $\mathbf{c}_A = \beta_1 \mathbf{X} + \beta_2 \mathbf{X} + \cdots + \beta_A \mathbf{X}$ is the vector of regression coefficients using A factors. Then as the regression coefficients and factor loadings are interchangeable quantities

$$\mathbf{c}_a = \mathbf{W}_a(\mathbf{P}_a'\mathbf{W}_a)^{-1}\mathbf{q}_a, \tag{5}$$

where $\mathbf{q}_a = (q_1, q_2, \ldots, q_a)'$, $\mathbf{W}_a = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_a)$, and $\mathbf{P}_a = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_a)$. Fortunately, it has been shown that $\mathbf{P}_a'\mathbf{W}_a$ is bidiagonal, with all diagonal entries equal to one [5], which is of help in the computation stage (see §5).

3. Reformulation of the original PLS algorithm. We define $\mathbf{U}_0 = \mathbf{I}$

$$\mathbf{U}_{a+1} = \mathbf{U}_a \left[\mathbf{I} - \frac{(\mathbf{U}_a'\mathbf{s})(\mathbf{U}_a'\mathbf{s})'(\mathbf{U}_a'\mathbf{S}\mathbf{U}_a)}{(\mathbf{U}_a'\mathbf{s})(\mathbf{U}_a'\mathbf{S}\mathbf{U}_a)(\mathbf{U}_a'\mathbf{s})} \right] \tag{6}$$

for $i = 1, 2, \ldots, k$. The Helland algorithm given in (3)–(8) is equivalent to the following.

$$\mathbf{w}_a = \mathbf{U}_{a-1}'\mathbf{s}, \tag{7.1}$$

$$\mathbf{t}_a = \mathbf{X}\mathbf{U}_{a-1}\mathbf{U}_{a-1}'\mathbf{s}, \tag{7.2}$$

$$\mathbf{p}_a = \frac{(\mathbf{U}_{a-1}'\mathbf{S}\mathbf{U}_{a-1})(\mathbf{U}_{a-1}'\mathbf{s})}{(\mathbf{U}_{a-1}'\mathbf{s})'(\mathbf{U}_{a-1}'\mathbf{S}\mathbf{U}_{a-1})(\mathbf{U}_{a-1}'\mathbf{s})}, \tag{7.3}$$

$$\mathbf{q}_a = \frac{(\mathbf{U}_{a-1}'\mathbf{s})'(\mathbf{U}_{a-1}'\mathbf{s})}{(\mathbf{U}_{a-1}'\mathbf{s})'(\mathbf{U}_{a-1}'\mathbf{S}\mathbf{U}_{a-1})(\mathbf{U}_{a-1}'\mathbf{s})}, \tag{7.4}$$

$$\mathbf{E}_a = \mathbf{X}\mathbf{U}_a, \tag{7.5}$$

$$\mathbf{f}_a = \mathbf{y} - \mathbf{X}\sum_{i=0}^{a-1} \frac{(\mathbf{U}_i\mathbf{U}_i'\mathbf{s})(\mathbf{U}_i'\mathbf{s})'(\mathbf{U}_i'\mathbf{s})}{(\mathbf{U}_i'\mathbf{s})'(\mathbf{U}_i'\mathbf{S}\mathbf{U}_i)(\mathbf{U}_i'\mathbf{s})}. \tag{7.6}$$

The proof of equivalence given below is dependent on the following lemma and is proved in the Appendix.

LEMMA. $\mathbf{U}_a = \mathbf{U}_i\mathbf{U}_a \forall i \leq a$.

Proof of equivalence. Proof is by induction. First, the forms of (7.5) and (7.6) are assumed to be true. Then, using (3.1),

$$\mathbf{w}_a = \mathbf{U}_{a-1}'\mathbf{s} - \mathbf{U}_{a-1}'\mathbf{S}\sum_{i=0}^{a-2} \frac{(\mathbf{U}_i\mathbf{U}_i'\mathbf{s})(\mathbf{U}_i'\mathbf{s})'(\mathbf{U}_i'\mathbf{s})}{(\mathbf{U}_i'\mathbf{s})'(\mathbf{U}_i'\mathbf{S}\mathbf{U}_i)(\mathbf{U}_i'\mathbf{s})}.$$

Now, using the lemma,

$$\mathbf{w}_a = \mathbf{U}'_{a-1}\mathbf{s} - \mathbf{U}'_{a-1}\sum_{i=0}^{a-2}\mathbf{U}'_{i+1}\mathbf{S}\frac{(\mathbf{U}_i\mathbf{U}'_i\mathbf{s})(\mathbf{U}'_i\mathbf{s})'(\mathbf{U}'_i\mathbf{s})}{(\mathbf{U}'_i\mathbf{s})'(\mathbf{U}'_i\mathbf{S}\mathbf{U}_i)(\mathbf{U}'_i\mathbf{s})},$$

and from (6)

$$\mathbf{w}_a = \mathbf{U}'_{a-1}\mathbf{s} - \mathbf{U}'_{a-1}\sum_{i=0}^{a-2}\left[\mathbf{U}'_i\mathbf{S}\frac{(\mathbf{U}_i\mathbf{U}'_i\mathbf{S})(\mathbf{U}'_i\mathbf{S})(\mathbf{U}'_i\mathbf{S})}{(\mathbf{U}'_i\mathbf{s})(\mathbf{U}'_i\mathbf{S}\mathbf{U}_i)(\mathbf{U}_i\mathbf{s})} - \frac{(\mathbf{U}'_i\mathbf{S}\mathbf{U}_i)(\mathbf{U}8_i\mathbf{s})(\mathbf{U}'_i\mathbf{s})'\mathbf{U}'_i\mathbf{S}(\mathbf{U}_i\mathbf{U}'_i\mathbf{s})(\mathbf{U}'_i\mathbf{s})'(\mathbf{U}'_i\mathbf{s})}{(\mathbf{U}'_i\mathbf{s})'(\mathbf{U}'_i\mathbf{S}\mathbf{U}_i)(\mathbf{U}'_i\mathbf{s})^2}\right] = \mathbf{U}'_{a-1}\mathbf{s}.$$

Using (3.2)–(3.4), the derivation of (7.2)–(7.4) is then trivial. Using (3.5) and (6), the form of (7.5) is easily verified as

$$\mathbf{E}_a = \mathbf{X}\mathbf{U}_{a-1}\left[\mathbf{I} - \frac{(\mathbf{U}'_{a-1}\mathbf{s})(\mathbf{U}'_{a-1}\mathbf{s})'(\mathbf{U}'_{a-1}\mathbf{S}\mathbf{U}_{a-1})}{(\mathbf{U}'_{a-1}\mathbf{s})'(\mathbf{U}'_{a-1}\mathbf{S}\mathbf{U}_{a-1})(\mathbf{U}'_{a-1}\mathbf{s})}\right] = \mathbf{X}\mathbf{U}_a,$$

and, finally, the form of (7.6) is also easily verified.

In order to complete the induction process it is only necessary to show that (7.5) and (7.6) hold for a trivial example. Consider $a = 0$, we know that $\mathbf{E}_0 = \mathbf{X}$, $\mathbf{f}_0 = \mathbf{y}$, and $\mathbf{U}_0 = \mathbf{I}$, in which case (7.5) and (7.6) are clearly correct. Hence equivalence of the algorithms is shown.

4. Implementation of the new algorithm. Although theoretically more simple, the above form of the algorithm is not practical. The regression equation is defined as

$$\hat{\mathbf{y}} = \bar{\mathbf{y}} + q_1\mathbf{X}\mathbf{b}_1 + q_2\mathbf{X}\mathbf{b}_2 + \cdots + q_A\mathbf{X}\mathbf{b}_A. \tag{8}$$

In the case where $\mathbf{b}_a = \mathbf{U}_a\mathbf{U}'_a\mathbf{s}$, consider the following:

$$\mathbf{U}_{a+1} = \mathbf{U}_a\left[\mathbf{I} - \frac{\mathbf{U}'_a\mathbf{s}\mathbf{s}'\mathbf{U}_a\mathbf{U}'_a\mathbf{S}\mathbf{U}_a}{(\mathbf{U}_a\mathbf{U}'_a\mathbf{s})'\mathbf{S}(\mathbf{U}_a\mathbf{U}'_a\mathbf{s})}\right] = \left[\mathbf{I} - \frac{\mathbf{b}_a\mathbf{b}'_a\mathbf{S}}{\mathbf{b}'_a\mathbf{S}\mathbf{b}_a}\right]\mathbf{U}_a.$$

Therefore,

$$\mathbf{U}_{a+1} = \prod_{i=1}^{a}\left(\mathbf{I} - \frac{\mathbf{b}_i\mathbf{b}'_i\mathbf{S}}{\mathbf{b}'_i\mathbf{S}\mathbf{b}_i}\right),$$

but it is known that $\mathbf{b}'_i\mathbf{S}\mathbf{B}_j = 0$ if $i \neq j$. Hence

$$\mathbf{U}_{a+1} = \mathbf{I} - \sum_{i=1}^{a}\left(\frac{\mathbf{b}_i\mathbf{b}'_i\mathbf{S}}{\mathbf{b}'_i\mathbf{S}\mathbf{b}_i}\right) = \mathbf{U}_a - \frac{\mathbf{b}_a\mathbf{b}'_a\mathbf{S}}{\mathbf{b}'_a\mathbf{S}\mathbf{b}_a}.$$

Therefore,

$$\begin{aligned}\mathbf{b}_{a+1} &= \mathbf{U}_{a+1}\mathbf{U}'_{a+1}\mathbf{S} = \mathbf{U}_{a+1}\mathbf{U}'_a\mathbf{s} - \mathbf{U}_{a+1}\mathbf{S}\mathbf{b}_a q_{a+1}\\ &= \left[\mathbf{I} - \frac{\mathbf{b}_a\mathbf{b}'_a\mathbf{S}}{\mathbf{b}'_a\mathbf{S}\mathbf{b}_a}\right]\mathbf{U}_a\mathbf{U}'_a\mathbf{s} - \mathbf{U}_{a+1}\mathbf{S}\mathbf{b}_a q_{a+1} = -\mathbf{U}_{a+1}\mathbf{S}\mathbf{b}_a q_{a+1}\\ &= -\left[\mathbf{I} - \frac{\mathbf{b}_a\mathbf{b}'_a\mathbf{S}}{\mathbf{b}'_a\mathbf{S}\mathbf{b}_a}\right]\mathbf{U}_a\mathbf{S}\mathbf{b}_a q_{a+1} = -\prod_{i=1}^{a}\left[\mathbf{I} - \frac{\mathbf{b}_i\mathbf{b}'_i\mathbf{S}}{\mathbf{b}'_i\mathbf{S}\mathbf{b}_i}\right]\mathbf{S}\mathbf{b}_a q_{a+1}\end{aligned}$$

remembering that $\mathbf{b}_i'\mathbf{S}\mathbf{B}_j = 0$ for $i \neq j$.

$$\mathbf{b}_{a+1} = \left[\left(\sum_{i=1}^{a} \frac{\mathbf{b}_i\mathbf{b}_i'\mathbf{S}}{\mathbf{b}_i'\mathbf{S}\mathbf{b}_i}\right) - I\right]\mathbf{S}\mathbf{b}_a q_{a+1}. \tag{9}$$

The new algorithm is then defined as follows.

Initially, obtain $\mathbf{S}$, $\mathbf{s}$, and define $\mathbf{g}_0 = \mathbf{S}\mathbf{s}$, $\mathbf{h}_0 = \mathbf{s}'\mathbf{S}\mathbf{s}$, and $q_1 = \mathbf{s}'\mathbf{s}/\mathbf{s}'\mathbf{S}\mathbf{s}$, then iteratively find

$$\mathbf{b}_{a+1} = \left(\sum_{i=0}^{a} \frac{\mathbf{b}_i(\mathbf{g}_i'\mathbf{g}_a)}{h_i} - \mathbf{g}_a\right) q_{a+1}, \tag{10}$$

and store $\mathbf{g}_{a+1} = \mathbf{S}\mathbf{b}_{a+1}$, $\mathbf{h}_{a+1} = \mathbf{b}_a'\mathbf{S}\mathbf{b}_a$, $\mathbf{q}_{a+2} = \mathbf{b}_{a+1}'\mathbf{s}/h_{a+1}$, and, of course $\mathbf{b}_{a+1}$.

5. Relative efficiency of the algorithms. If we consider the cost of the of the original PLS algorithm in terms of flops then we see that (3.1)–(3.3) are all matrix–vector multiplications each using nk flops. In addition, (3.3) requires an additional $n + 1$ flops for the scalar product of, and division by, $t_a't_a$. For (3.4) there is another scalar product requiring n flops and one extra for the scaling by $t_a't_a$. The vector-by-vector multiplications in (3.5) of $t_a p_a'$ require nk flops. For (3.6) only n flops are required. This gives a total of $4nk + 3n + 2$ flops per iteration. Therefore, if A factors are used, a total of

$$A(4nk + 3n + 2) \tag{11}$$

flops are required.

However, it is necessary to explicitly obtain the factor loadings. This requires the calculation and inversion of $\mathbf{P}_a'\mathbf{W}_a$, which is an $a \times a$ matrix. As the matrix is bidiagonal with all diagonal entries equal to one, we need only calculate the $a - 1$ off-diagonal entries which uses $k(a - 1)$ flops. The inversion itself is trivial using the Gauss–Jordan method which requires $a - 1$ flops. Postmultiplication by $\mathbf{q}_a$ requires $\frac{1}{2}a(a + 1)$ flops as the inverse is triangular. Finally, the premultiplication by $\mathbf{W}_a$ needs a^2 flops, which gives a total of $\frac{3}{2}a^2 + a(k + \frac{3}{2}) - (k + 1)$ flops. So, as A factors are required, an additional

$$\frac{1}{2}(A^3 + A^2k + 3A^2 - Ak) \tag{12}$$

flops will be required.

The new algorithm requires the multiplication of $\mathbf{g}_i'\mathbf{g}_a$, which takes k flops. This is then divided by h_i, and is then premultiplied by $\mathbf{b}_i$ using k flops. The procedure is repeated a times so that $a(2k + 1)$ are used. This vector is then scaled by q_{a+1} using another k_{a+1} flops. All that remains is to find $\mathbf{g}_{i+1}$, which needs k^2 flops, h_{i+1}, which requires k flops, and q_{i+2}, which uses another $k + 1$ flops. Hence a total of $a(2k + 1) + k^2 + 3k$ flops are required. Assuming that A factors are required, then

$$A\left(k^2 + Ak + 4k + \frac{1}{2}A + \frac{3}{2}\right) \tag{13}$$

flops are required.

We consider only the dominant terms of (11) and (13) for a large problem. For (11) the dominant term is $4Ank$, while for (13) it is $Ak^2 + A^2k$. Hence the ratio of the relative efficiency of the old method to the new is $4n/(A + k)$. If we then consider the cost of calculating the factor loadings in the old algorithm, the dominant term in (12) is $\frac{1}{2}(A^3 + A^2k)$, which alters the ratio of relative efficiency to $4n/(A + k) + A/2k$.

Appendix. Proof of lemma. The proof is in two parts. In the first part it is shown by induction that $\mathbf{U}_a\mathbf{U}_a = \mathbf{U}_a$. Assuming this to be true for $a = i$, consider $a = i + 1$. First, denoting

$$\mathbf{V}_{i+1} = \frac{(\mathbf{U}_i'\mathbf{s})(\mathbf{U}_i'\mathbf{s})'(\mathbf{U}_i'\mathbf{S}\mathbf{U}_i)}{(\mathbf{U}_i'\mathbf{s})'(\mathbf{U}_i'\mathbf{S}\mathbf{U}_i)(\mathbf{U}_i'\mathbf{s})},$$

it can be seen that

$$\mathbf{U}_{i+1}\mathbf{U}_{i+1} = \mathbf{U}_i[\mathbf{I} - \mathbf{V}_{i+1}]\mathbf{U}_i[\mathbf{I} - \mathbf{V}_{i+1}] = \mathbf{U}_i\mathbf{U}_i - \mathbf{U}_i\mathbf{V}_{i+1}\mathbf{U}_i - \mathbf{U}_i\mathbf{U}_i\mathbf{V}_{i+1} + \mathbf{U}_i\mathbf{V}_{i+1}\mathbf{U}_i\mathbf{V}_{i+1},$$

but, as $\mathbf{U}_i\mathbf{U}_i = \mathbf{U}_i$, it follows that $\mathbf{V}_{i+1}\mathbf{U}_i = \mathbf{V}_{i+1}$, and it is then easy to see that $\mathbf{U}_{i+1}\mathbf{U}_{i+1} = \mathbf{U}_{i+1}$. Hence all that remains is to show that $\mathbf{U}_a\mathbf{U}_a = \mathbf{U}_a$ hold for a trivial example, for which we consider $a = 0$, where $\mathbf{U}_0 = \mathbf{I}$.

We now note that $\mathbf{U}_a = \mathbf{U}_{a-1}(\mathbf{I} - \mathbf{V}_a)$ so that

$$\begin{aligned}\mathbf{U}_a &= (\mathbf{I} - \mathbf{V}_1)(\mathbf{I} - \mathbf{V}_2)\cdots(\mathbf{I} - \mathbf{V}_i)(\mathbf{I} - \mathbf{V}_{i+1})\cdots(\mathbf{I} - \mathbf{V}_a) = \mathbf{U}_i(\mathbf{I} - \mathbf{V}_{i+1})\cdots(\mathbf{I} - \mathbf{V}_a)\\ &= \mathbf{U}_i\mathbf{U}_i(\mathbf{I} - \mathbf{V}_{i+1})\cdots(\mathbf{I} - \mathbf{V}_a) = \mathbf{U}_i\mathbf{U}_a.\end{aligned}$$

REFERENCES

[1] I. FRANK AND B. KOWALSKI, *Prediction of wine quality and geographical origin from chemical measurements by partial least squares*, Analytica Chimica Acta, 162 (1984), pp. 241–251.

[2] I. HELLAND, *On the structure of partial least squares*, Comm. Statist. Simulation Comput., 17 (1988), pp. 581–607.

[3] I. JOLLIFFE, *Principal Component Analysis*, Springer-Verlag, Vienna 1986.

[4] K. JORESKOG AND H. WOLD, *System Under Indirect Observation: Causality-Structure-Prediction, Contributions to Economic Analysis*, Parts I and II, North-Holland, Amsterdam, 1982.

[5] R. MANNE, *Analysis of two partial-least-squares algorithms for multivariate calibration*, Chemometrics and Intelligent Laboratory Systems, 2 (1987), pp. 187–197.

[6] T. NAES AND H. MARTENS, *Comparison of prediction methods for multicollinear data*, Comm. Statist. Simulation Comput., 14 (1985), pp. 545–576.

[7] G. SMITH AND F. CAMPBELL, *A critique of some ridge regression methods*, J. Amer. Statist. Assoc., 75 (1980), pp. 74–103.

[8] M. STONE AND R. BROOKS, *Continuum regression: Cross-validation sequentially constructed prediction embracing ordinary least squares, partial least squares and principal component regression*, J. Roy. Statist. Soc. Ser. B, 52 (1990), pp. 237–269.

[9] 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. 15, No. 1, pp. 231–237, January 1994

016

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.

A NOVEL TWO-GRID METHOD FOR SEMILINEAR ELLIPTIC EQUATIONS*

JINCHAO XU†

Abstract. A new finite element discretization technique based on two (coarse and fine) subspaces is presented for a semilinear elliptic boundary value problem. The solution of a nonlinear system on the fine space is reduced to the solution of two small (one linear and one nonlinear) systems on the coarse space and a linear system on the fine space. It is shown, both theoretically and numerically, that the coarse space can be extremely coarse and still achieve asymptotically optimal approximation. As a result, the numerical solution of such a nonlinear equation is not significantly more expensive than the solution of one single linearized equation.

Key words. elliptic boundary value problem, finite elements, two-grid

AMS subject classifications. 65M60, 65N15, 65N30

1. Introduction. In this paper, we shall present a new discretization technique for semilinear elliptic equations based on finite element spaces defined on two grids of different sizes. Methods along this direction have been studied in a recent paper [1] for linear (nonsymmetric or indefinite) and especially nonlinear elliptic partial equations. (See also references cited in [1] for other methods for nonlinear elliptic equations.) As for nonlinear equations, the idea in [1] is basically to use a coarse space to produce a rough approximation of the solution, and then use it as the initial guess for one Newton iteration on the fine grid. This procedure involves a nonlinear solve on the coarse space and a linear solve on the fine space. A remarkable fact about this simple approach is, as shown in [1], that the coarse mesh can be quite coarse and still maintain an optimal approximation. The purpose of this paper is to make a further refinement in the aforementioned process by solving one more linear equation on the *coarse* space. This additional correction step (which needs very little extra work) improves the accuracy of the algorithms in [1] up to one or two orders. The fact that a further coarse grid correction *after* the fine grid correction can actually improve the accuracy appears to be of great theoretical interest.

The rest of the paper is organized as follows. Section 2 is devoted to the standard finite element discretization for a model semilinear equation. Section 3 contains the new algorithm of the paper with error analysis. Section 4 gives a simple numerical example.

2. Preliminaries. Given a bounded domain $\Omega \subset \mathbb{R}^d$ $(d = 1, 2, 3)$, we assume that $\partial\Omega$ is either smooth or convex and piecewise smooth. Let $\mathcal{W}_p^m(\Omega)$ be the standard Sobolev space with a norm $\|\cdot\|_{m,p}$ given by $\|v\|_{m,p}^p = \sum_{|\alpha|\le m} \|D^\alpha v\|_{L^p(\Omega)}^p$. For $p = 2$, we denote $\mathcal{H}^m(\Omega) = \mathcal{W}_2^m(\Omega)$ and $\mathcal{H}_0^1(\Omega)$ to be the subspace of $\mathcal{H}^1(\Omega)$ consisting of functions with vanishing trace on $\partial\Omega$. $\|\cdot\|_m = \|\cdot\|_{m,2}$ and $\|\cdot\| = \|\cdot\|_{0,2}$. The following Sobolev inequalities are well known:

*Received by the editors September 14, 1992; accepted for publication (in revised form) June 8, 1993. This work was supported by the National Science Foundation.

†Department of Mathematics, Pennsylvania State University, University Park, Pennsylvania 16802 (xu@math.psu.edu).

(2.1) $\|u\|_{0,p} \lesssim \|u\|_1 \quad (d = 2 \quad \text{and} \quad 1 \le p < \infty) \quad \text{and} \quad \|u\|_{0,6} \lesssim \|u\|_1 \quad (d = 3),$

where the notation "$\lesssim$" is equivalent to "$\le C$" for some positive constant C.

We consider the following semilinear equation:

$$-\Delta u + f(x, u) = 0, \quad x \in \Omega, \quad u|_{\partial\Omega} = 0. \tag{2.2}$$

Here the function f is sufficiently smooth. For brevity, we shall drop the dependence of variable x in $f(x, u)$ in the following exposition.

We assume that the above equation has at least one solution $u \in \mathcal{H}_0^1(\Omega) \cap \mathcal{H}^2(\Omega)$ and the linearized operator $L_u \equiv -\Delta + f'(u)$ is nonsingular. As a result of this assumption, $L_u : \mathcal{H}^2(\Omega) \cap \mathcal{H}_0^1(\Omega) \mapsto \mathcal{L}_2(\Omega)$ is a bijection and satisfies

$$\|w\|_2 \le C_0 \|L_u w\| \quad \forall\, w \in \mathcal{H}^2(\Omega) \cap \mathcal{H}_0^1(\Omega)$$

for some constant C_0 depending on u.

Given $h \in (0, 1)$, we assume that $V_h \subset \mathcal{H}_0^1(\Omega)$ is a piecewise linear finite element space defined on a quasi-uniform triangulation (with a meshsize h) of Ω satisfying

$$\inf_{\chi \in V_h} \{\|v - \chi\|_{0,p} + h\|v - \chi\|_{1,p}\} \lesssim h\|v\|_{2,p} \quad \forall\, v \in \mathcal{W}_p^2(\Omega) \cap \mathcal{H}_0^1(\Omega),\ 1 \le p \le \infty.$$

The standard finite element discretization of (2.2) is to find $u_h \in V_h$ so that

$$(\nabla u_h, \nabla \chi) + (f(u_h), \chi) = 0 \quad \forall\, \chi \in V_h. \tag{2.3}$$

It can be proven that (cf. [1] and the references cited therein) if h is sufficiently small, the above equation has a (locally unique) solution u_h satisfying

$$\|u - u_h\|_{0,p} + h\|u - u_h\|_{1,p} \le C(\|u\|_{2,p})h^2 \quad \forall\, 2 \le p < \infty \tag{2.4}$$

and

$$\|u - u_h\|_{0,\infty} \le C(\|u\|_{2,\infty})h^2 |\log h|, \qquad \|u - u_h\|_{1,\infty} \le C(\|u\|_{2,\infty})h.$$

LEMMA 2.1. *There exists a constant $\delta > 0$ such that for any $v \in \mathcal{H}_0^1(\Omega) \cap \mathcal{L}_\infty(\Omega)$ with $\|u - v\|_{0,\infty} \le \delta$, the following statements are valid.*

1. *The operator $L_v \equiv -\Delta + f'(v) : \mathcal{H}_0^1(\Omega) \cap \mathcal{H}^2(\Omega) \mapsto \mathcal{L}_2(\Omega)$ is bijective and there exists a constant $C = C(\delta)$, so that*

$$\|w\|_2 \le C(\delta)\|L_v w\| \quad \forall\, w \in \mathcal{H}_0^1(\Omega) \cap \mathcal{H}^2(\Omega).$$

2. *If h is sufficiently small, there exists a constant $c = c(\delta)$ such that*

$$\sup_{\chi \in V_h} \frac{a_v(w_h, \chi)}{\|\chi\|_1} \ge c(\delta)\|w_h\|_1 \quad \forall\, w_h \in V_h,$$

where

$$a_v(w_h, \chi) = (\nabla w_h, \nabla \chi) + (f'(v)w_h, \chi). \tag{2.5}$$

Proof. Since f is smooth, we have

$$\|f'(u) - f'(v)\|_{0,\infty} \le C_1\|u - v\|_{0,\infty} \le C_1\delta$$

for some constant C_1 depending on f and u. Thus

$$\|(L_u - L_v)w\| = \|(f'(u) - f'(v))w\| \le \|f'(u) - f'(v)\|_{0,\infty}\|w\| \le C_1\delta\|w\|_2,$$

and, if $\delta \le (2C_0C_1)^{-1}$,

$$\|w\|_2 \le C_0\|L_u w\| \le C_0(\|L_v w\| + \|(L_u - L_v)w\|) \le C_0\|L_v w\| + \tfrac{1}{2}\|w\|_2.$$

The first statement is then justified with $C = 2C_0$.

The assumption that L_u is nonsingular implies that

$$\sup_{\chi \in V_h} \frac{a_u(w_h, \chi)}{\|\chi\|_1} \ge c_0\|w_h\|_1 \quad \forall\, w_h \in V_h$$

for some constant c_0 and sufficiently small h. But

$$a_v(w_h, \chi) \ge a_u(w_h, \chi) - C_1\delta\|w_h\|_1\|\chi\|_1.$$

The second statement then follows with $c = c_0/2$ for $\delta \le c_0/(2C_1)$. □

3. Two-grid algorithms based on the Newton method. In this section, we shall present the main algorithm of this paper. The basic ingredient in our approach is another finite element space V_H ($\subset V_h \subset \mathcal{H}_0^1(\Omega)$) defined on a coarser quasi-uniform triangulation (with meshsize $H > h$) of Ω. Note that all the results for V_h in the previous section are valid for V_H if H is sufficiently small.

Setting $a_H(v, \phi) = a_{u_H}(v, \phi)$ (see (2.5)), the main algorithm of the paper is as follows.

ALGORITHM A1. Find $u_h^* = u_H + e_h + e_H$ such that

1. $u_H \in V_H$, $(\nabla u_H, \nabla\phi) + (f(u_H), \phi) = 0 \quad \forall\, \phi \in V_H$;
2. $e_h \in V_h$, $a_H(e_h, \chi) = -(f(u_H), \chi) - (\nabla u_H, \nabla\chi) \quad \forall\, \chi \in V_h$;
3. $e_H \in V_H$, $a_H(e_H, \phi) = -\frac{1}{2}(f''(u_H)e_h^2, \phi) \quad \forall\, \phi \in V_H$.

The new feature of the above algorithm mainly lies in step 3 where a further coarse grid correction is performed. We notice that the linearized operator used in this step is based on the first coarse grid approximation u_H (instead of the more accurate $u_H + e_h$). As we shall see later, such a correction indeed improves the accuracy of the approximation.

Corresponding to the form $a_H(\cdot, \cdot)$, we define a projection $P_H : \mathcal{H}_0^1(\Omega) \mapsto V_H$ by

$$a_H(\phi, P_H v) = a_H(\phi, v) \quad \forall\, \phi \in V_H, v \in \mathcal{H}_0^1(\Omega).$$

By Lemma 2.1, it can be easily shown that there exists $H_0 > 0$, if $H \le H_0$, P_H is well defined and satisfies for $k = 1, 2$

$$\|w - P_H w\| + H\|w - P_H w\|_1 \le C(H_0)H^k\|w\|_k \quad \forall\, w \in \mathcal{H}_0^1(\Omega) \cap \mathcal{H}^k(\Omega). \tag{3.1}$$

We begin our analysis for Algorithm A1 with a simple lemma.

LEMMA 3.1. *For any* $\chi \in V_h$

$$a_H(u_H + e_h, \chi) = (-f(u_H) + f'(u_H)u_H, \chi), \tag{3.2}$$

$$\begin{aligned} a_H(u_h^*, \chi) = {} & (-f(u_H) + f'(u_H)u_H - \tfrac{1}{2}f''(u_H)e_h^2, \chi) \\ & + \tfrac{1}{2}(f''(u_H)e_h^2, \chi - P_H\chi). \end{aligned} \tag{3.3}$$

Proof. The first equality follows directly for the definition of u_H and e_h. Again, by the definition of P_H and e_H,

$$a_H(e_H, \chi) = a_H(e_H, P_H\chi) = -\tfrac{1}{2}(f''(u_H)e_h^2, P_H\chi).$$

Adding this to the first equality then leads to the second one. □

LEMMA 3.2. *If* $u \in \mathcal{W}_p^2(\Omega)$ *with* $p > 2$ *for* $d = 2$ *and* $p = \frac{12}{5}$ *for* $d = 3$, *then*

$$\|u_h - (u_H + e_h)\|_1 \lesssim H^4.$$

Proof. It follows from (2.3) that

$$a_H(u_h, \chi) = -(f(u_h), \chi) + (f'(u_H)u_h, \chi),$$

which, together with (3.2), gives

$$a_H(u_h - (u_H + e_h), \chi) = (f(u_H) - f(u_h) + f'(u_H)(u_h - u), \chi) = (b(u_h - u_H)^2, \chi).$$

Here we have

$$b = -\int_0^1 (1-t)f''(u_H + t(u_h - u_H))dt.$$

By assumption and (2.4), it is easy to see that b is a uniformly (with respect to both H and h) bounded function on $\bar{\Omega}$. Using the Hölder inequality and (2.1), we then deduce that

$$((u_h - u_H)^2, \chi) \le \|(u_h - u_H)^2\|_{0,\frac{p}{2}}\|\chi\|_{0,\frac{p}{p-2}} \lesssim \|u_h - u_H\|_{0,p}^2\|\chi\|_1.$$

It follows from Lemma 2.1 and (2.4) that

$$\|u_h - (u_H + e_h)\|_1 \lesssim \|u_h - u_H\|_{0,p}^2 \lesssim H^4. \qquad \square$$

The estimate in Lemma 3.2, essentially contained in [1], is already quite remarkable because of the high power on the coarse meshsize H. But more remarkable estimates will be seen in the next theorem.

THEOREM 3.1. *If* $u \in \mathcal{W}_4^2(\Omega)$ *is the solution of* (2.2) *and* $u_h \in V_h$ *is the solution of* (2.3) (*satisfying* (2.4)), *then*

$$\|u_h - u_h^*\|_1 \lesssim H^5, \qquad \|u_h - u_h^*\| \lesssim H^6.$$

Consequently,

$$\|u - u_h^*\|_1 \lesssim h + H^5, \quad \|u - u_h^*\| \lesssim h^2 + H^6.$$

Proof. By the definition of u_h and the Taylor expansion, we have

$$\begin{aligned} a_H(u_h, \chi) = (&-f(u_H) + f'(u_H)u_H - \tfrac{1}{2}f''(u_H)(u_h - u_H)^2, \chi) \\ &+(O(u_h - u_H)^3, \chi), \end{aligned}$$

which, together with (3.3), gives that for any $\chi \in V_h$,

$$\begin{aligned} a_H(u_h - u_h^*, \chi) = &-\tfrac{1}{2}(f''(u_H)(e_h^2 - (u_h - u_H)^2), \chi) + \tfrac{1}{2}(f''(u_H)e_h^2, \chi - P_H\chi) \\ &+(O(u_h - u_H)^3, \chi). \end{aligned}$$

By the Hölder inequality and (2.1)

$$\begin{aligned}(e_h^2 - (u_h - u_H)^2, \chi) &\le \|(u_h - (u_H + e_h))(e_h + u_h - u_H)\|_{0,\frac{5}{6}} \|\chi\|_{0,6} \\ &\le \|u_h - (u_H + e_h)\|_{0,3} \|e_h + u_h - u_H\|_{0,2} \|\chi\|_{0,6} \\ &\le \|u_h - (u_H + e_h)\|_1 (\|e_h\| + \|u_h - u_H\|) \|\chi\|_1.\end{aligned}$$

It follows from (2.4) and Lemma 3.2 that

$$(e_h^2 - (u_h - u_H)^2, \chi) \le H^6 \|\chi\|_1.$$

By the Schwarz inequality and Lemma 3.2,

$$(f''(u_H) e_h^2, (I - P_H)\chi) \lesssim \|e_h\|_{0,4}^2 \|(I - P_H)\chi\| \lesssim H^4 \|(I - P_H)\chi\|.$$

By the Hölder inequality, (2.4), and (2.1),

$$(O(u_h - u_H)^3, \chi) \lesssim \|(u_h - u_H)^3\|_{0,\frac{6}{5}} \|\chi\|_{0,6} \lesssim \|u_h - u_H\|_{0,4}^3 \|\chi\|_1 \lesssim H^6 \|\chi\|_1.$$

Consequently,

$$a_H(u_h - u_h^*, \chi) \lesssim H^6 \|\chi\|_1 + H^4 \|(I - P_H)\chi\|. \tag{3.4}$$

This, together with Lemma 2.1 and (3.1), immediately implies the first estimate of the theorem. To derive the estimate in $\mathcal{L}_2(\Omega)$ norm, we use a duality argument by considering the auxiliary problem: Find $w \in \mathcal{H}_0^1(\Omega) \cap \mathcal{H}^2(\Omega)$ so that

$$-\Delta w + f'(u_H) w = u_h - u_h^*.$$

By Lemma 2.1, there exists $H_0 > 0$ and $C(H_0) > 0$, so that if $H \le H_0$,

$$\|w\|_2 \le C(H_0) \|u_h - u_h^*\|.$$

Then, if $w_h \in V_h$ is the nodal value interpolation of w, we have

$$\|u_h - u_h^*\|^2 = a_H(u_h - u_h^*, w) = a_H(u_h - u_h^*, w - w_h) + a_H(u_h - u_h^*, w_h).$$

Note that

$$a_H(u_h - u_h^*, w - w_h) \lesssim \|u_h - u_h^*\|_1 \|w - w_h\|_1 \lesssim H^5 h \|w\|_2 \lesssim H^6 \|u_h - u_h^*\|.$$

It follows from (3.4) that

$$\begin{aligned}a_H(u_h - u_h^*, w_h) &\lesssim H^6 \|w_h\|_1 + H^4 \|(I - P_H) w_h\| \\ &\lesssim H^6 \|w\|_2 + H^4 (\|(I - P_H) w\| + H \|w - w_h\|_1) \\ &\lesssim H^6 \|w\|_2 \lesssim H^6 \|u_h - u_h^*\|.\end{aligned}$$

The second estimate of the theorem then follows. □

Comparing Theorem 3.1 with Lemma 3.2, we find that one coarse grid correction leads to a one-order improvement in $\mathcal{H}^1(\Omega)$ norm and possibly a two-order improvement in $\mathcal{L}_2(\Omega)$ norm.

According to Theorem 3.1, to obtain the asymptotically optimal accuracy, it suffices to take $H = O(h^{1/3})$ for both $\mathcal{L}_2(\Omega)$ and $\mathcal{H}^1(\Omega)$ norms (for $\mathcal{H}^1(\Omega)$ norm, it even suffices to take $H = O(h^{1/5})$). As a result, the dimension of V_H can be much smaller than the dimension

of V_h, and thus the dominated part of the work in Algorithm A1 is the solution of the linear system in step 2.

With a standard treatment near the boundary where it is curved, the estimates corresponding to Theorem 3.1 for Algorithm A1 for elements of degree r are as follows:

$$\|u_h - u_h^*\|_1 \lesssim H^{2r+3}, \qquad \|u_h - u_h^*\| \lesssim H^{2r+4}.$$

Thus

$$\|u - u_h^*\|_1 \lesssim h^r + H^{2r+3}, \qquad \|u - u_h^*\| \lesssim h^{r+1} + H^{2r+4}.$$

A proper choice of meshsizes would be $h = O(H^{2+2/(r+1)})$ (or even $h = O(H^{2+3/r})$).

For most practical purposes, Algorithm A1, which involves only one Newton iteration on the fine grid, is sufficient for applications. Nevertheless, a more dramatic result can be derived if one more Newton iteration is performed on the fine grid.

ALGORITHM A2. Find $\tilde{u}_h = u_h^* + e^h$ such that
1. $u_h^* \in V_h$ is obtained by Algorithm A1.
2. $e^h \in V_h$, $a_{u_h^*}(e^h, \chi) = -(f(u_h^*), \chi) - (\nabla u_h^*, \nabla\chi) \quad \forall\, \chi \in V_h$.

For this algorithm, we can prove that, if $u \in W_4^2(\Omega)$, then

$$\|u_h - \tilde{u}_h\|_1 \lesssim H^{12} |\log h|^{\frac{1}{2}}.$$

In fact, by Taylor expansion, we have

$$a_{u_h^*}((u_h - \tilde{u}_h), \chi) = (O(u_h - u_h^*)^2, \chi).$$

For $d = 2$, using the well-known inequality that $\|\chi\|_{0,\infty} \lesssim |\log h|^{1/2} \|\chi\|_1$, we deduce that

$$(O(u_h - u_h^*)^2, \chi) = \|u_h - u_h^*\|^2 \|\chi\|_{0,\infty} \lesssim \|u_h - u_h^*\|^2 \|\chi\|_1 |\log h|^{\frac{1}{2}} \lesssim H^{12} |\log h|^{\frac{1}{2}} \|\chi\|_1.$$

The desired estimate for $d = 2$ then follows. The proof for $d = 3$ depends on the $L_p(\Omega)$ estimates of $u_h - u_h^*$, and the technical details will be omitted here.

It is also possible to make an additional coarse grid correction in Algorithm A2 to further improve the estimate, but we suspect such an improvement will not be that important since the order of H is already so high.

One final remark is that the technique presented in this paper extends naturally to other discretization methods such as finite difference and spectral methods.

4. A numerical example. In this section we present a simple numerical result to demonstrate the efficiency of our proposed schemes. Our model problem is

$$-\Delta u + u^3 = f(x) \quad \text{in } \Omega \quad \text{and} \quad u = 0 \quad \text{on } \partial\Omega.$$

Here Ω is the unit square $(0, 1) \times (0, 1)$ and f is so chosen that $u = \sin \pi x \sin \pi y$ is the exact solution. The domain Ω is divided into families T_H and T_h of quadrilaterals, and $V_H, V_h \subset H_0^1(\Omega)$ are linear spaces of piecewise continuous bilinear functions defined on T_H and T_h, respectively.

On the coarse grid level, we solve the nonlinear problem by the Newton iteration. (Because of its small size, this nonlinear system takes very little time to solve compared to the larger linear systems.) The fine grid linearized equations (which are symmetric positive definite for this example) are solved by the conjugate gradient method or multigrid method. We take $h = H^4$ with $H = \frac{1}{4}$. Notice that $\dim V_h = 65,025$ while $\dim V_H = 49$. The numerical results are shown in Table 1.

TABLE 1
Errors in $\mathcal{H}^1$ and $\mathcal{L}_2$ norms.

	$\mathcal{H}^1$-norm	$\mathcal{L}_2$-norm
$u - u_H$	5.01E−1	3.19E−02
$u - (u_H + e_h)$	7.90E−3	1.45E−04
$u - (u_H + e_h + e_H)$	7.87E−3	3.19E−05
$u - u_h$	7.87E−3	7.85E−06

Acknowledgment. The author wishes to thank Dr. Ping Lee for his help on numerical experiments.

REFERENCE

[1] J. XU, *Two grid finite element discretizations for linear and nonlinear elliptic equations*, Tech. Report, AM105, Dept. of Mathematics, Pennsylvania State University, University Park, July, 1992.

SIAM J. SCI. COMPUT.
Vol. 15, No. 1, p. 238, January 1994

017

ERRATUM: HYPERCUBE ALGORITHMS FOR DIRECT N-BODY SOLVERS FOR DIFFERENT GRANULARITIES*

JEAN-PHILIPPE BRUNET†, ALAN EDELMAN‡, AND JILL P. MESIROV†

The authors were inadvertently listed out of alphabetical order in the previous publication of this article [1].

REFERENCES

[1] J.-P. BRUNET, J. P. MESIROV, AND A. EDELMAN, *Hypercube algorithms for direct N-body solvers for different granularities*, SIAM J. Sci. Comput., 14 (1993), pp. 1143–1158.

*Received by the editors September 15, 1993; accepted for publication September 16, 1993.

†Thinking Machines Corporation, 245 First Street, Cambridge, Massachusetts 02142 (brunet@think.com, mesirov@think.com).

‡Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (edelman@math.mit.edu).

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 239–262, March 1994

DATA ANALYSIS, MATRIX DECOMPOSITIONS, AND GENERALIZED INVERSE*

AGNAR HÖSKULDSSON†

Abstract. An approach to data analysis and matrix analysis is presented with the aid of a general algorithm to carry out data analysis, matrix decompositions, and computation of the generalized inverse. The algorithm represents a novel approach to common numerical and statistical analysis of data. It allows users to specify their views on data so that the solution may reflect the importance of different parts of data. The algorithm provides a unified method to analyze data, to decompose the matrix similar to the singular value decomposition, and to compute the generalized inverse. It is a fast and efficient way to handle numerical questions like solving linear equations or determining the rank of a matrix. The algorithm allows views on the column vectors, row vectors, and on both rows and columns. Different views on the data give different decompositions or solutions. Stopping rules are presented that can be used to identify the noise level in data. Some basic modelling questions are treated and applied to typical situations found in practice.

Key words. linear equations, rank of matrix, decomposition, generalized inverse

AMS subject classifications. 15A06–15A24, 65F05–65F30

1. Introduction.

The solution of linear equations and the decomposition of matrices are now fundamental techniques in many applied sciences. Excellent textbooks have been written on these topics, both from a numerical point of view (see, e.g., [1]) and from a statistical point of view (see, e.g., [2] and [3]). One can state that these theories have been extensively studied, and there are now good algorithms available for performing linear numerical analysis such as solving linear equations, computing eigenvalues and vectors of matrices, and handling similar matrix problems.

Modern measurement equipment often produces a large amount of data. In modern chemical laboratories it is common that the data matrices collected from the measurements are large, often containing more than 100 rows or columns. Matrices containing 100 or 200 rows and a similar number of columns are "medium"-sized matrices in many chemical laboratories. These matrices are usually of low rank. We often see matrices of size 100 times 100 or larger, but of rank only 10 or even less. For example, in these cases we can easily compute the singular value decomposition (SVD) of the matrix. But experience with these solutions has shown that they often do not function well. The reason is that the results are often very unstable. In statistics it is called "instability due to overfitting." In numerical analysis the phenomenon is well known in the case of polynomial approximation: If we choose too high a degree of the polynomial in order to get good fit to the data points, the estimated polynomial will tend to fluctuate between the data points. The approximation may be good at the data points, but can be very bad between the points. Experience with these large models has shown that what is important is that one gets a stable solution. The exact numerical solutions give good fit or description, but when applied they often give unstable and uncertain predictions. For example, a stable solution should, from a practical point of view, be almost equally good in terms of fit as the exact numerical solution, but have good performance when it is applied.

We present here an algorithm for solving linear equations and linear decomposition of matrices, which has been found efficient in handling large data matrices. The solutions arrived at by the algorithm have been found to be stable and function well from the point of view of the user, i.e., providing an almost equally good solution as the exact one, but giving more precise predictions.

*Received by the editors July 9, 1992; accepted for publication (in revised form) March 19, 1993.
†Danish Engineering Academy, DK-2800 Lyngby, Denmark (ah@m.dia.dk).

If a large data matrix $\mathbf{A}$ is given and it turns out to be of low rank, the linear equation $\mathbf{Ax} = \mathbf{b}$ is often solved by computing the generalized inverse $\mathbf{A}^\dagger$ by the SVD, and one works further with the solution $\mathbf{x} = \mathbf{A}^\dagger\mathbf{b}$. But this is usually not a good way to compute the solution vector. The reason is that SVD does not take into consideration the right-hand-side vector $\mathbf{b}$. If $\mathbf{b}$ projects onto the subspace associated with the small singular values of $\mathbf{A}$, the solution vector can be very bad or even useless, although there may be a solution given by the present algorithm that functions well. Similarly, the results of an SVD may not be easy to interpret for the user, because the criteria for SVD may not be the important ones for the user.

The present algorithm reduces at each step the matrix $\mathbf{A}$ by rank one. The result, if all components are extracted, is a decomposition of $\mathbf{A}$ of a kind that looks like SVD:

$$\mathbf{A} = \lambda_1\mathbf{u}_1\mathbf{v}_1{}' + \lambda_2\mathbf{u}_2\mathbf{v}_2{}' + \cdots.$$

But in the present algorithm the vectors $(\mathbf{u}_i)$ and $(\mathbf{v}_i)$ need not be orthogonal vectors. The properties of these vectors depend on the criteria used.

In many applied situations we have views or expectations concerning the results we obtain from the mathematical model. For example, if we wish to estimate the maximum of a quadratic surface that fits the given data, we have certain expectations concerning the results. We wish the maximum value to be well determined and the quadratic surface to fit well to data. Often the data follow approximately a quadratic surface in the neighbourhood of the maximum, while data points far away from maximum may not be adequately described by a second-order model. In such situations one may have to choose between a good description of data and a good determination of the maximum. In situations like that it can be useful to be able to specify that the algorithm should take into account the way we look at data. It is actually quite common that we have an attitude concerning the way in which data should be viewed. In neural networks, e.g., a_{ij} is sometimes the weight that goes from node i to node j in the network. If the values in a row are small, we want the solution vector not to depend much on such small values. Similarly, in geology substantial measurement values may be more important than samples with small values. In these situations we would like the results to take into account these views or preferences.

The algorithm presented here allows the user to specify the way he looks at the data. The views are formulated as criteria for determining weights on columns, or on rows, or on both columns and rows. Thus the user specifies a criterion for determining a vector $\mathbf{w}$ that states how the columns of $\mathbf{A}$ should be treated. Similarly, the user can specify a criterion for determining a vector $\mathbf{z}$, that states how the rows should be looked at. The fundamental idea of the present algorithm is that the user specifies criteria for determining $\mathbf{w}_i$, how columns should be treated, and $\mathbf{z}_i$, how rows should be treated, such that $\mathbf{w}_i$ and $\mathbf{z}_i$ reflect the views we have on the given data. The algorithm then orthogonalizes $\mathbf{A}$ with respect to these given choices, $\mathbf{z}_i'\mathbf{A}_j = \mathbf{0}$ and $\mathbf{A}_j\mathbf{w}_i = \mathbf{0}$ for $j \geq i$. This means that we, at each step, extract as much as possible from the given data and adjust the data so that they do not depend on what already has been selected. This means that we select "the best possible" at each step, and we stop when there is nothing more to fetch from data. This means that the algorithm is good at identifying the noise level in data, i.e., the situation where we cannot select more useful results from data. One can ask, "What about the numerical accuracy of the algorithm?" The answer is that, if the data are appropriately scaled from the beginning, we identify the noise level (where we stop) long before there are problems with the numerical accuracy.

The algorithm has been applied to numerous data from practice. What is the experience from these cases? The general experience is that the algorithm functions better than corresponding traditional approaches. This holds especially for large matrices of low rank. But in many cases it can be difficult for the user to know when to stop in the algorithm. Therefore,

we give some criteria and guidelines that we have found useful in determining when to stop in the algorithm.

The present algorithm is a decomposition of the coefficient matrix $\mathbf{A}$ into rank-one components according some choices of the criterion vectors $\mathbf{w}_i$ and $\mathbf{z}_i$. For special choices of $\mathbf{w}_i$ and $\mathbf{z}_i$ one can arrive at most decompositions found in numerical analysis and statistics, but it is not the purpose of the present paper to show that. However, we will point out some cases in which the connection may be of interest.

The emphasis here is on a unified approach to data analysis, matrix decompositions, and computation of the generalized inverse. The practical situation may or may not dictate the approach to choose. The algorithm and a stopping criterion identify the noise level, and we arrive at a decomposition that in a certain sense reflects the views on data as well as possible.

Since the algorithm is very fast and contains at most as many steps as the number of columns or rows in $\mathbf{A}$, we frequently find that scientists like to see more than one decomposition before they decide upon which to use. The scientist looks at different measures of quality and sizes and at several plots; on the basis of these, one decides which decomposition to use.

There are two situations considered in this paper. In the first situation we have only a given data matrix $\mathbf{A}$. In the second one, we need to solve a linear equation $\mathbf{Ax} = \mathbf{b}$. The algorithm is the same for both. In the case of a linear equation one more set of equations needs to be computed, namely, the present solution vector (step 6^o in the algorithm).

2. Notation. Let $\mathbf{A}$ be a rectangular matrix, an n-times-k matrix. We will write

$$\mathbf{A} = (\mathbf{a}_j) = (\mathbf{a}^i) = (a_{ij}).$$

Here $\mathbf{a}_j$ is the jth column vector of $\mathbf{A}$, $\mathbf{a}^i$ the ith row vector and a_{ij} is element number (i, j). A decomposition of $\mathbf{A}$ into rank-one components can be written as

$$\begin{aligned} \mathbf{A} &= \lambda_1 \mathbf{u}_1 \mathbf{v}_1' + \lambda_2 \mathbf{u}_2 \mathbf{v}_2' + \cdots \\ &= \mathbf{U}\Lambda\mathbf{V}'. \end{aligned} \tag{1}$$

Here $\mathbf{U} = (\mathbf{u}_i)$ is a matrix with n rows, $\Lambda = \operatorname{diag}(\lambda_i)$, a diagonal matrix with λ_i in the ith, diagonal and $\mathbf{V} = (\mathbf{v}_i)$ a matrix with k rows. The number of components in (1) is equal to the number of columns in $\mathbf{U}$, Λ, and $\mathbf{V}$. The last equality of (1) follows from the rules of matrix multiplication. When a new set of $(\lambda_i, \mathbf{u}_i, \mathbf{v}_i)$ is computed in the algorithm below, we will sometimes say that we are "extracting a component from $\mathbf{A}$." Note that if $\mathbf{A}$ is of full rank and $n > k$, the number of terms in (1) is equal to k. Thus, e.g., $\mathbf{U}$ will be an n-times-k matrix like $\mathbf{A}$, which shows that there is a lot of indeterminacy in (1). We utilize this indeterminacy to compute the decomposition reflecting the problem in question. $\mathbf{A}'$ and $\mathbf{v}'$ denote the transposes of $\mathbf{A}$ and $\mathbf{v}$, respectively.

3. Scaling of data values. If the units for different parts of data deviate much, it may be necessary to scale the data values. If we are using a criterion for columns, we consider the variation in the numbers $(\mathbf{a}_1'\mathbf{a}_1), \ldots, (\mathbf{a}_k'\mathbf{a}_k)$. If these values are very different, the data values in $\mathbf{A}$ should be scaled. It is common to scale the data values such that $\mathbf{a}_i'\mathbf{a}_i = 1$. Similarly, if we are using a row criterion, it may be necessary to scale the rows. If all values are given in the same units, scaling of data is not needed. For example, if all values are concentrations with values between 0 and 1, scaling is not needed. If the rank is low, one should be careful with scaling data. In order to show the problems involved, suppose that $\mathbf{A}$ has rank two and $\mathbf{u}_1'\mathbf{u}_2 = 0$ in the decomposition (1). Then we get

$$\mathbf{A}'\mathbf{A} = \lambda_1^2(\mathbf{u}_1'\mathbf{u}_1)\mathbf{v}_1\mathbf{v}_1' + \lambda_2^2(\mathbf{u}_2'\mathbf{u}_2)\mathbf{v}_2\mathbf{v}_2'.$$

Scaling $\mathbf{A}$ such that the diagonal elements of $\mathbf{A}'\mathbf{A}$ consist of ones means that there are constraints on the $\mathbf{v}$-vectors. If we know $\mathbf{\Lambda}$ and $\mathbf{U}$, $\mathbf{v}_2$ can be derived from $\mathbf{v}_1$. Thus scaling of columns puts constraints on the $\mathbf{v}$-vectors and if we have only a few $\mathbf{v}$-vectors, we may not be able to use the interpretations that the plots of $\mathbf{v}$-vectors suggest.

4. The singular value decomposition. The singular value decomposition (SVD) is the most used decomposition method in practice. Since the decompositions generated by the present algorithm are of a type similar to the SVD, it is useful to look closer at it. It is a decomposition (1), where the vectors $(\mathbf{u}_i)$ and $(\mathbf{v}_i)$ are chosen to be orthogonal,

$$\mathbf{u}_i'\mathbf{u}_j = 0 \quad \text{and} \quad \mathbf{v}_i'\mathbf{v}_j = 0 \quad \text{for } i \neq j$$

and

$$\mathbf{u}_i'\mathbf{u}_i = 1 \quad \text{and} \quad \mathbf{v}_i'\mathbf{v}_i = 1.$$

The vectors $(\mathbf{u}_i)$ and $(\mathbf{v}_i)$ are the eigenvectors to the eigenvalue problems

$$\mathbf{A}'\mathbf{A}\mathbf{v}_i = \lambda_i^2\mathbf{v}_i \quad \text{and} \quad \mathbf{A}\mathbf{A}'\mathbf{u}_i = \lambda_i^2\mathbf{u}_i.$$

The singular values (λ_i) have the appealing interpretation

$$\lambda_1^2 = \max \|\mathbf{A}\mathbf{w}\|^2 = \max \|\mathbf{A}'\mathbf{z}\|^2 = \max(\mathbf{z}'\mathbf{A}\mathbf{w})^2 \tag{2}$$

for $\|\mathbf{w}\| = 1$ and $\|\mathbf{z}\| = 1$, $\mathbf{w}$ a k vector and $\mathbf{z}$ an n vector.

Here $\|\mathbf{w}\|$ denotes the length of a vector $\mathbf{w}$. The solutions to (2) are given by $\mathbf{w} = \mathbf{v}_i$ and $\mathbf{z} = \mathbf{u}_i$. This means that the first pair, $\mathbf{u}_1$ and $\mathbf{v}_1$, is a set of n-vector and k-vector that gives maximal rank-one reduction, λ_1^2, of $\mathbf{A}$,

$$\operatorname{tr}(\mathbf{A}'\mathbf{A}) = \lambda_1^2 + \lambda_2^2 + \lambda_3^2 + \cdots. \tag{3}$$

$\lambda_1^2 = (\mathbf{u}_1'\mathbf{A}\mathbf{v}_1)^2$ is the maximal size of all possible rank-one reductions of $\mathbf{A}$. The next set $(\mathbf{u}_2, \mathbf{v}_2)$ is selected to be orthogonal to $\mathbf{u}_1$ and $\mathbf{v}_1$, respectively, and gives a maximal reduction in $\mathbf{A}$, λ_2^2, i.e., the largest possible second term in (3). (Here $\operatorname{tr}(\mathbf{A}'\mathbf{A})$ denotes the trace of a matrix $\mathbf{A}'\mathbf{A}$.) This interpretation of the SVD is important and easy to understand.

Most technical literature uses the SVD to determine the generalized inverse, because it has a very simple form,

$$\mathbf{A}^\dagger = 1/\lambda_1\mathbf{v}_1\mathbf{u}_1' + 1/\lambda_2\mathbf{v}_2\mathbf{u}_2' + \cdots. \tag{4}$$

But, as mentioned earlier, in many situations it need not be the proper generalized inverse to use, because the situation may dictate another generalized inverse.

5. The basic algorithm. The basic algorithm is somewhat complicated because it allows judgement of rows, columns, or both. We suppose that we have some view or criterion of treating column vectors $(\mathbf{a}_i)$ of $\mathbf{A}$. $(\mathbf{w}_i)$-vectors reflect this view. Similarly, some principle or view determines $(\mathbf{z}_i)$, which states how to look at the row vectors of $\mathbf{A}$.

Let $\mathbf{A}_o = \mathbf{A}$, $\mathbf{x}_o = \mathbf{0}$ and let $\mathbf{E}_o = \mathbf{I}_k$ and $\mathbf{B}_o = \mathbf{I}_n$, where $\mathbf{I}_k$ and $\mathbf{I}_n$ are the identity matrices of size k and n, respectively. The steps of the algorithm are as follows:

For $i = 1, 2, \ldots$

1°. **Determine criterion vector for columns.**

Choose the k-vector $\mathbf{w}_i$ that reflects how columns of $\mathbf{A}$ should be treated.

2°. **Determine criterion vector for rows.**
Choose the n-vector $\mathbf{z}_i$ that reflects how rows of $\mathbf{A}$ should be treated.

3°. **Compute components.**
Compute

$$\mathbf{u}_i = \mathbf{A}_{i-1}\mathbf{w}_i, \qquad \mathbf{v}_i = \mathbf{A}_{i-1}{}'\mathbf{z}_i, \quad \text{and} \quad \lambda_i = 1/(\mathbf{z}_i{}'\mathbf{A}_{i-1}\mathbf{w}_i).$$

4°. **Rank-one reduction of A.**

$$\mathbf{A}_i = \mathbf{A}_{i-1} - \lambda_i \mathbf{u}_i \mathbf{v}_i{}'.$$

5°. **Compute the transformation vectors.**

$$\mathbf{r}_i = \mathbf{E}_{i-1}\mathbf{w}_i, \qquad \mathbf{s}_i = \mathbf{B}_{i-1}\mathbf{z}_i,$$

Update the transformation matrices.

$$\mathbf{E}_i = \mathbf{E}_{i-1} - \lambda_i \mathbf{r}_i \mathbf{v}_i{}', \qquad \mathbf{B}_i = \mathbf{B}_{i-1} - \lambda_i \mathbf{s}_i \mathbf{u}_i{}'.$$

6°. **In case of solving a linear equation, $\mathbf{Ax} = \mathbf{b}$.**
Compute the improved solution vector.

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \lambda_i(\mathbf{b}'\mathbf{s}_i)\mathbf{r}_i = \lambda_1(\mathbf{b}'\mathbf{s}_1)\mathbf{r}_1 + \cdots + \lambda_i(\mathbf{b}'\mathbf{s}_i)\mathbf{r}_i.$$

7°. **Determine if a new iteration should be carried out**. Determine from the criteria used if a new iteration is to be carried out. In that case set i to $i + 1$ and start from 1°.

Discussion of the algorithm. We shall now discuss the individual steps in the algorithm.

1°. We suppose that there is a criterion that determines $\mathbf{w}_i$. The criterion states how we look at the columns of $\mathbf{A}$. If the user does not specify such a column criterion, we choose $\mathbf{w}_i = \mathbf{v}_i$.

2°. Similarly, we suppose that there is a criterion that determines $\mathbf{z}_i$, which shows how rows should be looked at. If the user does not specify such a row criterion, we choose $\mathbf{z}_i = \mathbf{u}_i$.

3°. The vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are the resulting vectors,

$$\mathbf{u}_i = \mathbf{A}_{i-1}\mathbf{w}_i = \mathbf{a}_i w_{1i} + \mathbf{a}_2 w_{2i} + \cdots$$

and

$$\mathbf{v}_i = \mathbf{A}_{i-1}{}'\mathbf{z}_i = \mathbf{a}^1 z_{1i} + \mathbf{a}^2 z_{2i} + \cdots.$$

These vectors are the best we can select from $\mathbf{A}$ and the value of a common criterion is $1/\lambda_i = \mathbf{z}_i{}'\mathbf{A}_{i-1}\mathbf{w}_i$. The larger the numerical value of this common criterion is, the better are the vectors $\mathbf{u}_i$ and $\mathbf{v}_i$. If the value is zero, we cannot reduce $\mathbf{A}$ more by using the given criteria.

4°. Here we reduce $\mathbf{A}$ by the vectors selected. That it always gives a rank-one reduction of $\mathbf{A}$, when $\mathbf{z}_i{}'\mathbf{A}_{i-1}\mathbf{w}_i$ is different from zero, is a well-known theorem of Laplace. If all possible vectors are selected, we get (1).

5°. The transformation vectors $\mathbf{r}_i$ and $\mathbf{s}_i$ are needed in order to compute the generalized inverse that is associated with the decomposition. The vector $\mathbf{r}_i$ is determined so that $\mathbf{u}_i = \mathbf{A}\mathbf{r}_i$ and $\mathbf{r}_i$ is a linear combination of $\mathbf{w}_1, \ldots, \mathbf{w}_i$. We write

$$\begin{aligned}\mathbf{u}_i = \mathbf{A}\mathbf{r}_i &= \mathbf{A}_{i-1}\mathbf{w}_i = \mathbf{A}_{i-2}(\mathbf{I} - \lambda_{i-1}\mathbf{w}_{i-1}\mathbf{v}_{i-1}{}')\mathbf{w}_i \\ &= \mathbf{A}(\mathbf{I} - \lambda_1\mathbf{w}_1\mathbf{v}_1{}')\cdots(\mathbf{I} - \lambda_{i-1}\mathbf{w}_{i-1}\mathbf{v}_{i-1}{}')\mathbf{w}_i = \mathbf{A}\,\mathbf{E}_{i-1}\mathbf{w}_i.\end{aligned}$$

For

$$\mathbf{E}_{i-1} = (\mathbf{I} - \lambda_1 \mathbf{w}_1 \mathbf{v}_1{}') \cdots (\mathbf{I} - \lambda_{i-1} \mathbf{w}_{i-1} \mathbf{v}_{i-1}{}')$$

we can write

$$\mathbf{E}_i = \mathbf{E}_{i-1}(\mathbf{I} - \lambda_i \mathbf{w}_i \mathbf{v}_i{}') = \mathbf{E}_{i-1} - \lambda_i \mathbf{r}_i \mathbf{v}_i{}'.$$

Similarly, $\mathbf{s}_i$ is determined so that $\mathbf{v}_i = \mathbf{A}'\mathbf{s}_i$ and $\mathbf{s}_i$ is a linear combination of $\mathbf{z}_1, \ldots \mathbf{z}_i$. An expansion of $\mathbf{A}_{i-1}$ given by

$$\begin{aligned}\mathbf{A}_{i-1} &= (\mathbf{I} - \lambda_{i-1} \mathbf{u}_{i-1} \mathbf{z}_{i-1}{}')\mathbf{A}_{i-2} \\ &= (\mathbf{I} - \lambda_{i-1} \mathbf{u}_{i-1} \mathbf{z}_{i-1}{}') \cdots (\mathbf{I} - \lambda_1 \mathbf{u}_1 \mathbf{z}_1{}')\mathbf{A} = \mathbf{B}_{i-1}{}'\mathbf{A}\end{aligned}$$

gives similarly the equation of $\mathbf{s}_i$ and the recursive one for $\mathbf{B}_i$.

6°. In the case where we are solving a linear equation, we use this step. The vector $\mathbf{x}_i$ is an estimate of the linear least squares solution $\mathbf{x} = (\mathbf{A}'\mathbf{A})^\dagger \mathbf{A}'\mathbf{b}$, when i terms are used in the expansion of $\mathbf{x}$.

The algorithm works with three pairs of vectors. The vectors $(\mathbf{w}_i)$ and $(\mathbf{z}_i)$ are determined from the criteria given by the practical problem in question. $(\mathbf{u}_i)$ and $(\mathbf{v}_i)$ reduce $\mathbf{A}$ by rank one, and $(\mathbf{r}_i)$ and $(\mathbf{s}_i)$ reduce $\mathbf{A}^\dagger$ by rank one. When $\mathbf{w}_i$ or $\mathbf{z}_i$ are selected, they can be scaled to be of length one. In the algorithm we do not scale $\mathbf{u}_i$ or $\mathbf{v}_i$. The scaling is taken care of by the constant λ_i. This is done in order to secure numerical stability of the algorithm. All computations in the algorithm are products and inner products except the computation of λ_i. This means that the algorithm can be programmed to give very precise numerical results.

In the case of SVD the $(\mathbf{w}_i)$ are the eigenvectors to the eigenvalue problem $\mathbf{A}'\mathbf{A}\mathbf{w} = \lambda\mathbf{w}$. In that case, $\mathbf{w}_i = \mathbf{v}_i = \mathbf{r}_i$ and $\mathbf{u}_i = \mathbf{z}_i = \mathbf{s}_i$ apart from a scaling constant. Note that in the SVD, $\mathbf{u}$- and $\mathbf{v}$-vectors are scaled to be of length one, while in the present algorithm they are not. See below what criterion gives SVD.

6. Geometric properties of the vectors. The vectors in the algorithm have the following geometric properties.

PROPERTY 1. $\mathbf{z}_i$ *is orthogonal to later* $\mathbf{A}$*-matrices,*

$$\mathbf{z}_i{}'\mathbf{A}_j = \mathbf{0} \quad \textit{for } i \leq j. \tag{5}$$

PROPERTY 2. $\mathbf{z}_i$ *is orthogonal to later* $\mathbf{u}$*-vectors,*

$$\mathbf{z}_i{}'\mathbf{u}_j = \mathbf{0}, \qquad i < j. \tag{6}$$

PROPERTY 3. $\mathbf{w}_i$ *is orthogonal to later* $\mathbf{A}$*-matrices,*

$$\mathbf{A}_j\mathbf{w}_i = \mathbf{0}, \qquad i \leq j. \tag{7}$$

PROPERTY 4. $\mathbf{w}_i$ *is orthogonal to later* $\mathbf{v}$*-vectors,*

$$\mathbf{w}_i{}'\mathbf{v}_j = \mathbf{0}, \qquad i < j. \tag{8}$$

PROPERTY 5. $(\mathbf{s}_i)$ *and* $(\mathbf{u}_i)$ *are mutually orthogonal,*

$$\mathbf{s}_i{}'\mathbf{u}_j = \mathbf{0} \quad \textit{and} \quad \mathbf{s}_i{}'\mathbf{u}_i = 1/\lambda_i, \qquad i \neq j. \tag{9}$$

PROPERTY 6. $(\mathbf{r}_i)$ *and* $(\mathbf{v}_i)$ *are mutually orthogonal,*

$$\mathbf{r}_i'\mathbf{v}_j = \mathbf{0}, \quad \mathbf{r}_i'\mathbf{v}_i = 1/\lambda_i \quad \textit{for } i \neq j. \tag{10}$$

PROPERTY 7. *If* $(\mathbf{w}_i)$ *is chosen such that it is in the range of* $\mathbf{A}_{i-1}'$, $(\mathbf{w}_i)$ *are mutually orthogonal*

$$\mathbf{w}_i'\mathbf{w}_j = \mathbf{0} \quad \textit{for } i \neq j.$$

PROPERTY 8. *The generalized inverse of* $\mathbf{A}$ *can be computed as*

$$\begin{aligned} \mathbf{A}^\dagger &= \lambda_1\mathbf{r}_1\mathbf{s}_1' + \lambda_2\mathbf{r}_2\mathbf{s}_2' + \cdots \\ &= \mathbf{R}\Lambda\mathbf{S}'. \end{aligned} \tag{11}$$

These properties are proved in Appendix 1.

Properties 1 and 3 are the most basic properties. They show that the reduction of $\mathbf{A}$ can be viewed such that $\mathbf{A}$ is adjusted for the actual choices of $\mathbf{w}_i$ and $\mathbf{z}_i$. The rows of $\mathbf{A}$ are made orthogonal to $\mathbf{w}_i$ and columns of $\mathbf{A}$ are made orthogonal to $\mathbf{z}_i$. Usually $\mathbf{w}_i$ is chosen to be in the range of $\mathbf{A}_{i-1}'$, which gives orthogonal set $(\mathbf{w}_i)$. By symmetry, if $\mathbf{z}_i$ is chosen in the range of $\mathbf{A}_{i-1}$, the set $(\mathbf{z}_i)$ becomes an orthogonal set of vectors. Note that the coefficients (λ_i) in (11) differ from the corresponding ones in (4), because the vectors $(\mathbf{r}_i)$ and $(\mathbf{s}_i)$ are not scaled to be of unit length.

In terms of matrices, Properties 5 and 6 can be written as

$$\mathbf{R}'\mathbf{V} = \Lambda^{-1} \quad \text{and} \quad \mathbf{S}'\mathbf{U} = \Lambda^{-1},$$

where $\mathbf{R} = (\mathbf{r}_i)$, $\mathbf{V} = (\mathbf{v}_i)$, $\mathbf{S} = (\mathbf{s}_i)$, $\mathbf{U} = (\mathbf{u}_i)$, and $\Lambda^{-1} = \text{Diag}(1/\lambda_i)$.

7. The quality of solutions: Example 1. We shall here consider some questions concerning the quality of solutions derived from the decompositions arrived at by the algorithm. We shall discuss the question of quality in the light of a small example. We suppose that the situation is that there is given a set of linear equations, $\mathbf{Ax} = \mathbf{b}$. The data are given as follows.

$$\mathbf{A} = \begin{matrix} -101.00001 & 98.00001 & 101.00002 \\ -99.00001 & 102.00001 & 99.00002 \\ 98.99999 & -101.99999 & -98.99998 \\ 100.99999 & -97.99999 & -100.99998 \end{matrix}, \quad \mathbf{b} = \begin{matrix} 98.00002 \\ 102.00002 \\ -101.99998 \\ -97.99998 \end{matrix}.$$

Here $\mathbf{A}$ has approximately the rank two ($\mathbf{a}_3 \approx -\mathbf{a}_1$). But the exact numerical solution can be computed and it is given by $\mathbf{x} = (1, 1, 1)$. Although this example is small, it is representative for the situations in which large data matrices are involved. If we use the column criterion (17) below and no row criterion, the solution vector $\mathbf{x}_i$ obtains the following values in the algorithm.

Solution vector $\mathbf{x}_i$ at iterations 1 – 3:

1	2	3
−0.33326669332267	0.00000000000287	1.00000000013348
0.33346669333867	0.99999999999445	1.00000000000000
0.33326669332268	0.00000000000844	1.00000000013349

Here we would normally select as a solution the one obtained at the second iteration, $\mathbf{x}_2$. We note that the solution $\mathbf{x}_3$ is close to the exact solution, while $\mathbf{x}_2$ differs much from the exact solution. When should we select as exact a solution as possible, and when should we select as stable a solution as possible? In cases where we have a technical construction and the coefficients are exact numbers derived from the given construction, we should select the exact solution. But in most cases in practice the coefficients are uncertain. In these cases we should select as good a solution as possible. There are three aspects of a good solution that we shall consider here.

I. *Sizes of criteria used.*

(1) Size of the column criterion.

(2) Size of the row criterion.

(3) The combined criteria, $\mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i$.

Interpretation of (3) is similar to the one in the case of the SVD, i.e., the variation extracted by the row and column criteria. In the example above $\mathbf{z}_3'\mathbf{A}_2\mathbf{w}_3 = 2^*10^{-10}$. This shows that the third component, $(\lambda_3, \mathbf{u}_3, \mathbf{v}_3)$, is not explaining any variation in data.

II. *Sizes of results.*

(1) Residuals, $\mathbf{A}_{i-1}$ or $\mathbf{A}\mathbf{x}_i - \mathbf{b}$.

(2) Size of reduction in $\mathbf{A}$ and/or $\mathbf{b}$ due to the rank-one reduction.

(3) Size of extracted matrices, $\mathbf{A} - \mathbf{A}_i$ and $(\mathbf{A} - \mathbf{A}_i)^\dagger$.

Here the elements of $\mathbf{A}\mathbf{x}_2 - \mathbf{b}$ are less than or equal to 10^{-5}. Thus $\mathbf{x}_2$ gives sufficiently small error estimates. The precision of the solution vector $\mathbf{x}_i$ can be expressed in terms of $((\mathbf{A} - \mathbf{A}_i)'(\mathbf{A} - \mathbf{A}_i))^\dagger$, see Appendix 2. Since the rank of $\mathbf{A}$ is approximately two, the inverse increases extremely, when adding the third component. This means that $\mathbf{x}_3$ is a very unstable solution.

III. *Measures of performance.*

(1) Total prediction variance,

(a) $\|\mathbf{A}_i\|_F{}^2\|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2$, (b) $\|\mathbf{A}\mathbf{x}_i - \mathbf{b}\|^2\|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2$.

(2) Total prediction error variance,

(a) $\|\mathbf{A}_i\|_F{}^2(1 + \|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2)$, (b) $\|\mathbf{A}\mathbf{x}_i - \mathbf{b}\|^2(1 + \|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2)$.

(3) The H-index,

(a) $\|\Delta\mathbf{A}_i\|_F{}^2/\|\Delta\mathbf{A}_i{}^\dagger\|_F{}^2$, (b) $\|\Delta(\mathbf{A}\mathbf{x}_i)\|^2/\|\Delta\mathbf{A}_i{}^\dagger\|_F{}^2$.

Since the size of $(\mathbf{A} - \mathbf{A}_2)^\dagger$ is small, while that of $(\mathbf{A} - \mathbf{A}_3)^\dagger$ is very large, $\mathbf{x}_3$ when it is applied, will have very bad performance.

The formulas (a) refer to the case in which we have only $\mathbf{A}$, while (b) refers to the case of solving a linear equation. The motivation for these three measures is given in Appendix 2. Note that in the algorithm $\mathbf{A}_i$ is the residual matrix, when i components have been extracted, while $(\mathbf{A} - \mathbf{A}_i)$ represents what has been extracted.

8. Criteria based on columns only. In many situations we wish to select only $\mathbf{w}_i$. This happens when there are not big differences from row to row in $\mathbf{A}$. In many statistical analyses the rows of $\mathbf{A}$ can be viewed as repeated samples and thus expected to be of approximately same size. If we have no special information or theory on the variation of the row, it is natural to seek only to determine $\mathbf{w}_i$.

In this case we have

$$\mathbf{u}_i = \mathbf{A}_{i-1}\mathbf{w}_i, \qquad \mathbf{z}_i = \mathbf{s}_i = \mathbf{u}_i, \quad \text{and} \quad \mathbf{v}_i = \mathbf{A}_{i-1}{}'\mathbf{u}_i.$$

From Property 2 we get that $(\mathbf{u}_i)$ are mutually orthogonal:

$$\mathbf{u}_i{}'\mathbf{u}_j = 0 \qquad \text{for } i \neq j,$$

and from the definition of λ_i we get

$$\lambda_i = 1/(\mathbf{u}_i'\mathbf{u}_i). \tag{12}$$

Since $\mathbf{A}_{i-1}'\mathbf{u}_i = \mathbf{A}'\mathbf{u}_i$, it follows that $\mathbf{s}_i = \mathbf{u}_i$. The size H_i of the reduction in $\mathbf{A}$ is given by

$$\begin{aligned} H_i &= \operatorname{tr}[(\lambda_i\mathbf{u}_i\mathbf{v}_i')(\lambda_i\mathbf{u}_i\mathbf{v}_i')'] = \lambda_i^2(\mathbf{u}_i'\mathbf{u}_i)(\mathbf{v}_i'\mathbf{v}_i) = (\mathbf{v}_i'\mathbf{v}_i)/(\mathbf{u}_i'\mathbf{u}_i) \\ &= (\mathbf{u}_i'\mathbf{u}_i)\|\mathbf{A}_{i-1}'\mathbf{A}_{i-1}\mathbf{w}_i\|^2/(\|\mathbf{A}_{i-1}\mathbf{w}_i\|^2)^2. \end{aligned} \tag{13}$$

If $\mathbf{w}_i$ is any eigenvector of $\mathbf{A}_{i-1}'\mathbf{A}_{i-1}$ of unit length, H_i will be equal to $(\mathbf{u}_i'\mathbf{u}_i) = \mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i$. Often the $\mathbf{w}$-vector will be close to an eigenvector of $\mathbf{A}_{i-1}'\mathbf{A}_{i-1}$, and therefore H_i will be approximately the squared size of the $\mathbf{u}$-vector. Also H_i is invariant to the scale of $\mathbf{w}_i$, and thus $\mathbf{w}_i$ can always be chosen to be of length one.

It is often convenient to work with the algorithm in terms of the positive definite matrix associated with $\mathbf{A}$, namely, $\mathbf{A}'\mathbf{A}$, i.e., not to compute the vectors $\mathbf{u}_i$. This is the case when we want to work with the normal equations only: $\mathbf{A}'\mathbf{A}\mathbf{x} = \mathbf{A}'\mathbf{b}$. The algorithm can now be formulated as follows:

0°. **Compute** $\mathbf{Q} = \mathbf{A}'\mathbf{A}$ and let $\mathbf{Q}_o = \mathbf{Q}$.

1°. **Determine** $\mathbf{w}_i$.
Select $\mathbf{w}_i$, e.g., according to some criterion.

2°. **Compute components**.

$$\mathbf{v}_i = \mathbf{Q}_{i-1}\mathbf{w}_i \quad \text{and} \quad \lambda_i = 1/(\mathbf{w}_i'\mathbf{Q}_{i-1}\mathbf{w}_i).$$

3°. **Rank-one reduction of** $\mathbf{Q}$.

$$\mathbf{Q}_i = \mathbf{Q}_{i-1} - \lambda_i\mathbf{v}_i\mathbf{v}_i'.$$

4°. **Compute the transformation vector**.

$$\mathbf{r}_i = \mathbf{E}_{i-1}\mathbf{w}_i.$$

Update the transformation matrix.

$$\mathbf{E}_i = \mathbf{E}_{i-1} - \lambda_i\mathbf{r}_i\mathbf{v}_i'.$$

5°. **In case of solving a linear equation,** $\mathbf{Ax} = \mathbf{b}$.
Compute the improved solution vector, $\mathbf{q} = \mathbf{A}'\mathbf{b}$,

$$\mathbf{x}_i = \mathbf{x}_{i-1} + \lambda_i(\mathbf{q}'\mathbf{r}_i)\mathbf{r}_i = \lambda_1(\mathbf{q}'\mathbf{r}_1)\mathbf{r}_1 + \cdots + \lambda_i(\mathbf{q}'\mathbf{r}_i)\mathbf{r}_i.$$

6°. **Determine if a new iteration should be carried out**.
If a new component is to be selected, $i \to i + 1$ and go to 1°.

This form of the algorithm is based on the properties that the $\mathbf{u}$-vectors are orthogonal and that λ_i is given by (12). If the matrix $\mathbf{A}'\mathbf{A}$ is fully reduced, we have

$$\mathbf{A}'\mathbf{A} = \lambda_1\mathbf{v}_1\mathbf{v}_1' + \lambda_2\mathbf{v}_2\mathbf{v}_2' + \cdots,$$

$$(\mathbf{A}'\mathbf{A})^\dagger = \lambda_1\mathbf{r}_1\mathbf{r}_1' + \lambda_2\mathbf{r}_2\mathbf{r}_2' + \cdots.$$

This form of the algorithm is useful if we are updating the decomposition with a new row of $\mathbf{A}$, or analyzing the dependence of the decomposition on one or more rows. This follows from

$$\mathbf{A}'\mathbf{A} = \mathbf{a}^1\mathbf{a}^{1\prime} + \mathbf{a}^2\mathbf{a}^{2\prime} + \cdots + \mathbf{a}^n\mathbf{a}^{n\prime}. \tag{14}$$

Using (14) and the above form of the algorithm makes it easy to update the decomposition when a new row is added to $\mathbf{A}$. If $\mathbf{a}^{n+1}$ is a new row, we compute the new $\mathbf{Q}$-matrix as $\mathbf{Q} = \mathbf{A}'\mathbf{A} + \mathbf{a}^{n+1}\mathbf{a}^{n+1\prime}$ and start from 1° in the modified algorithm.

8.1. Some criteria based on A only. In order to simplify the notation we shall drop the indices associated with each step in the algorithm. We are to determine a vector $\mathbf{w}$ such that

$$\mathbf{u} = \mathbf{A}\mathbf{w} = w_1\mathbf{a}_1 + w_2\mathbf{a}_2 + \cdots + w_k\mathbf{a}_k$$

has some desirable properties or characteristics. There are some views that are common in practice that we will review.

8.1.1. Variation in u. One view is to determine $\mathbf{w}$ such that the variation in the values in the $\mathbf{u}$-vector has some properties. One is to require that the length of $\mathbf{u}$ is as large as possible for $|\mathbf{w}| = 1$. This is the criterion of the SVD. The power method, see [3], can be used to determine $\mathbf{w}$, because it gives one eigenvector at a time.

8.1.2. Selection of variables. Frequently, each column in the data table contains repeated measurements of a given physical magnitude. In practice we say that we have a variable and the column represents measurements of the variable at different instances. A basic question is often, "Which variables (columns) are important?" In our setup it corresponds to choosing $\mathbf{w}$ to be a unit vector having zeros at all coordinates except one, which is equal to 1. The coordinate equal to 1 corresponds to the variable that is selected. Let $\mathbf{w}$ be given by

$$\mathbf{w}' = (0, 0, \ldots, 0, 1, 0, \ldots)$$

with 1 at the jth coordinate. With this notation we have

$$\mathbf{u} = \mathbf{a}_j, \quad \mathbf{v}' = (\mathbf{a}_j'\mathbf{a}_1, \mathbf{a}_j'\mathbf{a}_2, \ldots) \quad \text{and} \quad \lambda = 1/(\mathbf{a}_j'\mathbf{a}_j).$$

This gives H_j of (13) equal to

$$H_j = [(\mathbf{a}_j'\mathbf{a}_1)^2 + (\mathbf{a}_j'\mathbf{a}_2)^2 + \cdots + (\mathbf{a}_j'\mathbf{a}_k)^2]/(\mathbf{a}_j'\mathbf{a}_j). \tag{15}$$

In [4] the H-principle, the Heisenberg principle of mathematical modelling, is formulated and applied to different modelling tasks. According to the H-principle we should consider

$$\begin{aligned} &\{\text{size of reduction in } \mathbf{A}\}\{\text{size of the } \mathbf{u}\text{-vector}\} \\ &= H_j(\mathbf{u}_j'\mathbf{u}_j) \\ &= (\mathbf{a}_j'\mathbf{a}_1)^2 + (\mathbf{a}_j'\mathbf{a}_2)^2 + \cdots + (\mathbf{a}_j'\mathbf{a}_k)^2. \end{aligned} \tag{16}$$

Thus the criterion of determining a column of $\mathbf{A}$ is

$$\underset{j}{\text{maximize}}\,[(\mathbf{a}_j'\mathbf{a}_1)^2 + (\mathbf{a}_j'\mathbf{a}_2)^2 + \cdots + (\mathbf{a}_j'\mathbf{a}_k)^2] \quad \text{for} \quad j = 1, 2, \ldots, k. \tag{17}$$

The intuitive argument for criterion (17) is that the size of the reduction of $\mathbf{A}$, H_j, does not depend on the length of the $\mathbf{u}$-vector, only on its direction (H_j does not change if $\mathbf{u}$ is replaced by $c\mathbf{u}$, where c is a constant different from zero). Thus if we only use (15) as a criterion, the risk is that we select a column that is small and will "blow up" (15) because it is small. On the other hand, a large $\mathbf{u}$-vector need not be good to "explain" the remaining columns in the $\mathbf{A}$-matrix. The criterion (17) is balancing these two opposite points of view on the modelling task.

In the literature there have been many suggestions of how to select columns. An interesting one is the concept of "pure variables," see [5]. The basic idea is to consider σ_i/m_i, where σ_i is the standard deviation of the values in the ith column and m_i the average value (where m_i is close to zero, it is replaced by $(m_i + \epsilon)$). The purest variable is the one where this is smallest. When a pure variable has been selected, the $\mathbf{A}$-matrix is adjusted for the selected column, as in the present algorithm. The disadvantage of the concept of pure variables is that a pure variable need not be good in explaining other variables. Thus the concept of pure variables should be modified in a way similar to the way in which we treated the reduction H_j due to a variable.

8.1.3. w as weight coefficients. It was shown in the algorithm that the rows of reduced **A** become orthogonal to the **w**-vectors that have been selected. Sometimes the data or the situation dictates some weights that are appropriate to use. In [6] the concept of a marker is introduced, which is a row of **A** that is "important" and appropriate to use as relative weights. Another choice of **w** consists of ones, $\mathbf{w} = (1, 1, \ldots, 1)$. Here we are actually centering the rows of **A** because all rows in reduced **A**'s will have sum zero.

Criteria other than those mentioned above can be used. Also the above criteria can be mixed. For example, it sometimes happens that in the selection of variables once some variables have been selected, none of the remaining ones are doing a good job, though we are clearly not at the noise level. In these cases it can be advantageous to start selecting **w** according to the variation in **u** like, e.g., the eigenvectors of the reduced **A** matrix (eigenvector of $\mathbf{A}_{i-1}'\mathbf{A}_{i-1}$).

We shall now consider more closely the case when **A** is not of full rank and we are selecting variables. Suppose that j variables have been selected and $\mathbf{A}_j$ is judged to be zero. To simplify the notation, we will suppose that it is the first j columns of **A** that have been chosen. Let **A** be partitioned as $\mathbf{A} = (\mathbf{A}_1, \mathbf{A}_2)$, where $\mathbf{A}_1$ is the first j columns of **A**. If **V** is partitioned in a similar way, $\mathbf{V}' = (\mathbf{V}_1', \mathbf{V}_2')$, where $\mathbf{V}_1$ is j times j, we get from $\mathbf{A} = \mathbf{UDV}'$,

$$\mathbf{A}_1 = \mathbf{UDV}_1' \quad \text{and} \quad \mathbf{A}_2 = \mathbf{UDV}_2'. \tag{18}$$

Here **U** has j orthogonal columns. We are here selecting columns and $(\mathbf{DV}_1')$ is upper triangular with 1 in its diagonal. Thus here the algorithm is numerically equivalent to the well-known QR algorithm in numerical analysis. Equation (18) also shows how the last columns of **A** depend on the first j ones:

$$\mathbf{A}_2 = \mathbf{A}_1\mathbf{V}_1'^{-1}\mathbf{V}_2'.$$

8.2. Selection of $\mathbf{w}_i$ in case of linear equations. The choice of **w** is one of the most extensively studied problems in numerical analysis and theoretical statistics. In numerical analysis it has been studied in relation to "conjugate" and "steepest descent" methods, see [7]. In statistics it is known as variable or component selection, see [8], [9], and [11]. We shall only mention here how the H-principle suggests that we choose **w**. Similar to (16), we should consider

$$\{\text{reduction in variation in } \mathbf{b}\}\{\text{size of the } \mathbf{u}\text{-vector}\}. \tag{19}$$

The reduction in variation in **b** is $(\mathbf{b}'\mathbf{u})^2/(\mathbf{u}'\mathbf{u})$ (the squared length of the projection of **b** onto **u**). Thus (19) becomes

$$[(\mathbf{b}'\mathbf{u})^2/(\mathbf{u}'\mathbf{u})](\mathbf{u}'\mathbf{u}) = (\mathbf{b}'\mathbf{u})^2 = (\mathbf{b}'\mathbf{Aw})^2. \tag{20}$$

Thus the H-principle suggests choosing **w** so that it maximizes (20), which has the solution

$$\mathbf{w} = \mathbf{A}'\mathbf{b}/\|\mathbf{A}'\mathbf{b}\|. \tag{21}$$

This choice of **w** is the criterion of partial least squares (PLS) regression, see [12]. The intuitive argument for (19) is the same as for (16). With this choice of **w** they become orthogonal, $\mathbf{w}_i\mathbf{w}_j = 0$ for $i \neq j$.

Example 1, cont.

We shall look at the properties of the solution shown earlier, where **w** was chosen as given in (21). If **w** in (21) is inserted in (20), we get $(\mathbf{b}'\mathbf{Aw})^2 = \|\mathbf{A}'\mathbf{b}\|^2$. Let us look more closely at Table 1.

TABLE 1

i	Size of column criterion $\|\mathbf{A}_{i-1}'\mathbf{b}\|$	Combined criteria $\mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i$	Prediction variance	Prediction error variance
1	69282.035074038	119999.990401923	$0.01110667\ 10^{-7}$	15.99360064
2	19.5919995419	24.000001920	$0.00000695\ 10^{-7}$	0.0000000004
3	0.0000000003	0.000000000	$0.10097420\ 10^{-7}$	0.0000000101

The first two columns tell us how much we are selecting at each step of the algorithm. And we see that at the third step we are actually not selecting anything. The last two columns are approximate estimates of variances, when the linear model is applied. Generally, we select the number where they are smallest. We see that they are smallest at the second step, $i = 2$. This table therefore confirms the previous conclusion that we should carry out only two steps (iterations) in the algorithm. □

The approach given by (19) to (21) identifies the projection of $\mathbf{b}$ onto the linear column space of $\mathbf{A}$. If there is a very high degree of collinearity in the column space of $\mathbf{A}$, then this approach may not be the best approach. If large portions of columns of $\mathbf{A}$ depend linearly on the other columns of $\mathbf{A}$, it can be beneficial to identify the "good" linearly independent columns and use the approach given by (19) to (21) on these columns instead of the whole of $\mathbf{A}$. The reason for this is that $\mathbf{w}$ in (21) involves the linear combination of all columns in $\mathbf{A}$, and one can show that the redundant columns may worsen the quality of the solution. We shall here consider how to choose "good" columns of $\mathbf{A}$ using the criteria (19) and (20). Let $\mathbf{w} = (0, \ldots, 0, 1, 0, \ldots)$, where the jth element of $\mathbf{w}$ contains one and others are zero. Then $\mathbf{Aw} = \mathbf{a}_j$ and the criterion (20) becomes

$$\text{maximize } (\mathbf{b}'\mathbf{a}_j)^2 \quad \text{for} \quad j = 1, \ldots, k. \tag{22}$$

Thus the procedure here is to determine the column giving the maximum value of (22). We shall see how this functions for the small example we have considered.

Table 2, which is similar to Table 1, follows:

TABLE 2

Col	Size of column criterion $\|\mathbf{A}_{i-1}'\mathbf{b}\|$	Combined criteria $\mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i$	Prediction variance	Prediction error variance
2	40016.0000000008	40016.0000000004	$0.0000000000003\ 10^{-6}$	$0.0004\ 10^{-6}$
3	0.0000000004	35.9856057581	$0.0000012351853\ 10^{-6}$	$0.0004\ 10^{-6}$
1	0.0000000004	0.0000000004	$0.1312664544592\ 10^{-6}$	$0.1313\ 10^{-6}$

From the table we see that when we have selected the second column very little is left in data. Thus the results here suggest that only column number 2 should be used. □

9. Criteria based on rows only. Criteria that are the same or similar to the ones we have discussed previously can be used for the rows. But frequently we have different views on rows than on columns. We often expect some rows to be more important than some other rows. This can often be formulated in terms of weights determined by the row.

When there is given a criterion for rows only, we choose only the vectors $(\mathbf{z}_i)$. We have

$$\mathbf{v}_i = \mathbf{A}_{i-1}'\mathbf{z}_i, \quad \mathbf{w}_i = \mathbf{r}_i = \mathbf{v}_i, \quad \text{and} \quad \mathbf{u}_i = \mathbf{A}_{i-1}\mathbf{w}_i.$$

From Property 4 it follows that $(\mathbf{v}_i)$ are orthogonal:

$$\mathbf{v}_i'\mathbf{v}_j = 0 \quad \text{for } i \neq j.$$

From the definition of λ_i we get

$$\lambda_i = 1/\mathbf{v}_i'\mathbf{v}_i.$$

The size of the reduction in $\mathbf{A}$, which is similar to H_i of (13), is here given as

$$H_i = (\mathbf{u}_i'\mathbf{u}_i)/(\mathbf{v}_i'\mathbf{v}_i) = (\mathbf{v}_i'\mathbf{v}_i)\|\mathbf{A}_{i-1}\mathbf{A}_{i-1}'\mathbf{z}_i\|^2/[\|\mathbf{A}_{i-1}'\mathbf{z}_i\|^2]^2.$$

If $\mathbf{z}_i$ is any eigenvector of $\mathbf{A}_{i-1}\mathbf{A}_{i-1}'$ of unit length, the size of the reduction will be $(\mathbf{v}_i'\mathbf{v}_i) = \mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i$. Thus one can state that the size of the reduction when $\mathbf{z}_i$ is close to an eigenvector is approximately equal to $(\mathbf{v}_i'\mathbf{v}_i)$.

9.1. Sizes. An important type of choice of $\mathbf{z}_i$ is the size of rows of $\mathbf{A}$. An example is to let $\mathbf{z}_i$ be the length of rows of $\mathbf{A}$:

$$\mathbf{z}' = (\|\mathbf{a}^1\|, \|\mathbf{a}^2\|, \ldots, \|\mathbf{a}^n\|). \tag{23}$$

A way to look at $\mathbf{z}$ is provided in the following equation:

$$\mathbf{z}'\mathbf{z} = \|\mathbf{A}\|_F^2 = \operatorname{tr}(\mathbf{A}\mathbf{A}') = a_{11}^2 + a_{21}^2 + \cdots + a_{nk}^2.$$

Thus the squared length of $\mathbf{z}$ is equal to the sum of squares of the data that is present at each step of the algorithm.

Sometimes we want $\mathbf{v}_i$ to be one of the "important" rows of $\mathbf{A}$. For example, we may want $\mathbf{v}_i$ to be the row that has the largest value in $\mathbf{A}$. This can be achieved by setting $\mathbf{z}_i = (0, \ldots, 0, 1, 0, \ldots)$ with the index of one being the number of the row containing the largest value in $\mathbf{A}$.

9.2. Weights. Sometimes parts of data carry more information than other parts. This is often solved by defining $\mathbf{z}$ to be equal to one for the important part of data and smaller than one for other parts. In control theory, e.g., see [13], this kind of scaling is fundamental to most applications. There typically exponential decaying weights are used. But it is more natural to let the data themselves define the weights. We shall look at how the type of weights can be defined. Suppose that $\mathbf{a}^n$ is the important row (or the most "recent" row). If rows are close to $\mathbf{a}^n$, they will be given weight one, and weight less than one, if the rows differ much from $\mathbf{a}^n$. (Another view is to use the correlation coefficient between a row and $\mathbf{a}^n$ as criterion.) This can be achieved by defining for the vector $\mathbf{z} = (z_1, \ldots, z_n)$:

$$z_j = 1 \quad \text{for } \|\mathbf{a}^j - \mathbf{a}^n\| < c,$$
$$z_j = c/\|\mathbf{a}^j - \mathbf{a}^n\| \quad \text{for } \|\mathbf{a}^j - \mathbf{a}^n\| > c$$

for some constant c. There are other ways to define $\mathbf{z}$ that reflect views on the rows of $\mathbf{A}$. For example, there can be a "target," and we may want to use the same views on the rows as we treated the columns in the case of linear equations, where the $\mathbf{b}$-vector was the "target" vector.

10. Criteria based on rows and columns. All the criteria mentioned above can be combined and used on both rows and columns. A possible complication is the constant

$$1/\lambda_i = (\mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i). \tag{24}$$

This must be different from zero in order for the algorithm to reduce the rank of $\mathbf{A}_{i-1}$ by one. Values of (24) close to or equal to zero indicate that there is nothing more to select in the sense that the two criteria are not able to select components from $\mathbf{A}_{i-1}$. The criteria treated above will be discussed later in this paper.

11. Stopping rules. When this algorithm is used, the basic question concerns when one should stop. The algorithm stops automatically at $i = n$ or $i = k$, whichever is smallest. But usually one should stop earlier. There is, however, no unique answer to this question. It depends on the problem one is trying to solve. When dealing with engineering construction where the coefficients $\mathbf{A} = (a_{ij})$ and $\mathbf{b} = (b_i)$ are uniquely determined, the maximal number of iterations should be carried out. But in most applied sciences these coefficients are uncertain. In these cases it is important to obtain a "stable" solution. There are three aspects we can consider when evaluating the results. They are sizes, predictions, and crossvalidation. We shall consider them closer.

11.1. Sizes. If the criteria that are being used have very small values, it indicates that we have selected all that can be selected. We can also look at the elements of $(\mathbf{A} - \mathbf{A}_i)$ or $(\mathbf{A}\mathbf{x}_i - \mathbf{b})$. If all of them are small, there is no need to continue the iterations. This aspect is discussed in more detail in the study of the (Flow Injection Analysis) data (§15).

11.2. Predictions. It is always good to look at the measures of predictions, both the (total) prediction variance and the (total) prediction error variance. From the point of view of prediction, we should choose the number of components (iterations) where they are at their smallest. Usually the two measures give approximately the same number of components. The H-index is a measure that the H-principle is trying to maximize. It is the ratio {decrease in residual variation}/{increase in model variance}. If the H-index gets small compared to the values of the H-index for the first components, it indicates that the prediction ability is getting small and therefore it is time to stop extracting components.

11.3. Crossvalidation. In this approach some 10 to 15% of data are left out. Initially, these values are estimated by the average values. If only column criterion is used, a_{ij} is estimated by a_j, where a_j is the average value of the jth column. Similarly, if only row criterion is used, we estimate a_{ij} by a_i. If both row and column criteria are used, we estimate a_{ij} by $a_j + a_i - a$, where a is the average over all data values. The decomposition $\mathbf{A}_i$ is computed using these estimated values. The steps in the crossvalidation are as follows:

(1) Delete randomly 10 to 15% of the data values. Compute $\mathbf{A}_i$. Replace the estimated a_{ij}-values by those from $\mathbf{A}_i$. Compute a new $\mathbf{A}_i$. Repeat this procedure until $\mathbf{A}_i$ does not change.

(2) We compute the PRESS value,

$$\text{PRESS} = \Sigma(a_{ij} - \mathbf{A}_i(i, j))^2. \tag{25}$$

The sum in (25) is over the left-out values, and $\mathbf{A}_i(i, j)$ are the corresponding values in $\mathbf{A}_i$.

(3) Repeat (1) and (2) a number of times, e.g., $N = 100$ times.

(4) Compute the average of the N PRESS values for fixed i:

$$\text{PRESS}_i = \Sigma \text{ PRESS/N}. \tag{26}$$

(5) Repeat (1) to (4) for $i = 1, 2, \ldots$. Then we choose the number of components i that gives the smallest value of (26).

The basic idea of crossvalidation is to determine the number of components needed to determine unknown data values. The experience with large numbers of data tables indicates that the number of components may be slightly underestimated by this procedure. Note that if the PRESS value is used to estimate residual variance for given numbers of components, it should be divided by the appropriate number of degrees of freedom.

This is an empirical approach, and it is useful, if the data table is not large. Where there are large data tables, this approach may be time consuming to carry out on the computer.

In the case of linear equations the theoretical formula corresponding to the crossvalidation procedure (25) and (26) is given by (A.2.5) in Appendix 2. If we assume the standard assumptions in linear regression analysis, one may show that (see [4]) requiring (A.2.5) to be as small as possible is equivalent to requiring

$$(\mathbf{b}'\mathbf{u}_i)^2/(\mathbf{u}_i'\mathbf{u}_i) > \sigma^2 n/(n-i), \tag{27}$$

where σ^2 is the residual variance. It can be estimated by

$$\sigma^2 \approx \|\mathbf{A}\mathbf{x}_i - \mathbf{b}\|^2/(n-i).$$

The interpretation of (27) is that, if it is not satisfied, we are generating more noise than the variation we are explaining. Thus (27) represents an upper bound on the number of components that should be used, i.e., if (27) is not satisfied, the component should not be extracted. Experience has shown that (27) is generally better for estimating the number of components than the crossvalidation steps (1)–(5). (There can, however, be problems with this method, as, for example, when the estimate of σ^2 tends to zero because of overfitting and where an estimate of σ^2 must be obtained by crossvalidation.)

12. Some guidelines concerning choice of criteria. The algorithm has been applied to many data from practice. In this section we shall discuss our experiences concerning the choice of criteria. The criteria treated in this paper are as follows:

(a) $\mathbf{w}_i$ is the eigenvector associated with the largest eigenvalue of the eigenvalue problem $\mathbf{A}_{i-1}'\mathbf{A}_{i-1}\mathbf{w}_i = \lambda_i \mathbf{w}_i$.

(b) $\mathbf{w}_i = (0, \ldots, 0, 1, 0, \ldots)$, where the index of 1 is the column number, which maximizes $\max((\mathbf{a}_j'\mathbf{a}_1)^2 + (\mathbf{a}_j'\mathbf{a}_2)^2 + \cdots + (\mathbf{a}_j'\mathbf{a}_k)^2)$.

(c) $\mathbf{w}_i = \mathbf{a}^j$ is an important row in $\mathbf{A}_{i-1}$.

(d) $\mathbf{w}_i = (0, \ldots, 0, 1, 0, \ldots)$, where the index of 1 is the column number of the column, which contains largest element a_{ij} in $\mathbf{A}_{i-1}$ (we can also choose the column that is the largest in size).

(e) $\mathbf{w}_i = (\|\mathbf{a}_1\|, \ldots, \|\mathbf{a}_k\|)$.

(f) $\mathbf{w} = \mathbf{w}_i = (w_1, w_2, \ldots)$ with $w_j = 1$ for $\|\mathbf{a}_j - \mathbf{a}_k\| < c$, and $w_j = c/\|\mathbf{a}_j - \mathbf{a}_k\|$ for $\|\mathbf{a}_j - \mathbf{a}_k\| > c$.

In the case of a linear equation $\mathbf{A}\mathbf{x} = \mathbf{b}$:

(g) $\mathbf{w}_i = \mathbf{A}_{i-1}'\mathbf{b}/\|\mathbf{A}_{i-1}'\mathbf{b}\|$

(h) $\mathbf{w}_i = (0, \ldots, 0, 1, 0, \ldots)$, where the index of 1 is the number of the column giving the maximum of $(\mathbf{b}'\mathbf{a}_j)^2$.

Here we have formulated the criteria as column criteria. Similar criteria can be obtained for rows by changing column and row indices. (Note that $\mathbf{w}_i$ need not be scaled to be of unit length. But we recommend doing so, when $\mathbf{w}_i$ has been selected in order to prevent numerical instability in the algorithm.) One set of the vectors $(\mathbf{w}_i)$ and $(\mathbf{z}_i)$ should be orthogonal. The criteria (a), (b), (c), (d), (g), and (h) all give orthogonal $\mathbf{w}$-vectors. In the case where we are reducing a matrix $\mathbf{A}$, the criteria (a) and (b) are proper ones according to the H-principle, where an optimal balance between residual variance and model variance is determined. Similarly, (g) and (h) are the criteria determined by the H-principle when we are solving a linear equation. If the columns are of approximately equal size, the criteria (b), (g), and (h) are numerically very stable and can be applied to large matrices containing hundreds or thousands of rows and columns. In many cases $\mathbf{w}$ is determined according to (a), (b), (g), or (h), while $\mathbf{z}$ is used to determine the weights that should be used. The weights may be defined from the given applied situation. They may also be derived from one's expectation of data. What happens if $\mathbf{w}_i$ is selected according to (a) and $\mathbf{z}_i$ according to (e)? Well, $(\mathbf{w}_i)$ will be orthogonal and only $\mathbf{u}_1$

will be the same as $\mathbf{u}_1$ in SVD. It means that at each step we select $\mathbf{u}_i$ to be of maximal length, but we weigh the rows, and rows having large length will be more important than small ones. This may be important when we want rows with small values to be scaled down.

It is the author's opinion that the row and column criteria should be selected such that they reflect the views one has on the data.

13. Discussion. The algorithm presented here can be used in many fields of applied sciences. Here we shall discuss some practical aspects of the algorithm.

13.1. Statistics. In the statistical analysis of data the rows are usually considered as repeated samples. In these cases we are mainly interested in the columns. For example, in stepwise regression we determine "good" **w**-vectors. Program packages such as SAS, BMDP, SPSS, and others, contain a large collection of criteria for determining **w**, all of which are different from the choices treated here. These program packages allow a weight vector **z** to take into account differences between rows (samples). This weight vector is independent of the data, and the same at each step (iteration) in our setup. In general it is better to let data themselves determine appropriate weight vectors, in our case the **z**-vectors.

13.2. Biology. Biologists, in their field work, often work with a certain number of species. For example, a biologist may have, say, 51 samples of species. The species are given, but they can be measured in many ways. The main features of the 51 species can be supplemented by different kinds of chemical measurements. New measurements on the 51 species mean that the data table is expanded by new columns representing the new measurements (as opposed to many situations in statistics, where new measurements mean new rows). Here the biologist has certain views on the 51 species that **z**-vectors can represent, while the columns are further information on the given set of species. In these cases the biologist may wish to treat the rows in much the same way that the statistician treats the columns; columns, on the other hand, may be treated as rows. An example of this situation is shown in [14].

13.3. Archaeology and analytic chemistry. In archaeology some excavations are rich in materials while other are scarce. One row of the data table can represent the amounts of different types of materials found at a given location. That the quality of data is different at different locations is often solved by "closing the data table," i.e., scaling the rows to get sum equal to 1 each. The values in a row thus represent the percentage of each type found at that location. In analytical chemistry this approach is also often used. The argument for this scaling of original data is that one is mainly interested in the relative proportions in each sample (row). But this type of scaling often has several complications. In chemistry, for example, some proportions are so small that they are below the detection level. And scaling values to sum to 1 may cause some undesirable side effects, such as overemphasizing samples that are at the noise level. See, e.g., [15]. It will be better to let the data define the weights of samples through the **z**-vectors.

13.4. Neural networks. In some neural networks a_{ij} represents the weight of the link between neuron j to neuron i. In such neural networks the rows of **A** that contain only small values carry very little information. In the analysis such rows should be weighed down, or perhaps not used at all. Sometimes this is solved by defining a threshold value for coefficients in a row, see [16]. But this is not a proper way to handle "small effects," when the phenomenon is continuous. It is better to introduce a weight vector **z** if the data indicate that there is a need for one.

14. Plots. Plots of appropriate vectors are useful in order to learn the variation in data. The most useful plots involving the vectors in the algorithm are the following ones.

(a) $\mathbf{b}$ *versus* $\mathbf{u}_i$. *Added variable plots.* They show how the new component relates to the right-hand-side vector when solving linear equations. The plots should show some degree of linearity. If the scatter plot does not show any kind of linearity, it indicates that the component and later ones should not be used.

(b) $\mathbf{u}_i$ *versus* $\mathbf{u}_j$. *Score plots.* They show how the components vary. They are useful in detecting outliers and groups in data.

(c) $\mathbf{v}_i$ *versus* $\mathbf{v}_j$. *Loading plots.* They show how the variables vary. They are useful in detecting special variables and groups of variables.

(d) $\mathbf{r}_i$ *versus* $\mathbf{r}_j$. *Transformation plots.* They show how the score values are computed. They are useful in detecting the influence of variables and groups of variables.

Reference [17] is a good tutorial paper on the interpretation of plots of the above type.

15. Example 2: FIA data. We shall consider a data table from analytical chemistry. It is a data table derived from flow injection analysis (FIA). FIA is popular in chemical laboratories, because a high degree of automation can be used and because the data collection happens over a relatively short period of time, see [18]. We shall here consider a data table that has 41 rows and 241 columns. One row in the data table is an observed spectrum and the columns are the time points where registration takes place. The contents of the data table are the concentrations of the chemical substance that is being analyzed. The data table considered here has $41 \times 241 = 9881$ data values and all of them are between zero and one. Table 3 gives the ten largest singular values associated with $\mathbf{A}$ that is a 41-times-241 matrix. The numerical

TABLE 3

No. i	Singular value λ_i	Cumulative Percentage $\Sigma^i \lambda_j^2 / \Sigma \lambda_j^2$	Size $\|\mathbf{A}_{i-1}\|_F$	Largest element in $\mathbf{A}_i$, $\max(\|a_{ij}\|)$
1	21.0829379	98.5371	21.2388612	0.142489
2	2.3627405	99.7747	2.5688423	0.034013
3	1.0047351	99.9985	1.0081708	0.006078
4	0.0607094	99.9993	0.0831609	0.003875
5	0.0293583	99.9995	0.0568340	0.003520
6	0.0272765	99.9996	0.0486641	0.002379
7	0.0193580	99.9997	0.0403012	0.002372
8	0.0124964	99.9998	0.0353476	0.001878
9	0.0111743	99.9998	0.0330650	0.001744
10	0.0097538	99.9998	0.0311196	0.001335

precision of the individual data values in an FIA analysis is approximately 0.006. This means that the measured data values are given with an uncertainty of ± 0.003. Thus, from a practical point of view, the rank of $\mathbf{A}$ is at most 10. The size of a matrix is given by

$$\|\mathbf{A}_i\|_F = [\mathrm{tr}(\mathbf{A}_i{}'\mathbf{A}_i)]^{1/2}.$$

Note that $\|\mathbf{A}_o\|_F = 21.2388612$ is the size of the original matrix $\mathbf{A}$. From Table 3 we see that the size of the residual matrix after 10 steps is not zero. This is due to the existence of so many elements (in this case, 9881). Also the singular values do not approach zero as one would expect, $\lambda_{20} = 0.0057$, $\lambda_{30} = 0.0045$, $\lambda_{40} = 0.0031$, and $\lambda_{41} = 0.0030$. The last column in Table 3 gives the largest element in the reduced $\mathbf{A}$-matrix, $\mathbf{A}_i$, when the ith component has been subtracted from $\mathbf{A}_{i-1}$. We see that the last value in that column is well below the numerical precision of the measurements.

The SVD of this data table is not especially interesting from the point of view of the chemist. The important parts of the data table are those points where the concentrations are large. An important question is: "At what time points would it be sufficient to measure the

chemical process? The problem can be phrased as: "What are 'good' columns?" And if we know them, can we compute the remaining columns?" This problem is important, not because we want to limit the number of measurements, but because in FIA analysis we get many tables, and it is difficult to compare the tables when they are so large. The measurement equipment gives us 9881 data values for each experiment, but we want to reduce that data, when storing data and when comparing results from experiments. One way to handle this problem is to select columns according to the criterion (17). The results are shown in Table 4. This table

TABLE 4

No. i	Column number	Criterion (17)	Size $\|\mathbf{A}_{i-1}\|_F$	Largest element $\max(\|a_{ij}\|)$
1	91	3146.692156845	21.238861	0.19597
2	127	2.050383374	3.153601	0.03967
3	73	0.036819414	1.096335	0.00785
4	157	0.000000740	0.099461	0.00729
5	109	0.000000229	0.075478	0.00408
6	47	0.000000047	0.058675	0.00263
7	139	0.000000015	0.050093	0.00261
8	231	0.000000014	0.044776	0.00204
9	61	0.000000003	0.043120	0.00201
10	226	0.000000002	0.039775	0.00167

shows that we can select 10 columns, which can represent the remaining 231 columns with the desired numerical accuracy. We see that the value of criterion (17) is very small for the tenth component compared with the first one. We notice that criterion (17) is already very small at step number 4, where column number 157 is selected. The first impression is that only three components should be selected. If 10 components are selected, the resulting matrix will be 41 times 10. The same procedure can in principle be used on the rows of this new matrix to give a 10-times-10 matrix that is able to reproduce the original data matrix. Similar analysis can be carried out selecting the spectra (rows) instead of columns (time points).

When we look at the data table, large spectra (rows) are more important than the small ones. Here the size of a row is the length of the row vector

$$\|\mathbf{a}^i\| = \left(\sum_j {a_{ij}}^2\right)^{1/2}. \tag{28}$$

In Table 5 we consider both criterion (17) and the maximization of (28). We determine the column that gives maximal value of (17). The coordinates of $\mathbf{w}$ are equal to zeros except for the one that corresponds to the index of the maximum, where it has value one. Similarly, $\mathbf{z}$ has zeros except for the index that gives the maximum value of (28), where it has value 1. The results of the decomposition are given in Table 5. From the table we see that when selecting the first component, the time (column) number 91 has the most predictive power and spectrum (row) number 1 is the largest among the 41 spectra. The last column shows that we can manage with at most 10 components. Note that the size of $\mathbf{A}$ need not decrease, when it is reduced by rank one. In Table 5 the size of $\mathbf{A}$ slightly increases at the fifth, eighth, and tenth components, respectively. This is due to our working with two independent criteria, which when applied may give a slight increase in the size of $\mathbf{A}$, although $\mathbf{A}$ is always reduced by rank one. This happens often when we have reached the noise level in data. It is basically due to the fact that the vectors $(\mathbf{u}_i)$ and $(\mathbf{v}_i)$ are not a set of orthogonal vectors when we have both column and row criteria. Experience with many data tables has shown that we may need more components when we use criteria for both columns and rows than if we use only a criterion for one of them.

TABLE 5

No. i	Column number	Criterion (17)	Row j	Row size $\|\mathbf{a}^j\|$	Size $\|\mathbf{A}_{i-1}\|_F$	Largest element $\max(\|a_{ij}\|)$
1	91	3146.69215684526	1	5.975865	21.23886	0.06715
2	128	26.01563248754	41	1.104906	5.71131	0.08252
3	73	0.40918002568	14	0.520859	2.00683	0.00938
4	155	0.00001107016	19	0.051685	0.17976	0.01542
5	109	0.00063866157	25	0.098970	0.37716	0.00394
6	43	0.00000116260	11	0.028896	0.10209	0.00367
7	163	0.00000008812	18	0.022980	0.06830	0.00338
8	231	0.00000024607	35	0.018013	0.07341	0.00339
9	119	0.00000009667	31	0.015004	0.06579	0.00532
10	108	0.00000066585	5	0.029082	0.08567	0.00282

We shall study more closely the decomposition derived from (17), where the main results were given in Table 4. Some descriptive measures are given in Table 6. We see that the common

TABLE 6

Number	Column number	$\mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i$	$\lambda_i^2\|\mathbf{u}_i\|^2\|\mathbf{v}_i\|^2$	Cumulative %	$\|(\mathbf{A}-\mathbf{A}_i)^\dagger\|_F^2$
1	91	7.1330268	441.14402307	97.7953	0.374423
2	127	0.2345104	8.74325000	99.7335	2.428226
3	73	0.0308873	1.19205798	99.9978	6.459474
4	157	0.0001764	0.00419565	99.9987	89.363820
5	109	0.0001017	0.00225422	99.9992	167.450845
6	47	0.0000502	0.00093339	99.9994	228.700527
7	139	0.0000294	0.00050444	99.9996	329.757384
8	231	0.0000935	0.00014557	99.9996	356.663916
9	61	0.0000097	0.00027722	99.9996	533.998761
10	226	0.0000111	0.00014826	99.9997	640.703899

criterion $\mathbf{z}_i'\mathbf{A}_{i-1}\mathbf{w}_i$ is already small for the fourth component. (Note that this criterion has a similar interpretation to that of the singular values. Since the smallest singular value is 0.0030, we notice that the value of this criterion at the fourth component is smaller than the smallest singular value.) The total variation in $\mathbf{A}$, $\|\mathbf{A}\|_F^2$, is at each step reduced by $\lambda_i^2\|\mathbf{u}_i\|^2\|\mathbf{v}_i\|^2$. The size of this reduction is given in the next column, and the cumulative percentage in the following one. We see that the first component accounts for 97.7953% of the variation in $\mathbf{A}$. Already the first three components account for 99.9978%. When we select the fourth component, the size of the inverse, $\sum^i \lambda_j\mathbf{r}_j\mathbf{s}_j$, increases from 6.459474 to 89.363820. All of these measures indicate that only three components should be used. In Table 7 the predictive measures associated with this decomposition are given. Table 7 confirms that only three components should be extracted from $\mathbf{A}$ because the prediction variances are smallest at the third component. The prediction variances increase 100 times when going from three to four components. Also the H-index, the last column, shows that components number 4 and up have no predictive abilities. These results conform well to the chemical knowledge of data.

As mentioned above, we could have chosen the corresponding row (spectra) criterion and selected the three spectra that can represent the remaining 38 spectra and carried out a similar analysis.

A similar analysis, like that shown in Tables 6 and 7, can be carried out for the decompositions shown in Tables 3 and 5. For the decomposition in Table 5 we get results analogous to the ones in Table 6 and 7.

For a chemist there is much more information in Tables 4 and 5 than in Table 3, because the chemist has a more direct interpretation of Tables 4 and 5 than of Table 3. But everything

TABLE 7

Number	Column number	$\|\mathbf{A}_i\|^2\|(\mathbf{A}-\mathbf{A}_i)^\dagger\|^2$	$\|\mathbf{A}_i\|^2(1+\|(\mathbf{A}-\mathbf{A}_i)^\dagger\|^2)$	$\|\Delta\mathbf{A}_i\|_F{}^2/\|\Delta\mathbf{A}_i^\dagger\|_F{}^2$
1	91	1.3942469	11.3394475	3146.69215684526
2	127	7.0870388	8.2889894	1.51895682315
3	73	0.4127659	0.4226585	0.03327120666
4	157	45.495088	45.5007851	0.00000052814
5	109	96.532847	96.5362896	0.00000011241
6	47	131.247229	131.2497385	0.00000003847
7	139	218.009909	218.0119134	0.00000000894
8	231	236.520152	236.5220116	0.00000000788
9	61	451.138113	451.1396951	0.00000000176
10	226	588.585720	588.5871538	0.00000000118

can be used to represent the original matrix or to compute a generalized inverse. The SVD is popular, because one gets maximal reduction in the variation of $\mathbf{A}$ at each step. But as Table 4 shows, the decomposition according to the criterion (17) gives almost the same size of reduction. The general experience is that decompositions using (17) require only very few more components than does the SVD to get the same size of reduction in variation.

If all computations in this example were carried out in one run on a computer, it would take approximately 300 Mflops. Thus on a modern workstation or PC these computations can be carried out in seconds or minutes, depending on the size of the computer.

16. Conclusions. We have presented here an approach for solving standard matrix problems such as determining the rank of a matrix, solving linear equations, and computing the generalized inverse. This methodology allows "tailoring" a mathematical model to the practical situation. The approach is given in terms of a general algorithm to decompose a data matrix into rank-one components. The algorithm can reflect different views on rows, on columns, or on both rows and columns. If the criteria used are based on the H-principle of mathematical modelling and data are initially appropriately scaled, the algorithm has been found efficient and numerically stable even for very large matrices containing hundreds or thousands of rows and columns. The algorithm is easy to implement in matrix languages such as MATLAB, GLIM, MATEMATICA, SASIML, and other similar languages. The main advantage of this approach is that the noise level can be appropriately identified. The algorithm is important for scientists who want more nuanced decompositions of data tables that reflect their theories on data than those provided by standard numerical methods.

Appendix 1. Here we shall prove that the vectors have the geometric properties stated in the text.

PROPERTY 1. $\mathbf{z}_i$ *is orthogonal to later* $\mathbf{A}$*-matrices,*

$$\mathbf{z}_i'\mathbf{A}_j = \mathbf{0} \quad \textit{for } i \le j. \tag{A.1.1}$$

Proof. Expressing $\mathbf{A}_j$ in terms of earlier $\mathbf{A}$-matrices we get

$$\begin{aligned}\mathbf{A}_j &= \mathbf{A}_{j-1} - \lambda_j \mathbf{u}_j \mathbf{v}_j' \\ &= \mathbf{A}_{j-1} - \lambda_j \mathbf{A}_{j-1}\mathbf{w}_j\mathbf{v}_j' \\ &= \mathbf{A}_{j-1}(\mathbf{I} - \lambda_j \mathbf{w}_j\mathbf{v}_j') \\ &= \mathbf{A}_{j-1}\mathbf{Q}_o \\ &= \mathbf{A}_i\mathbf{Q}_1.\end{aligned}$$

Here $\mathbf{Q}_o$ and $\mathbf{Q}_1$ are matrices that are not used. From

$$\mathbf{z}_i'\mathbf{u}_i = 1/\lambda_i \quad \text{and} \quad \mathbf{z}_i'\mathbf{A}_i = \mathbf{z}_i'\mathbf{A}_{i-1} - \lambda_i(\mathbf{z}_i'\mathbf{u}_i)\mathbf{v}_i' = \mathbf{v}_i' - \mathbf{v}_i' = \mathbf{0}$$

(A.1.1) follows. □

PROPERTY 2. $\mathbf{z}_i$ *is orthogonal to later* $\mathbf{u}$*-vectors,*

$$\mathbf{z}_i'\mathbf{u}_j = 0, \qquad i < j. \tag{A.1.2}$$

Proof. Follows from (A.1.1) and the definition of $\mathbf{u}_j$. □

PROPERTY 3. $\mathbf{w}_i$ *is orthogonal to later* $\mathbf{A}$*-matrices,*

$$\mathbf{A}_j\mathbf{w}_i = \mathbf{0}, \qquad i \leq j. \tag{A.1.3}$$

Proof. The iteration is performed in a way similar to the way in which the proof of Property 1 was performed:

$$\begin{aligned}\mathbf{A}_j &= \mathbf{A}_{j-1} - \lambda_j\mathbf{u}_j\mathbf{z}_j'\mathbf{A}_{j-1} = (\mathbf{I} - \lambda_j\mathbf{u}_j\mathbf{z}_j)\mathbf{A}_{j-1} \\ &= \mathbf{Q}\mathbf{A}_i,\end{aligned}$$

where $\mathbf{Q}$ is some matrix that is not used. Equation (A.1.3) follows now from

$$\mathbf{w}_i'\mathbf{v}_i = 1/\lambda_i \quad \text{and} \quad \mathbf{A}_i\mathbf{w}_i = \mathbf{A}_{i-1}\mathbf{w}_i - \lambda_i\mathbf{u}_i(\mathbf{w}_i'\mathbf{v}_i) = \mathbf{u}_i - \mathbf{u}_i = \mathbf{0}. \quad \square$$

PROPERTY 4. $\mathbf{w}_i$ *is orthogonal to later* $\mathbf{v}$*-vectors,*

$$\mathbf{w}_i'\mathbf{v}_j = 0, \qquad i < j. \tag{A.1.4}$$

Proof. The proof follows from (A.1.3) and the definition of $\mathbf{v}_j$. □

PROPERTY 5. $(\mathbf{s}_i)$ *and* $(\mathbf{u}_i)$ *are mutually orthogonal,*

$$\mathbf{s}_i'\mathbf{u}_j = 0 \quad \textit{and} \quad \mathbf{s}_i'\mathbf{u}_i = 1/\lambda_i, \quad i \neq j. \tag{A.1.5}$$

Proof. Consider $i = j = 1$. We get

$$\begin{aligned}\mathbf{u}_1 &= \mathbf{A}\mathbf{w}_1 = \mathbf{A}\mathbf{r}_1 \quad \text{or} \quad \mathbf{r}_1 = \mathbf{w}_1. \\ \mathbf{v}_1 &= \mathbf{A}'\mathbf{z}_1 = \mathbf{A}'\mathbf{s}_1 \quad \text{or} \quad \mathbf{s}_1 = \mathbf{z}_1.\end{aligned}$$

This gives

$$\mathbf{s}_1'\mathbf{u}_1 = \mathbf{z}_1'\mathbf{A}\mathbf{w}_1 = 1/\lambda_1.$$

For $i = 2$ and $j = 1$, we get

$$\mathbf{v}_2 = \mathbf{A}_1'\mathbf{z}_2 = \mathbf{A}'(\mathbf{I} - \lambda_1\mathbf{z}_1\mathbf{u}_1')\mathbf{z}_2 \quad \text{or} \quad \mathbf{s}_2 = (\mathbf{I} - \lambda_1\mathbf{z}_1\mathbf{u}_1')\mathbf{z}_2.$$

This gives

$$\mathbf{u}_1'\mathbf{s}_2 = (\mathbf{u}_1' - \lambda_1(\mathbf{u}_1'\mathbf{z}_1)\mathbf{u}_1')\mathbf{z}_2 = (\mathbf{u}_1' - \mathbf{u}_1')\mathbf{z}_2 = 0.$$

For $i = j = 2$ we get from Property 1,

$$\mathbf{u}_2'\mathbf{s}_2 = \mathbf{w}_2'\mathbf{A}_1'\mathbf{s}_2 = \mathbf{w}_2\mathbf{A}_1'(\mathbf{I} - \lambda_1\mathbf{z}_1\mathbf{u}_1')\mathbf{z}_2 = \mathbf{w}_2'\mathbf{A}_1'\mathbf{z}_2 = 1/\lambda_2.$$

Higher indices are proved along the lines of the proofs of Properties 1 and 3. □

PROPERTY 6. $(\mathbf{r}_i)$ *and* $(\mathbf{v}_i)$ *are mutually orthogonal,*

$$\mathbf{r}_i'\mathbf{v}_j = 0, \quad \mathbf{r}_i'\mathbf{v}_i = 1/\lambda_i \quad \textit{for } i \neq j. \tag{A.1.6}$$

Proof. The proof is shown in a way similar to that of Property 5. □

PROPERTY 7. *If* $(\mathbf{w}_i)$ *is chosen such that it is in the range of* $\mathbf{A}_{i-1}'$, $(\mathbf{w}_i)$ *are mutually orthogonal,*

$$\mathbf{w}_i'\mathbf{w}_j = 0 \quad \textit{for } i \neq j.$$

Proof. Let $i < j$. That $\mathbf{w}_j$ is in the range of $\mathbf{A}_{j-1}'$ means that there is a vector $\mathbf{q}_j$ such that $\mathbf{w}_j = \mathbf{A}_{j-1}'\mathbf{q}_j$. This gives

$$\mathbf{w}_i'\mathbf{w}_j = (\mathbf{A}_{j-1}\mathbf{w}_i)'\mathbf{q}_j = 0,$$

by Property 3. □

PROPERTY 8. *The generalized inverse of* $\mathbf{A}$ *can be computed as*

$$\begin{aligned}\mathbf{A}^\dagger &= \lambda_1\mathbf{r}_1\mathbf{s}_1' + \lambda_2\mathbf{r}_2\mathbf{s}_2' + \cdots \\ &= \mathbf{R}\Lambda\mathbf{S}'.\end{aligned} \tag{A.1.7}$$

Proof. The equation $\mathbf{A}\mathbf{A}^\dagger\mathbf{A} = \mathbf{A}$ follows from Properties 5 and 6. □

Note that if only column criterion is given, the vectors $(\mathbf{u}_i)$ will be orthogonal, and similarly if we have only a row criterion, $(\mathbf{v}_i)$ will be orthogonal.

Appendix 2. Predictive performance measures. In this section we shall consider more closely the performance measures suggested in this paper. The set of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ suggests that there is given a linear model,

$$b = a_1x_1 + a_2x_2 + \cdots + a_kx_k = \mathbf{a}'\mathbf{x}.$$

Suppose that $\mathbf{a}_o = (a_{o1}, a_{o2}, \ldots, a_{ok})$ is a new row. We estimate the new b-value, b_o, by inserting this new vector in the equation, giving

$$b_o = a_{o1}x_1 + a_{o2}x_2 + \cdots + a_{ok}x_k = \mathbf{a}_o'\mathbf{x}.$$

The important question is now: "What variation can be expected for b_o, when the coefficients $\mathbf{x}$ are uncertain?" The answer is given in standard textbooks in statistics, see, e.g., [19]. Assuming the usual assumptions in linear regression analysis, it follows that the variance of b_o can be computed as

$$\text{Var}\,(b_o) = \sigma^2\mathbf{a}_o'(\mathbf{A}'\mathbf{A})^\dagger\mathbf{a}_o \approx \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2\mathbf{a}_o'(\mathbf{A}'\mathbf{A})^\dagger\mathbf{a}_o/(n-k). \tag{A.2.1}$$

We see that the variance consists of two parts, the residual variation $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$, and the model variance $\mathbf{a}_o'(\mathbf{A}'\mathbf{A})^\dagger\mathbf{a}_o$. The model variance is due to the uncertainty of the vector $\mathbf{x}$. The model variance depends on the specific $\mathbf{a}_o$ that is used. But when we compare one situation to another,

$\text{tr}((\mathbf{A}'\mathbf{A})^\dagger)$ gives a fairly good impression on the size of the model variance. Therefore, when we compare solution vectors, we compare what we call the *total prediction variance*:

$$\|\mathbf{Ax} - \mathbf{b}\|^2 \text{tr}((\mathbf{A}'\mathbf{A})^\dagger) \approx \|\mathbf{Ax} - \mathbf{b}\|^2 \|\mathbf{A}^\dagger\|_F{}^2. \tag{A.2.2}$$

In the present algorithm, $\mathbf{A}_i$ is the residual matrix and $(\mathbf{A} - \mathbf{A}_i)$ is the extracted part. With this notation (A.2.2) becomes

$$\|\mathbf{Ax}_i - \mathbf{b}\|^2 \|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2. \tag{A.2.3}$$

In case we are only reducing $\mathbf{A}$, a similar line of argument suggests that we consider

$$\|\mathbf{A}_i\|_F{}^2 \|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2 \tag{A.2.4}$$

to be an analogous measure of total prediction variance.

If we have a value of the right-hand-side, $\mathbf{b}$, we may want to compare b and b_o. Usually b is what is observed and b_o is the computed value according to the model. One may similarly show that the variance of the difference is given by

$$\text{Var}\,(b - b_o) = \sigma^2(1 + \mathbf{a}_o'(\mathbf{A}'\mathbf{A})^\dagger \mathbf{a}_o) \approx \|\mathbf{Ax} - \mathbf{b}\|^2 (1 + \mathbf{a}_o'(\mathbf{A}'\mathbf{A})^\dagger \mathbf{a}_o)/(n - k). \tag{A.2.5}$$

As in (A.2.3), we determine the *total prediction error variance* to be

$$\|\mathbf{Ax} - \mathbf{b}\|^2 (1 + \|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2) \tag{A.2.6}$$

in the case of reducing $\mathbf{A}$, it is analogous to (A.2.4):

$$\|\mathbf{A}_i\|_F{}^2 (1 + \|(\mathbf{A} - \mathbf{A}_i)^\dagger\|_F{}^2). \tag{A.2.7}$$

For further details on (A.2.1) and (A.2.5) see [4].

The H-principle is concerned with obtaining a balance between the improvement in fit, $\boldsymbol{\Delta}(\mathbf{Ax}_i - \mathbf{b}) = \boldsymbol{\Delta}Ax_i$, and the increase in model variance $\boldsymbol{\Delta}(\mathbf{A}_i{}^\dagger)$. The H-principle tries to get at each step of the algorithm as large a value as possible for the H-index, where the H-index for linear equations is

$$\|\boldsymbol{\Delta}(\mathbf{Ax}_i)\|^2 / \|\boldsymbol{\Delta}\mathbf{A}_i{}^\dagger\|_F{}^2$$

and for reducing $\mathbf{A}$ it is

$$\|\boldsymbol{\Delta}\mathbf{A}_i\|_F{}^2 / \|\boldsymbol{\Delta}\mathbf{A}_i{}^\dagger\|_F{}^2.$$

For a more precise treatment of the H-principle, see [4]. We compute these measures from

$$\begin{aligned}
\boldsymbol{\Delta}Ax_i &= \lambda_i(\mathbf{b}'\mathbf{s}_i)\mathbf{r}_i, & \|\boldsymbol{\Delta}(\mathbf{Ax}_i)\|^2 &= \lambda_i{}^2(\mathbf{b}'\mathbf{s}_i)^2(\mathbf{r}_i'\mathbf{r}_i).\\
\boldsymbol{\Delta}A_i &= \lambda_i \mathbf{u}_i \mathbf{v}_i', & \|\boldsymbol{\Delta}\mathbf{A}_i\|_F{}^2 &= \lambda_i{}^2(\mathbf{u}_i'\mathbf{u}_i)(\mathbf{v}_i'\mathbf{v}_i).\\
\boldsymbol{\Delta}A_i^\dagger &= \lambda_i \mathbf{r}_i \mathbf{s}_i', & \|\boldsymbol{\Delta}\mathbf{A}_i{}^\dagger\|_F{}^2 &= \lambda_i{}^2(\mathbf{r}_i'\mathbf{r}_i)(\mathbf{s}_i'\mathbf{s}_i).
\end{aligned}$$

Acknowledgment. The author wishes to thank Carsten Ridder, Chemical Laboratory A, Danish Technical University, for supplying small, medium, and large data matrices as well as many useful discussions.

REFERENCES

[1] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1989.

[2] J. E. JACKSON, *Users Guide to Principal Components*, Wiley, New York, 1991.

[3] I. T. JOLLIFFE, *Principal Component Analysis*, Springer-Verlag, Berlin, 1986.

[4] A. HÖSKULDSSON, *The H-principle in modelling with applications to chemometrics*, Chemometr. Intell. Lab. Systems, 14 (1992), pp. 139–153.

[5] W. WINDIG, C. E. HECKLER, F. A. AGBLEVOR, AND R. J. EVANS, *Self-modelling mixture analysis of categorized pyrolysis mass spectra data with the SIMPLISMA approach*, Chemometr. Intell. Lab. Systems, 14 (1992), pp. 195–207.

[6] O. KVALHEIM, *Model building in chemistry, a unified approach*, Analytica Chemica Acta, 223 (1989), pp. 53–73.

[7] M. HESTENES, *Conjugate Direction Methods in Optimization*, Springer-Verlag, Berlin, 1980.

[8] R. R. HOCKING, *The analysis and selection of variables in linear regression*, Biometrics, 32 (1976), pp. 1–49.

[9] M. L. THOMPSON, *Selection of variables in multiple regression: Part* I. *Review and evaluation*, Internat. Statist. Rev., 46 (1978), pp. 1–19.

[10] ———, *Part* II. *Chosen procedures, computations and examples*, Internat. Statist. Rev., 46 (1978), pp. 129–146.

[11] P. R. KRISHNAIAH, *Selection of variables under univariate regression models*, *Handbook of Statistics*, Vol. 2, Krishnaiah and Kanal, eds., North Holland, Amsterdam, 1982, pp. 805–820.

[12] A. HÖSKULDSSON, PLS *Regression Methods*, J. Chemometrics, 2 (1988), pp. 211–228.

[13] K. J. ÅSTRÖM AND B. WITTENMARK, *Adaptive Control*, Addison-Wesley, New York, 1989.

[14] J. C. FRISVAD, *Chemometrics and chemotaxanomy: A comparison of multivariate statistical methods for the evaluation of binary fungal secondary metabolite data*, Chemometr. Intell. Lab. Systems, 14 (1992), pp. 253–269.

[15] H. AITCHISON, *The statistical analysis of geochemical compositions*, Math. Geo., 16 (1984), pp. 531–564.

[16] Y. KAMP, *Recursive Neural Networks for Associative Memory*, Wiley, New York, 1990.

[17] O. KVALHEIM AND T. V. KARSTANG, *Interpretation of latent-variable regression models*, Chemometr. Intell. Lab. Systems, 7 (1989), pp. 39–51.

[18] J. RUZICKA AND E. H. HANSEN, *Flow Injection Analysis*, 2nd ed., Wiley, New York, 1988.

[19] K. V. MARDIA, J. T. KENT, AND J. M. BIBBY, *Multivariate Analysis*, Academic Press, New York, 1979.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 263–284, March 1994

002

A HIGHER-ORDER GODUNOV METHOD FOR MULTIDIMENSIONAL IDEAL MAGNETOHYDRODYNAMICS*

ANDREW L. ZACHARY†, ANDREA MALAGOLI‡, AND PHILLIP COLELLA§

Abstract. The authors present a higher-order Godunov method for the solution of the two- and three-dimensional equations of ideal magnetohydrodynamics (MHD). This work is based both on a suitable operator-split approximation to the full multidimensional equations, and on a one-dimensional Riemann solver. This Riemann solver is sufficiently robust to handle the nonstrictly hyperbolic nature of the MHD equations and the presence of local linear degeneracies. Results from a set of test problems show that this operator-split methodology has no problems handling any of the three MHD waves, yet resolves shocks to three or four computational zones. The advantages and limitations of this method are discussed.

Key words. magnetohydrodynamics (MHD), Godunov methods, hyperbolic systems, finite difference equations

AMS subject classifications. 65M06, 35L665, 3504, 76W05

1. Introduction.

Conservative, finite-difference schemes based on higher-order Godunov methods have proven very effective at computing discontinuous solutions to hyperbolic systems of conservation laws. Several examples of such schemes are available for the equations of hydrodynamics (van Leer [23], Roe [20], Harten, Lax, and van Leer [13], and Colella and Woodward [9]) and have been used extensively to simulate highly supersonic flows in aerodynamics and astrophysics. For a comparative review of some of these methods, see, e.g., Woodward and Colella [24]. An extension of Godunov methods to general systems of hyperbolic conservation laws has been suggested by Bell, Colella, and Tragenstein [3], hereinafter refered to as BCT, who have also discussed modifications to the basic method when the systems have points at which they are no longer strictly hyperbolic. At these locations, some of the eigenvalues of the linearized matrix are identical and the corresponding eigenvectors become degenerate.

In this paper, we present a new, multidimensional scheme for the equations of ideal MHD, which is based on a higher-order Godunov method specially developed to treat the degeneracies that occur in these equations. Our formulation is a direct extension of the method developed by Zachary and Colella [25], but with several modifications and amendments that have proven necessary when testing the algorithm on a variety of one- and two-dimensional problems. The core of the problem is to construct an appropriate solution of a local, linearized Riemann problem along the ray $\frac{x}{t} = 0$ at the boundary between two adjacent cell edges. This solution is then used to compute the flux $\mathbf{F}^G(\mathbf{U}_R, \mathbf{U}_L)$, the generalized Engquist–Osher [12] flux for our system of conservation laws (see BCT [3] for details).

Although this approach has several computational advantages over finding the solution to the complete nonlinear Riemann problem, it does have two major difficulties that require special attention. First, the equations of ideal MHD involve seven characteristics that can become mutually degenerate in several ways. Each type of degeneracy requires a different,

*Received by the editors June 22, 1992; accepted for publication (in revised form) June 24, 1993.

†Cray Research, Inc., 397 Post Road, Darien, Connecticut 06820. The work of this author was supported by Cray Research Inc.

‡Laboratory for Astrophysics and Space Research (LASR), University of Chicago, 933 E. 56th St., Chicago, Illinois 60637 (malagoli@mhd4.uchicago.edu). The work of this author was partly supported in part by National Aeronautics and Space Administration grant NAG 5-1485.

§Department of Mechanical Engineering, University of California, Berkeley, California 94720 (colella@barkley.Berkeley.edu). The work of this author was partly supported at the University of California at Berkeley by Army Research Office grant DAALO3-88-K-0197, Defense Advanced Research Projects Agency and National Science Foundation grant DMS-8919074, and National Science Foundation Presidential Young Investigator award grant ACS-8958522.

special treatment. Second, linearized Riemann solvers are known to fail in some extreme situations, even if the underlying physical problem admits a solution. Einfeldt et al. [11] have shown, for example, that certain types of hydrodynamic Riemann solvers fail to compute strong rarefaction waves correctly. Based on the experience obtained by computing a large set of test problems, we have incorporated an algorithm in our Riemann solver that detects and classifies the occurrence of degeneracies and strong rarefactions. This Riemann solver is so sufficiently robust that we have used it as the basis of a multidimensional scheme.

Following common procedures for higher-order methods, e.g., the piecewise parabolic method (PPM) and MUSCL schemes, the most natural way to extend our Riemann solver to two or three dimensions is by operator-splitting, in which a fully multidimensional step is achieved by combining a sequence of one-dimensional steps in alternating directions. However, the use of operator-splitting for the MHD equations cannot be achieved in a straightforward way, because the condition $\nabla \cdot \mathbf{B} = 0$ introduces cross-coupling between the spatial directions. In this paper we have still adopted an operator-split scheme, because of its relative simplicity, and we have written the MHD equations in a particular nonconservation form that makes them suitable for operator-splitting. The condition $\nabla \cdot \mathbf{B} = 0$ is further enforced by performing a projection (see, e.g., Brackbill and Barnes [4]) at the end of each timestep. A fully unsplit, multidimensional method will be developed at a later stage following the ideas of Colella [8].

In §2, we present an extensive description of the one-dimensional, higher-order Godunov method. We also briefly discuss our particular form of the operator-split MHD equations, and then we provide a detailed look at the Riemann solver. Finally, in §3, we present and discuss the results from our extensive series of one- and two-dimensional test problems.

2. The operator-split higher-order Godunov method. Zachary and Colella [25] describe a higher-order Godunov method designed for purely one-dimensional MHD equations. As with similar higher-order methods, e.g., PPM, MUSCL, BCT, this method has two parts: a predictor step that traces characteristics to determine time-centered values of the solution extrapolated to each cell edge from each of the left and right cell centers; and, a corrector step in which fluxes are computed at the cell edge and differenced to update the conserved quantities at the cell centers. These fluxes are computed using an approximate Riemann solver, using as input the left and right states obtained in the predictor step. In an earlier paper, we showed that our Riemann solver was sufficiently robust to handle the wave interactions present in a set of one-dimensional test problems. When these techniques are extended to a fully multidimensional set of MHD equations, they must handle a much wider range of wave phenomena. As a consequence, we have been led to make a number of modifications in our earlier solver to deal with these new interactions. Specifically, we must: account for terms of the form $\partial B_x/\partial x$ that arise from the operator-splitting; handle degeneracies that commonly arise in multidimensional studies, but which rarely occur in purely one-dimensional test problems; refine the treatment of rarefaction waves; and, finally, modify the predictor step to preserve monotonic gradients in both the characteristic fields and in the primitive variables. Since this solver differs significantly from the version described earlier, we present a detailed description of it below.

Before we begin our discussion, we make one general comment about our method. Many finite-difference algorithms such as MacCormack's method, Lax–Wendroff, and flux-corrected transport work only with $\mathbf{U}$, the conserved variables. Our experience with a higher-order Godunov method has led us to work with $\mathbf{U}$, as well as with the primitive variables $\mathbf{W} = [\rho, u_x, u_y, u_z, B_x, B_y, B_z, p]^t$. There are several reasons for our choice. First, working in $\mathbf{W}$-space provides the simplest possible expressions for the right and left eigenvectors. Using these representations reduced the complexity and improved the performance of the resulting computer code. Second, our algorithm must preserve positivity in the density and the pressure. When the predictor or the corrector step used conserved variables, we found there was a broad

range of conditions under which our Riemann solver produced a negative pressure. As our section below on rarefaction waves makes clear, switching to primitive variables fixed most, but not all, of the cases under which either the pressure or the density became negative.

2.1. Operator-split equations of MHD. The full set of equations for multidimensional hydrodynamics naturally splits into a set of one-dimensional equations with no cross-coupling. Unfortunately, this statement is not true for the equations of MHD; the restriction $\nabla\cdot\mathbf{B} = 0$ naturally induces some cross-coupling between the different directions. In this section, we present a set of one-dimensional equations for MHD that is suitable for operator-splitting. We begin with the full set of equations written as:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\mathbf{u}) = 0, \tag{1a}$$

$$\frac{\partial \rho u_x}{\partial t} + \frac{\partial}{\partial x}\left(\rho u_x^2 + p + \frac{B_{\perp,x}^2}{8\pi}\right) + \nabla\cdot\left(\mathbf{u}_{\perp,x} u_x \rho\right) - \frac{1}{4\pi}\left(\mathbf{B}_{\perp,x}\cdot\nabla\right) B_x = 0, \tag{1b}$$

$$\frac{\partial \rho u_y}{\partial t} + \frac{\partial}{\partial y}\left(\rho u_y^2 + p + \frac{B_{\perp,y}^2}{8\pi}\right) + \nabla\cdot\left(\mathbf{u}_{\perp,y} u_y \rho\right) - \frac{1}{4\pi}\left(\mathbf{B}_{\perp,y}\cdot\nabla\right) B_y = 0, \tag{1c}$$

$$\frac{\partial \rho u_z}{\partial t} + \frac{\partial}{\partial z}\left(\rho u_z^2 + p + \frac{B_{\perp,z}^2}{8\pi}\right) + \nabla\cdot\left(\mathbf{u}_{\perp,z} u_z \rho\right) - \frac{1}{4\pi}\left(\mathbf{B}_{\perp,z}\cdot\nabla\right) B_z = 0, \tag{1d}$$

$$\frac{\partial \mathbf{B}}{\partial t} + \nabla\cdot(\mathbf{u}\mathbf{B} - \mathbf{B}\mathbf{u}) = 0, \tag{1e}$$

$$\frac{\partial(\rho E)}{\partial t} + \nabla\cdot(\rho\mathbf{u}E + \mathbf{u}p) + \sum_{i=1}^{3}\frac{\partial}{\partial x}\left(u_{x_i}\frac{B_{\perp,x_i}^2}{8\pi}\right) - \frac{1}{4\pi}\sum_{i=1}^{3}\frac{\partial}{\partial x_i} B_{x_i}\left(\mathbf{B}_{\perp,x_i}\cdot\mathbf{u}\right) = 0, \tag{1f}$$

$$\nabla\cdot\mathbf{B} = 0. \tag{1g}$$

The right-hand side of these equations differs from the conservation form of the equations by terms of the form $\nabla\cdot\mathbf{B}(\ldots)$, where the terms inside the parentheses are not differentiated. Our reason for using this form was to minimize the extent to which various terms in the discrete evolution operators in each of the coordinate directions in the operator split algorithm would have to sum to zero due to the divergence-free constraint on the magnetic field. The use of this nonconservation form is similar to the common practice, in incompressible flow calculations, of using advective differencing of the velocity fields, rather than conservative differencing (see, for example, Bell, Colella, and Glaz [1]). Also, it has been shown by Brackbill and Barnes [4] that using the nonconservation form of the momentum equations reduces the effect of magnetic monopole forces on the dynamics of the system. In the above equations, we have used the following notation:

$$\rho E = \frac{1}{2}\rho u^2 + \frac{p}{\gamma - 1} + \frac{B^2}{8\pi},$$

$$\mathbf{B}_{\perp,x_i} = \mathbf{B} - B_{x_i}\hat{\mathbf{e}}_i,$$

$$\mathbf{u}_{\perp,x_i} = \mathbf{u} - u_{x_i}\hat{\mathbf{e}}_i,$$

$$\nabla\cdot\left(\mathbf{q}_{\perp,x_i} a\right) = \sum_{i\neq j}\frac{\partial}{\partial x_j}(q_{x_j} a) = \nabla_{\perp,x_i}(\mathbf{q}a),$$

$$\mathbf{q}_{\perp,x_i}\cdot\nabla a = \sum_{i\neq j} q_j\frac{\partial a}{\partial x_j} = \left(\mathbf{q}\cdot\nabla_{\perp,x_i}\right) a.$$

With the equations written in this form, it is immediately apparent that a nonconservative, one-dimensional, operator-split set will be

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_x}{\partial x} = 0, \tag{2a}$$

$$\frac{\partial \rho u_x}{\partial t} + \frac{\partial}{\partial x}\left(\rho u_x^2 + p + \frac{B_{\perp,x}^2}{8\pi}\right) = 0, \tag{2b}$$

$$\frac{\partial \rho u_y}{\partial t} + \frac{\partial}{\partial x}\left(u_x u_y \rho\right) - \frac{B_x}{4\pi}\frac{\partial B_y}{\partial x} = 0, \tag{2c}$$

$$\frac{\partial \rho u_z}{\partial t} + \frac{\partial}{\partial x}\left(u_x u_z \rho\right) - \frac{B_x}{4\pi}\frac{\partial B_z}{\partial x} = 0, \tag{2d}$$

$$\frac{\partial B_x}{\partial t} = 0, \tag{2e}$$

$$\frac{\partial B_y}{\partial t} + \frac{\partial}{\partial x}\left(u_x B_y - u_y B_x\right) = 0, \tag{2f}$$

$$\frac{\partial B_z}{\partial t} + \frac{\partial}{\partial x}\left(u_x B_z - u_z B_x\right) = 0, \tag{2g}$$

$$\frac{\partial (\rho E^{1D})}{\partial t} + \frac{\partial}{\partial x}\left(\rho u_x E^{1D} + u_x p + u_x \frac{B_{\perp,x}^2}{8\pi}\right) - \frac{1}{4\pi}\frac{\partial}{\partial x} B_x \left(B_y u_y + B_z u_z\right) = 0. \tag{2h}$$

Note that we have not completely eliminated terms of the form $\partial B_x/\partial x$; they appear in the B_y, B_z, and ρE evolution equations. However, since $\partial B_x/\partial t = 0$, we can formally treat them as source terms.

The one-dimensional equations are all of the form

$$\frac{\partial \mathbf{U}^{1D}}{\partial t} + \frac{\partial \mathbf{F}^{1D}}{\partial x} + \mathbf{A}(\mathbf{U}^{1D})\frac{\partial \mathbf{H}(\mathbf{U}^{1D})}{\partial x} = 0, \tag{3}$$

where $\mathbf{U}^{1D} = [\rho, \rho u_x, \rho u_y, \rho u_z, B_x, B_y, B_z, \rho E^{1D}]^T$. On a one-dimensional row of cells, our discretization of these equations has the following form:

$$\begin{aligned} \mathbf{U}_i^{n+1} = \mathbf{U}_i^n &+ \frac{\Delta t}{\Delta x}\left(\mathbf{F}\left(\mathbf{U}_{i-1/2}^{n+1/2}\right) - \mathbf{F}\left(\mathbf{U}_{i+1/2}^{n+1/2}\right)\right) \\ &+ \frac{\Delta t}{\Delta x}\mathbf{A}\left(\frac{1}{2}\left(\mathbf{U}_{i+1/2}^{n+1/2} + \mathbf{U}_{i-1/2}^{n+1/2}\right)\right)\cdot\left(\mathbf{H}\left(\mathbf{U}_{i-1/2}^{n+1/2}\right)\mathbf{H}\left(\mathbf{U}_{i+1/2}^{n+1/2}\right)\right), \end{aligned} \tag{4}$$

where by $\mathbf{U}$ we mean $\mathbf{U}^{1D}$. By adding the contribution of B_x^2 back into the energy, we arrive at an evolution operator for the conserved quantities, $\mathbf{U}$. We denote this evolution operator by $(\mathbf{L}_x^{\Delta t}\mathbf{U}^n)_i - \mathbf{U}_i^{n+1}$. Similarly, we can define operators $\mathbf{L}_y^{\Delta t}$ and $\mathbf{L}_z^{\Delta t}$ for the one-dimensional equations in the y and z directions.

We can extend these one-dimensional operators to act on the full two- or three-dimensional grid of data,

$$\begin{aligned} \left(\mathcal{L}_x^{\Delta t}\mathbf{U}\right)_{i,j,k} &= \left(\mathbf{L}_x^{\Delta t}\mathbf{U}_{\cdot,j,k}\right)_i, \\ \left(\mathcal{L}_y^{\Delta t}\mathbf{U}\right)_{i,j,k} &= \left(\mathbf{L}_y^{\Delta t}\mathbf{U}_{i,\cdot,k}\right)_j, \\ \left(\mathcal{L}_z^{\Delta t}\mathbf{U}\right)_{i,j,k} &= \left(\mathbf{L}_z^{\Delta t}\mathbf{U}_{i,j,\cdot}\right)_k. \end{aligned} \tag{5}$$

Our full operator-split scheme is given by

$$\mathbf{U}_{i,j,k}^{n+1} = \left(\mathcal{L}_x^{\Delta t}\mathcal{L}_y^{\Delta t}\mathcal{L}_z^{\Delta t}\mathcal{L}_z^{\Delta t}\mathcal{L}_y^{\Delta t}\mathcal{L}_x^{\Delta t}\mathbf{U}^n\right)_{i,j,k}, \tag{6}$$

where we have used symmetric Strang splitting to obtain an overall evolution that is second-order accurate in time provided the individual one-dimensional operators are second-order accurate.

2.2. The predictor step: tracing characteristics. As in other higher-order Godunov methods, we predict states at the zone interface by tracing characteristics. To trace characteristics, we must first construct an approximate gradient $\Delta\mathbf{W}$ for each primitive variable in each zone. There are two alternate ways to construct this gradient: construct $\Delta\mathbf{W}$ by either monotonizing each primitive variable or by monotonizing each characteristic field. In a purely hydrodynamic model, the monotonized central difference algorithm used in van Leer [23] and Colella [7] gives good results. This approximation, which we subsequently refer to as the MUSCL gradient, follows the first approach and monotonizes each primitive variable separately, perhaps with some nonlinear flattening algorithm. In a series of experiments with the MHD code on one-dimensional test problems, using this form of the gradient introduced moderate amplitude (5–10%) fluctuations at shocks, compound waves, and certain types of rarefaction waves. These oscillations occur because under some conditions the MUSCL gradient is too steep. The usual cure—the flattening algorithm of [7]—when applied to the seven MHD variables, does not fix the problem.

The other approach, discussed in [3] and [8] for general systems of conservation laws, monotonizes each characteristic field separately. Unfortunately, this technique does not guarantee that the resulting gradient will be monotonic in each primitive variable. Rather than use either approach independently, we have adopted a modified hybrid of the two. This hybrid could be thought of as a MUSCL gradient combined with a highly sophisticated flattening algorithm.

Our method begins by computing eigenvalues, eigenvectors, and structure coefficients at each zone center. Zachary and Colella [25] give expressions for the eigenvalues and eigenvectors and we do not repeat them here. The structure coefficients, κ_{lm}, are the gradient of each wave speed in the direction of a characteristic field

$$\kappa_{lm} = (\nabla\lambda_l)\cdot\mathbf{r}_m. \tag{7}$$

We then form the following differences:

$$\begin{aligned}
\Delta^-\mathbf{W} &= (\mathbf{W}_i - \mathbf{W}_{i-1}) = \sum \alpha_k^-\mathbf{r}_k,\\
\Delta^+\mathbf{W} &= (\mathbf{W}_{i+1} - \mathbf{W}_i) = \sum \alpha_k^+\mathbf{r}_k,\\
\Delta^c\mathbf{W} &= \frac{1}{2}(\mathbf{W}_{i+1} - \mathbf{W}_{i-1}) = \sum \alpha_k^c\mathbf{r}_k,\\
\Delta^m\mathbf{W} &= \sum \alpha_k^m\mathbf{r}_k.
\end{aligned} \tag{8}$$

The expressions Δ^-, Δ^+, and Δ^c correspond to backward, forward, and centered differences, respectively. The final difference, Δ^m, is the monotonized, fourth-order difference used in the MUSCL algorithm.

We combine these differences to construct a fourth-order monotonized gradient by monotonizing each characteristic field independently. That is, we write $\Delta\mathbf{W}$ as

$$\Delta\mathbf{W} = \sum \overline{\alpha}_k\mathbf{r}_k, \tag{9}$$

where

$$\overline{\alpha}_k = \begin{cases} \min(\gamma|\alpha_k^-|, \gamma|\alpha_k^+|, |\alpha_k^c|, |\alpha_k^m|) \times \text{sign}(\alpha_k^c) & \text{if } \alpha_k^- \alpha_k^+ > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The expansion coefficients α_k^m from the MUSCL gradient enforce monotonicity in each variable.

For strictly hyperbolic systems $\gamma = 2$, however, if an eigenvector is locally linearly degenerate, we use $\gamma = 1$. As in BCT, we detect local linear degeneracy through the structure coefficients. That is, at meshpoint i and for a given characteristic k, whenever

$$\kappa_{kk,i-1}\kappa_{kk,i} < 0 \quad \text{or} \quad \kappa_{kk,i+1}\kappa_{kk,i} < 0, \tag{11}$$

we assume the corresponding eigenvector is linearly degenerate.

Using the monotonized expansion coefficients, we approximate the states at $\mathbf{W}_{i+1/2,L}^{n+1/2}$ and $\mathbf{W}_{i-1/2,R}^{n+1/2}$ by tracing each characteristic forward or backward from zone center. Following the discussion in BCT, we write

$$\begin{aligned} \mathbf{W}_{i+1/2,L}^{n+1/2} &= \mathbf{W}_i^n + \frac{1}{2}\sum_{\lambda_k>0} \overline{\alpha}_k \left(1 - \frac{\Delta x}{\Delta t}\lambda_k\right)\mathbf{r}_k, \\ \mathbf{W}_{i-1/2,R}^{n+1/2} &= \mathbf{W}_i^n - \frac{1}{2}\sum_{\lambda_k<0} \overline{\alpha}_k \left(1 + \frac{\Delta x}{\Delta t}\lambda_k\right)\mathbf{r}_k. \end{aligned} \tag{12}$$

Finally, the predicted states, $\mathbf{W}_L$ and $\mathbf{W}_R$, are modified by the presence of source terms. These source terms can originate from: external forces, i.e., gravity; geometry, i.e., spherical or cylindrical coordinates; or $\partial B_x/\partial x$. Written in strict conservation form, the equations of MHD are each of the form

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla\cdot\mathbf{F} = S(\mathbf{U}), \tag{13}$$

where $S(\mathbf{U})$ represents sources terms. However, we have chosen to perform the characteristic analysis in Lagrangian or primitive variables, $\mathbf{W}$. In the $\mathbf{W}$-basis, (13) becomes

$$\frac{\partial \mathbf{W}}{\partial t} + \mathbf{A}(\mathbf{W})\cdot\frac{\partial \mathbf{W}}{\partial x} = S(\mathbf{W}). \tag{14}$$

The matrix $\mathbf{A}(\mathbf{W})$ is

$$\mathbf{A}(\mathbf{W}) = \left[\frac{\partial \mathbf{U}}{\partial \mathbf{W}}\right]^{-1}\frac{\partial \mathbf{F}}{\partial \mathbf{W}}.$$

Some straightforward algebra shows that the source terms must be transformed according to the rule

$$S(\mathbf{W}) = \left[\frac{\partial \mathbf{U}}{\partial \mathbf{W}}\right]^{-1} S(\mathbf{U}). \tag{15}$$

Assume for the moment that the only source terms come from the incorrect representation of $\partial B_x/\partial x$ in the $\mathbf{B}_\perp$ and in the ρE equations. Then we have

$$S(\mathbf{U}) = \left(0, 0, 0, 0, u_y\frac{\partial B_x}{\partial x}, u_z\frac{\partial B_x}{\partial x}, \frac{\mathbf{u}_\perp\cdot\mathbf{B}_\perp}{4\pi}\frac{\partial B_x}{\partial x}\right)^T.$$

Using (15) to express $S(\mathbf{W})$ in terms of $S(\mathbf{U})$, we find that

$$S(\mathbf{W}) = \left(0, 0, 0, 0, u_y \frac{\partial B_x}{\partial x}, u_z \frac{\partial B_x}{\partial x}, 0\right)^T.$$

In the $\mathbf{W}$-representation, $\partial B_x/\partial x$ changes only the $\mathbf{B}_\perp$-equations, and not the pressure equation.

2.3. The corrector step. We seek solutions to the system of conservation laws

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{F} = 0. \tag{16}$$

As required by the Godunov methodology, we must compute the flux $\mathbf{F}^G(\mathbf{U}_R, \mathbf{U}_L)$ evaluated along the ray $\frac{x}{t} = 0$ in the Riemann problem solution to the above equation with initial conditions

$$\mathbf{U}(x, t = 0) = \begin{cases} \mathbf{U}_L & x \leq 0, \\ \mathbf{U}_R & x > 0. \end{cases} \tag{17}$$

As mentioned in [25], any general solution technique for the MHD version of (16) faces two serious difficulties. First, the MHD wave speeds are not strictly hyperbolic. The loss of strict hyperbolicity at a point means that the analytic structure of the weak solutions is generally unknown in the neighborhood of that point. Second, the genuinely nonlinear waves can be locally linearly degenerate. It is much more difficult to determine the correct entropy satisfying discontinuities for modes with local linear degeneracies.

To deal with these problems, the starting point for our methodology is a high-order Godunov method developed by BCT. We take the BCT point of view and treat the equations of MHD as a hyperbolic system of conservation laws whose weak solutions are uniquely determined by entropy conditions, such as those described in Liu [17] for these systems. The questions surrounding whether this general approach will produce the correct weak solutions are far from settled; some of these issues are discussed in BCT. The algorithm described by BCT is an extension of the Engquist–Osher [12] flux to general systems of conservation laws and is sufficiently robust to handle the nonstrictly hyperbolic nature of the MHD equations. In the exposition that follows, we rely on BCT for the higher-order solution of the Riemann problem.

As discussed in [25], we do not need the entire solution to the Riemann problem. Indeed, the full solution may not be well defined for nonstrictly hyperbolic systems, so it is sufficient to develop the solution as a series of approximations to the full Riemann problem along the ray $\frac{x}{t} = 0$. In the next subsection, we describe the basic structure of our Riemann solver without any of the modifications needed to deal with rarefaction waves or with eigenvector degeneracies. As in the previous section on tracing characteristics, we continue to work in $\mathbf{W}$-space.

2.3.1. The Riemann solver. At each interface, we expand the jump $\mathbf{W}_R - \mathbf{W}_L$ in terms of a linearly independent set of eigenvectors $\overline{\mathbf{R}}_k$, i.e.

$$\mathbf{W}_R - \mathbf{W}_L = \sum_k \alpha_k \overline{\mathbf{R}}_k. \tag{18}$$

We determine $\overline{\mathbf{R}}_k$ from $\mathbf{W}_e = \frac{1}{2}(\mathbf{W}_R + \mathbf{W}_L)$, and let $\overline{\mathbf{R}}_k = \mathbf{r}_k(\mathbf{W}_e)$ be the kth right eigenvector of $D\mathbf{F}$ at $\mathbf{W}_e$.

With these expansion coefficients, we define two intermediate states that "bound" the entropy wave. These states are defined as

$$\mathbf{W}_L^* = \mathbf{W}_L + \sum_{\lambda^-} \alpha_k \mathbf{r}_k, \qquad \mathbf{W}_R^* = \mathbf{W}_R - \sum_{\lambda^+} \alpha_k \mathbf{r}_k. \tag{19}$$

The λ^- waves move backward and the λ^+ waves move forward with respect to the entropy wave.

At each of the four states $\mathbf{W}_L$, $\mathbf{W}_L^*$, $\mathbf{W}_R^*$, and $\mathbf{W}_R$, we compute eigenvalues for all seven MHD waves. From these eigenvalues, we can construct

$$\bar{\lambda}_0^* = \frac{1}{2}\left(\lambda_{0,L}^* + \lambda_{0,R}^*\right),$$

which is the average advective velocity at the starred states. The sign of $\bar{\lambda}_0^*$ gives an unambiguous determination of the upwind state. If $\bar{\lambda}_0^* \geq 0$, then the upwind state lies to the left of the entropy wave, while if $\bar{\lambda}_0^* < 0$, it lies to the right.

In our previous paper, we used the structure coefficients and the wave speeds at the left- and right-hand states and the two intermediate states to construct a cubic Hermite interpolating polynomial for each eigenvalue. We interpolated λ^- waves from $\lambda(\mathbf{W}_L)$ and $\lambda(\mathbf{W}_L^*)$ and λ^+ waves from $\lambda(\mathbf{W}_R)$ to $\lambda(\mathbf{W}_R^*)$. We used the resulting set of interpolants, $\bar{\lambda}_k$, to detect sonic points along each wave path.

In this study, we find that in the presence of large gradients, the structure coefficients can often become so large that they make the Hermite polynomial useless. These large structure coefficients occur when there are substantial changes in density and pressure between $\mathbf{W}_e$, the point at which we know the eigenvectors, and $\mathbf{W}_{L,R}$, or $\mathbf{W}_{L,R}^*$, where we compute $\nabla\lambda_k$. To circumvent this difficulty, we use a simple linear interpolation of the wave speeds from $\mathbf{W}_L$ to $\mathbf{W}_L^*$ and from $\mathbf{W}_R$ to $\mathbf{W}_R^*$. Changing to a linear, rather than a cubic, interpolation does not alter any of our results.

BCT derived a generalized version of the Engquist–Osher flux suitable for systems of nonstrictly hyperbolic equations. Ignoring for the moment the treatment of degenerate waves, and assuming that $\mathbf{U}_L$ is the upwind state, the BCT extension of the Engquist–Osher flux is

$$\mathbf{F}^{EO}(\mathbf{U}_L, \mathbf{U}_R) = \mathbf{F}(\mathbf{U}_L) + \sum_{k=1}^{K} \left(\int_0^{\alpha_k} \min(\bar{\lambda}_k, 0) d\alpha \right) \cdot \bar{\mathbf{R}}_k. \tag{20}$$

As BCT discuss, the integral terms in (20) add a nonnegative dissipation to each of the characteristic modes in the expansion of $\mathbf{U}_R - \mathbf{U}_L$.

Unfortunately, this generalization is unsuitable for multidimensional MHD because the terms of the form $\partial B_x/\partial x$ require special treatment, and because some terms in the flux must be differenced as gradients and some terms as divergences. We, therefore, adopt an alternative formulation of the Engquist–Osher flux. This alternative formulation was presented but not derived in Zachary and Colella [25]. We now discuss the origin of our flux formulation.

Our motivation is the scalar case, for which

$$\begin{aligned} F^{EO} &= f(u_L) + \int_{u_L}^{u_R} \min(a(u), 0) du \\ &= f(u_L) + \int_{u_L}^{u_R} \chi(u) a(u) du, \end{aligned} \tag{21}$$

where $a(u)$ is the wave speed along the path $u_R - u_L$, and $\chi(u)$ is the characteristic function

$$\chi(u) = \begin{cases} 1 & \text{if } a(u) < 0, \\ 0 & \text{if } a(u) > 0. \end{cases}$$

This can be written as the sum of terms of the form

$$\int_{u^p}^{u^{p+1}} a(u)du = f(u^{p+1}) - f(u^p), \tag{22}$$

where $a < 0$ for all $u \in [u^p, u^{p+1}]$, and

$$\begin{aligned} a(u^p) = 0 \quad &\text{or} \quad u^p = u_L, \\ a(u^{p+1}) = 0 \quad &\text{or} \quad u^{p+1} = u_R. \end{aligned} \tag{23}$$

A simple formula for F^{EO} is then given by

$$F^{EO} = f(u_L) + \sum_{s=1}^{S} (-1)^s f(u^s), \tag{24}$$

where $q = \text{sign}(u_R - u_L)$, $qu^1 < qu^2 < \cdots < qu^S$, and u^s satisfies either $a(u^s) = 0$ or

$$\begin{aligned} a(u^1) = 0 \quad u^1 = u_L, \\ a(u^S) = 0 \quad u^S = u_R. \end{aligned}$$

We can use (24) as the starting point for systems. Given the approximate parameterization of the wave curves in §2 of BCT, we can define the Engquist–Osher flux for systems of equations as follows:

$$\mathbf{F}^{EO} = \mathbf{F}(\mathbf{W}_L) + \sum_{k=1}^{K} \sum_{s=1}^{S(k)} \mathbf{F}(\mathbf{W}_k^s)(-1)^s, \tag{25}$$

where

$$\begin{aligned} \mathbf{W}_k &= \mathbf{W}_L + \sum_{k'<k} \overline{\alpha}_{k'} \mathbf{R}_{k'}, \\ \mathbf{W}_k^s &= \mathbf{W}_k + \alpha_k^s \mathbf{R}_k. \end{aligned} \tag{26}$$

Also, $q\alpha_k^1 < q\alpha_k^2 < \cdots < q\alpha_k^S$, and $q = \text{sign}(\overline{\alpha}_k)$. Finally, α_k^s satisfies

$$\begin{aligned} \overline{\lambda}_k(\alpha_k^s) = 0 \quad & 2 \le s \le S(k) - 1, \\ \overline{\lambda}_k(\alpha_k^1) \le 0 \quad & \alpha_k^1 = 0, \\ \overline{\lambda}_k(\alpha_k^{S(k)}) \le 0 \quad & \alpha_k^{S(k)} = \overline{\alpha}_k, \end{aligned} \tag{27}$$

and $\overline{\lambda}_k(\alpha)$ is the linear approximation to the wave speed given above.

The advantage of this approach is that various subpieces of the flux can be defined, as might be required for quasi-conservative differencing. If $\mathbf{F} = \mathbf{Q}_1 + \mathbf{Q}_2$, then $\mathbf{Q}_i$ can be defined as

$$\mathbf{Q}_i^{EO} = \mathbf{Q}_i(\mathbf{W}_l) + \sum_{k=1}^{K} \sum_{s=0}^{S} \mathbf{Q}_i(\mathbf{W}_k)(-1)^s. \tag{28}$$

Then $\mathbf{Q}_1$ could be differenced as a divergence, while $\mathbf{Q}_2$ could be differenced as a gradient, as might be required in a spatially varying geometry.

2.3.2. Rarefaction waves. Rarefaction waves cause problems for most Riemann solvers. Even the fully nonlinear Riemann solver in PPM needs special modification to deal with rarefaction waves, and most other Riemann solvers simply treat rarefaction waves as if they were rarefaction shocks. Our technique of expanding the discontinuity at each zone interface in terms of eigenvectors at a single reference state also does not correctly handle rarefaction waves. In fact, Einfeldt et al. [11] show that there exist initial conditions for strong rarefaction waves under which any linear Riemann solver must fail. With this failure in mind, we have modified our Riemann solver whenever we detect a strong rarefaction. Our detection criterion is simply that if

$$v_f(\mathbf{W}_e) < \lambda_0(\mathbf{W}_R) - \lambda_0(\mathbf{W}_L), \tag{29}$$

where v_f is the velocity of the fast magnetosonic wave, then we assume there is a strong rarefaction wave at this interface. Using the procedures in Einfeldt et al., particularly, equations (4.4) and (4.5), we recompute the flux as

$$\mathbf{F}^G(\mathbf{W}_R, \mathbf{W}_L) = \frac{b^+\mathbf{F}(\mathbf{W}_R) - b^-\mathbf{F}(\mathbf{W}_L)}{b^+ - b^-} + \frac{b^+ \cdot b^-}{b^+ - b^-}(\mathbf{U}_R - \mathbf{U}_L). \tag{30}$$

Here, $b^+ = \max(\lambda_R^+, \lambda_e^+, 0)$ and $b^- = \min(\lambda_L^-, \lambda_e^-, 0)$; see [11] for details. This particular representation of the Godunov flux is positively conservative.

Rarefaction waves also pose problems for our algorithm since we use density rather than specific volume as an independent variable. (Using specific volume rather than density poses similar problems at compressions.) These problems arise when estimating the density at intermediate states, particularly at $\mathbf{W}^*_{L,R}$. When we construct $\mathbf{W}^*_L$ or $\mathbf{W}^*_R$, if

$$\rho(\mathbf{W}^*_L) < \rho(\mathbf{W}_L) \quad \text{or} \quad \rho(\mathbf{W}^*_R) < \rho(\mathbf{W}_R),$$

then we interpolate in specific volume rather than in density on that particular side of the entropy wave. Using the eigenvectors in [25], we note that a change in basis from ρ to τ only changes the density components of the right and left eigenvectors. The only nonzero density component of a left eigenvector belongs to the entropy wave, and since the left eigenvectors determine the expansion coefficients, only the expansion coefficient for the entropy wave, α_0, changes under this change in basis. Note, however, that α_0 is never used by construction. We simply keep track of the appropriate representation for the right eigenvectors on each side of the entropy wave. The rest of our method for constructing the Engquist–Osher flux remains unchanged.

2.3.3. Degenerate waves. The MHD equations are nonstrictly hyperbolic because there are points with degenerate eigenvalues. At these points, the wave ordering that applies in strictly hyperbolic systems no longer holds and two or more wave speeds coincide. In MHD, the eigenvalues become degenerate in two distinct limits: if $B_x = 0$ and if $\mathbf{B}_\perp = 0$. In the first limit, the Alfven and the slow magnetosonic wave travel at the same speed as the entropy wave, resulting in a five-fold degeneracy. In the second limit, there are three different degeneracies that can occur depending upon the values of the Alfven velocity, v_{ax}, and the sound velocity c_s. When $v_{ax} > c_s$, the fast magnetosonic and the Alfven wave are degenerate, while when $v_{ax} < c_s$, the Alfven wave and the slow magnetosonic wave are degenerate, and finally when $v_{ax} = c_s$, all three waves are mutually degenerate. In each limit, and for each subcase, the algorithm must detect any degeneracy and then take appropriate action.

It is important here to clarify the exact meaning of such degeneracies and why they are peculiar to MHD. According to the definition given by Lax [15], two or more waves are considered to be degenerate when their values coincide, resulting in a loss of strict hyperbolicity.

In this case it may happen that the eigenvectors corresponding to the orginally distinct eigenvalues become essentially parallel, resulting in an eigenvector deficiency like those considered in BCT. Also, Lax [16] defines a wave to be linearly degenerate when $(\nabla\lambda_k)\cdot\mathbf{r}_k = 0$, where $\mathbf{r}_k$ is the right eigenvector corresponding to the kth characteristic field. This is the case, for example, for the entropy and Alfven waves.

We emphasize that with our choice of eigenvectors (Zachary and Colella [25]), the degeneracy of the eigenvalues does not also imply a deficiency in the eigenvectors. We always have a set of complete, linearly independent eigenvectors spanning the phase space. The same situation already occurs in the equations for pure three-dimensional hydrodynamics: the entropy wave has in fact a three-fold degeneracy that is related to the transport of entropy and shear, and there are three linearly independent eigenvectors associated with it. In pure hydrodynamics, the three-fold degeneracy always occurs, and is independent of the value of the physical variables. Therefore, purely hydrodynamics equations can still be viewed as a strictly hyperbolic system, and it can be shown that the Engquist–Osher formulation [12] does compute fluxes that satisfy the entropy condition. In MHD, however, waves that are genuinely nonlinear in the sense of Lax [16] (specifically, the fast and slow magnetosonic waves) do become linearly degenerate when their eigenvalues coincide with those of already linearly degenerate waves, i.e., the entropy and Alfven waves. If this condition happens anywhere along the wave-path of integration between the left and right states, then the Engquist–Osher formulation may fail and the entropy condition may not be satisfied. When this happens, we need to modify our scheme and apply a different type of dissipation, as described below.

The first task in dealing with degeneracies is detecting them. Analytically, two eigenvectors are degenerate whenever their corresponding eigenvalues coincide. It is rare, however, that eigenvalues will, numerically, be exactly equal; therefore we have adopted here the BCT detection algorithm. The BCT algorithm was designed with the idea in mind to identify points in a general system of hyperbolic conservation laws at which eigenvectors become parallel and the corresponding expansion coefficients are no longer reliable. However, the same approach can be utilized more generally to detect loss of strict hyperbolicty, independent of the fate of the eigenvectors. As already mentioned, our construction of the MHD eigenvectors insures that when the eigenvalues coincide, the corresponding eigenvectors are not deficient, but an originally genuinely nonlinear wave may become linearly degenerate. We adopt the BCT idea that two eigenvectors are degenerate whenever the difference in their eigenvalues at $\mathbf{W}_e$ is less than some fraction of the difference in the predicted variation of the eigenvalues over the wave path. Specifically, we say that two eigenvectors are degenerate whenever

$$|\lambda_l^e - \lambda_m^e| \le c \sum_k |\alpha_k(\kappa_{kl} - \kappa_{km})|. \tag{31}$$

(Here, as elsewhere, we use $c = 0.1$.)

The eigenvector degeneracy detection criterion in (31) fails in the limit $B_x/\mathbf{B}_\perp \to 0$, where the Alfven and slow magnetosonic waves become degenerate, but where the structure coefficients are independent of B_x. (We define $\kappa_{lm} = (\nabla\lambda_l)\cdot\mathbf{r}_m$, but there is no component of $\mathbf{r}_m$ along B_x.) We, therefore, adopt an alternative criterion, empirically derived, which states that the eigenvectors corresponding to the Alfven and slow magnetosonic waves become degenerate whenever

$$\frac{|v_{ax}^e|}{v_f^e} \le c^2 \frac{\left(\mathbf{B}_\perp^2(\mathbf{W}_e) + \mathbf{B}_\perp^2(\mathbf{W}_L) + \mathbf{B}_\perp^2(\mathbf{W}_R)\right)^{1/2}}{|B_x(W_e)|}, \tag{32}$$

where v_{ax}^e and v_f^e are, respectively, the Alfven velocity and fast mode velocity at $\mathbf{W}_e$.

The linear Riemann solver described in the previous sections breaks the full Riemann problem into two distinct parts, with each part corresponding to waves that move forward or backward with respect to the entropy wave. Accordingly, we treat all waves that are degenerate with the entropy wave as if they were additional entropy waves. Recall that from (19), entropy waves are not included in the construction of $\mathbf{W}^*_{L,R}$, nor are they incorporated in the computation of the flux.

Any other changes in the flux computation depend upon whether the eigenvector degeneracy is associated with a change in the sign of the wave speed along the wave path. Assume that the eigenvalues λ^e_l and λ^e_m satisfy (31), and assume that $\bar{\lambda}^*_0 > 0$, so that the reference state is $\mathbf{W}_L$. Let

$$\lambda^{\max}_{lm} = \max(\lambda^L_l, \lambda^{L^*}_l, \lambda^L_m, \lambda^{L^*}_m)$$

and

$$\lambda^{\min}_{lm} = \min(\lambda^L_l, \lambda^{L^*}_l, \lambda^L_m, \lambda^{L^*}_m).$$

When $\lambda^{\min}$ and $\lambda^{\max}$ have the same sign, (25) remains unchanged. When $\lambda^{\min}_{lm}$ and $\lambda^{\max}_{lm}$ have opposite signs, then we assume that the eigenvector degeneracy is associated with a transonic wave. In this case, we replace the flux correction terms in (25) with a dissipative term from Rusanov's [21] scheme. More exactly, the added flux correction term is

$$\mathbf{F}^{EO} = \mathbf{F}(\mathbf{W}_L) - \frac{1}{2}\nu \sum_{k=l,m} \alpha_k \overline{\mathbf{R}}_k + \sum_{k\neq l,m} \sum_{s=1}^{S(k)} \mathbf{F}(\mathbf{W}^s_k)(-1)^s, \tag{33}$$

where $\nu = \max(|\lambda^{\max}_{lm}|, |\lambda^{\min}_{lm}|)$. Although this description of a Rusanov-type dissipation assumes that only two waves were degenerate, it is easily generalized to include the case where many different waves are degenerate.

Note that if (32) is true, and the Alfven and slow waves are degenerate, then it is possible for waves to be degenerate across the entropy wave. In that case, the definitions for $\lambda^{\min}$ and $\lambda^{\max}$ simply look at the wavespeeds at $\mathbf{W}_L$ and $\mathbf{W}_R$, without considering the wavespeeds at the intermediate states.

2.4. Insuring $\nabla\cdot\mathbf{B} = 0$. The problem of preserving $\nabla\cdot\mathbf{B} = 0$ to the highest possible accuracy, possibly even to machine accuracy, is a crucial one for discretized versions of the MHD equations. While the continuum equations insure that an initially solenoidal magnetic field will remain solenoidal when evolving in time, the discretized equations do not. As a consequence, numerical solutions of the MHD equations can generate magnetic monopole forces that strongly affect the dynamics of the fluid (See, e.g., Brackbill and Barnes [4]). Writing the equations in variables other than the magnetic field, like the vector potential or the magnetic flux function, partially solves the problem. This approach, however, has two drawbacks: it requires a staggered mesh, and the equations are not in a form suitable for characteristic analysis.

In our code we have used a centered, finite-difference scheme based on primitive variables because this scheme is the natural choice for higher-order Godunov methods. Our choice means that $\nabla\cdot\mathbf{B} = 0$ is preserved only to the truncation error of the method. In principle, that truncation error could be $O(1)$, since we are computing solutions in the presence of discontinuities. To deal with that problem, we follow the prescription in [4], remove the nonsolenoidal part of $\mathbf{B}$, and then apply a Hodge projection to $\mathbf{B}$ at the end of each timestep. First solve the Poisson equation for the potential ϕ

$$\nabla^2\phi + \nabla\mathbf{B} = 0, \tag{34}$$

then the new solenoidal magnetic field is defined by $\mathbf{B} = \mathbf{B} - \nabla\phi$. Here we used second-order central difference approximations for the gradient and divergence operators. This numerical form of the Hodge projection leads to nonstandard discretizations of the Laplacian. However, the resulting linear equations can be solved efficiently and accurately using interative methods, e.g., multigrid (see Bell, Colella, and Howell [2]). It is worth noting that, for the two-dimensional problems presented in this paper, we did not find any noticeable difference regardless of whether or not we applied a projection.

3. Numerical results. We have tested our algorithm on a variety of MHD problems in one and two dimensions to check the consistency and robustness of our method under different conditions on the physical variables. Each problem tests different features of the code, such as its behavior in special limiting conditions, e.g., in the purely hydrodynamic limit, or its ability to track discontinuities. Our testing procedure has been particularly useful as we tuned the degeneracy detection algorithm, which is a crucial component of our Riemann solver.

All computations described below were performed on a Cray Y-MP8e/8128-4 at the Cray Research, Inc., data center in Eagan, Minnesota, and on a Cray Y-MP8/464 at the NASA Center for Computational Sciences (NCCS) in Greenbelt, Maryland. At the moment, the current version of the code, including all the test and corrections for degeneracy, the construction of two different sets of eigenvectors for each meshpoint, and the modifications for rarefaction waves, takes 15 μs per cell in one dimension and twice that in two dimensions. In other words, the algorithm does approximately 32,000 meshpoints/s in two dimensions. Our timestep was set by the Courant–Friedrichs–Lewy (CFL) stability condition

$$\Delta t = \sigma \max_{i,j} \left(\frac{\Delta x_i}{|u_{x,i}| + v_{f,i}}, \frac{\Delta y_i}{|u_{y,j}| + v_{f,j}} \right), \tag{35}$$

where σ is the Courant number, and the maximum is taken over $i = 1, \ldots, N_x$, $j = 1, \ldots, N_y$. All the calculations were done using a Courant number $\sigma = 0.8$ without any artificial viscosity.

3.1. One-dimensional problems. Our suite of one-dimensional problems includes benchmarks commonly used to test numerical algorithms: the Sod shock tube problem [22], the Brio and Wu problem [5], and the strong version of the Sod shock tube problem [5]. In addition, we have studied the strong rarefaction problems considered by Einfeldt et al. [11]. Each problem tests a different aspect of the Riemann solver. The Sod shock tube problem tests the solver in the purely hydrodynamic limit, the Brio and Wu problem stresses our handling of linear degeneracies and compound waves, while the strong version of the Brio and Wu problem accents our handling of large-amplitude shock waves. Finally, the Einfeldt et al. problems test our ability to detect and handle strong rarefaction waves.

As mentioned above, the Sod shock tube is a purely hydrodynamical Riemann problem in which the initial condition consists of two uniform states $\mathbf{W}_L$ and $\mathbf{W}_R$ separated by a discontinuity. The gas polytropic index is $\gamma = 1.4$, and the initial conditions are

$$\mathbf{W}_L(x < 0) = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \mathbf{W}_R(x > 0) = \begin{bmatrix} .125 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.1 \end{bmatrix}.$$

The numerical solution has been obtained for 100 grid points (with $\Delta x = 1$) and is shown in Fig. 1 after 50 timesteps (Δt being given by the CFL condition). The shock discontinuity is well resolved within three grid points. The contact discontinuity is slightly more diffused, which is expected since we have not used a detection algorithm (see [9]). Also, there is a small undershoot at the base of the rarefaction wave that is a peculiar feature of how our linearized solver treats rarefaction waves. This undershoot can be partially eliminated by adding a quadratic artificial viscosity to our scheme of the type described by BCT. In Fig. 2 we show a comparison between the density profile at the same timestep as computed by our method and by the PPM method of Colella and Woodward [9]. Both solutions are very similar, with the main difference being the undershoot discussed above; one can see that the contact discontinuity is better represented by the PPM code that used a contact detection algorithm.

In Fig. 3 we present the results of the Brio and Wu [5] problem, which is an MHD analogue of the Sod shock tube problem. The initial configuration is the same as the Sod shock tube problem, but with $\gamma = 2$; the initial magnetic field is $By_l = \sqrt{4\pi}$, $By_r = -\sqrt{4\pi}$, and $Bx = 0.75\sqrt{4\pi}$. Here we have used a grid of 800 points and 400 timesteps. The solution is in excellent agreement with the one obtained by Brio and Wu [5] using a Riemann solver written specifically for this problem. Again, we note that our method introduces a small perturbation at rarefaction waves. This perturbation does not affect the solution seriously and was previously noted also by Zachary and Colella [25]. On the other hand, all the shocks are extremely sharp and do not show any post-shock oscillations. In Fig. 4 we present the last of our set of Riemann problems, the strong version of the magnetized Sod shock tube problem that was also discussed by [5]. Here we set $B_x = 0$, $P_L = 1000$, $\gamma = \frac{5}{3}$, and all the other parameters as before. The solution is very well behaved even in this extreme case, and we note that we could solve this problem with a $\gamma \neq 2$, while the Brio and Wu solver could not.

Our final one-dimensional test problem is drawn from the strong rarefaction problems discussed by Einfeldt et al. [11]. In our earlier discussion of the Riemann solver, we noted that conditions exist under which any linear Riemann will fail, even if the underlying physical problem admits a solution. We have implemented a strong rarefaction wave detection algorithm that automatically applies additional dissipation whenever the Riemann solver fails. In Fig. 5 we show the results for the problem defined as $1-2-0-3$ in [11]. This problem consists of two strong rarefaction waves moving symmetrically in opposite directions. With an adiabatic index $\gamma = 1.4$, the initial conditions are

$$\mathbf{W}_L(x<0) = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.4 \end{bmatrix}, \qquad \mathbf{W}_R(x>0) = \begin{bmatrix} l \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.4 \end{bmatrix}.$$

From the analysis in [11], these initial conditions have a physical solution, but the solution is not linearizable.

3.2. Two-dimensional problems. Among the various two-dimensional test cases that we have tried, we present here two particularly interesting ones: a spherically symmetric explosion in a uniformly magnetized medium and the compressible version of the Orszag–Tang vortex (see Dahlburg and Picone [10] and Picone and Dahlburg [19]). Both problems develop interesting wave patterns and shocks that propagate in all directions, and therefore

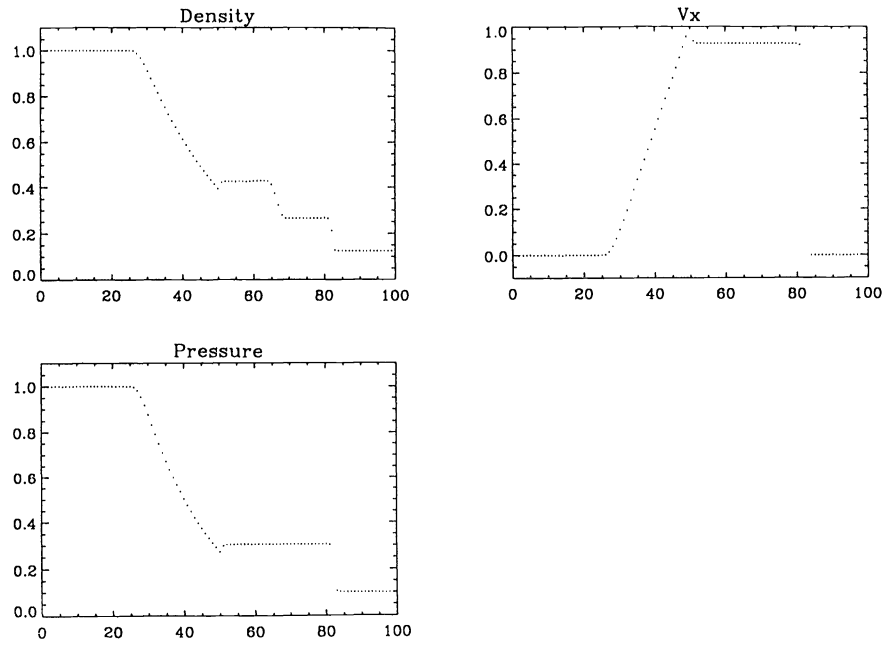


FIG. 1. *The solution to the Sod shock tube problem on a uniform mesh with* 100 *grid points is shown after* 50 *timesteps. The size of the timestep was controlled by the* CFL *condition with* CFL *number of* 0.8.

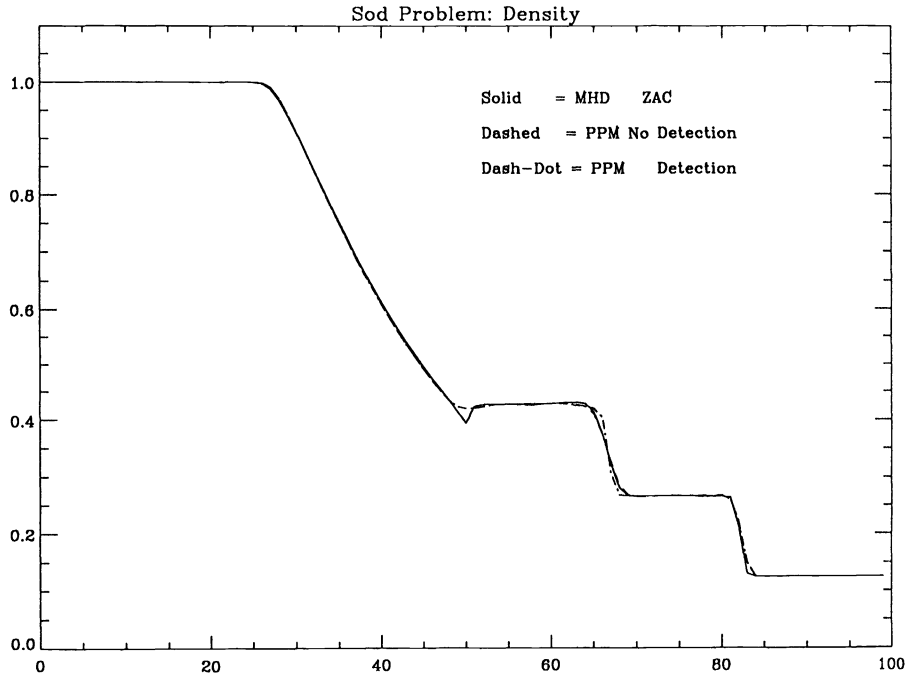


FIG. 2. *The comparison between the density profiles of the solution to the Sod shock tube problem computed after* 50 *timesteps by using different algorithms is presented. The solid line is the solution computed with the present code; the dashed line is the solution computed with the* PPM *code in which the contact detection algorithm has been turned off; the dashed-dotted line is the* PPM *solution with detection of contact discontinuities. The effect of contact detection on the resolution of the contact discontinuity is clearly visible.*

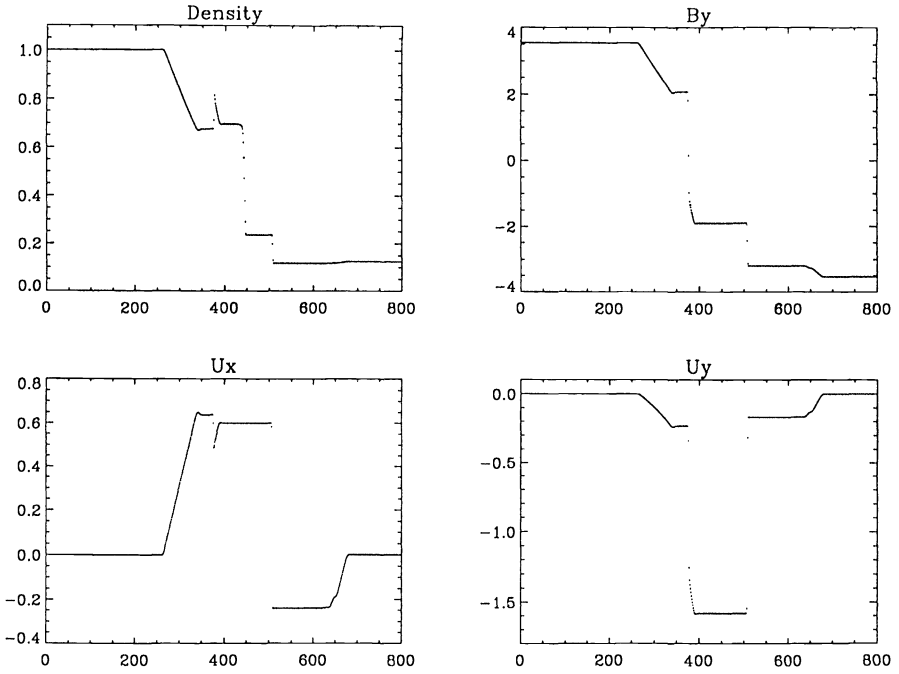


FIG. 3. *The solution to the Brio and Wu magnetized shock tube problem on a uniform mesh with* 800 *grid points is shown after* 400 *timesteps. The initial condition consists of a jump discontinuity in pressure and density separated by a current sheet.*

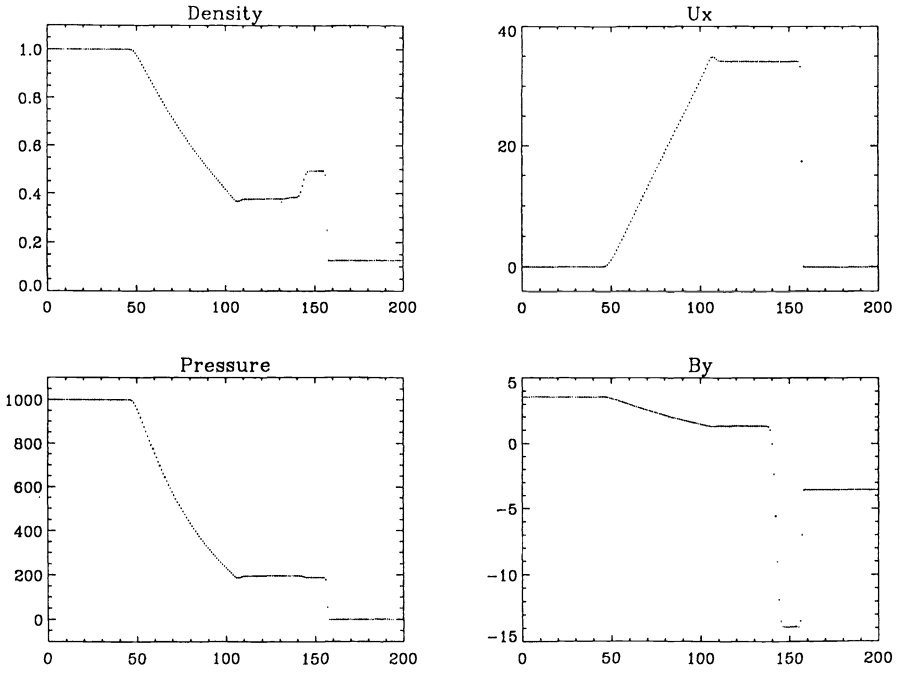


FIG. 4. *The solution to the strong version of the shock tube problem on a mesh with* 200 *grid points is shown after* 100 *timesteps. Here* $B_x = 0$ *and* B_y *changes sign across the initial discontinuity. A* CFL *condition of* 0.8 *was used.*

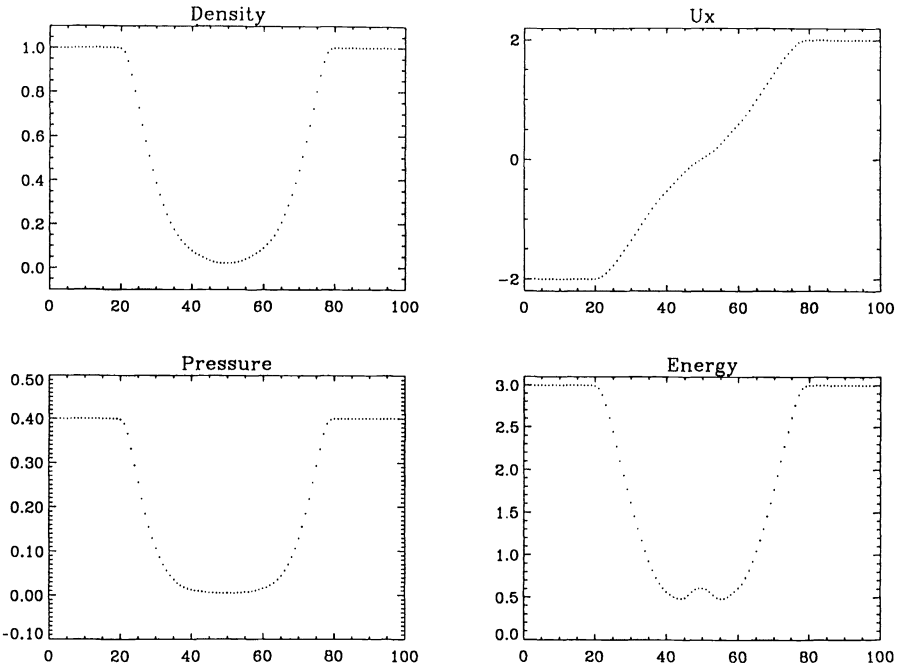


FIG. 5. *The solution to the Einfeldt et al.* [11] 1–2–0–3 *strong rarefaction problem on a mesh with* 100 *grid points is shown at time* 0.1.

they are very useful tests of the two-dimensional properties of our code. One of the main differences between the purely one-dimensional problems and the two-dimensional problems is that the magnetic field **B** varies in the direction of spatial integration in each of the one-dimensional sweeps of the operator-split method. This introduces extra source terms in the equations and may give rise to a new set of degeneracies that are not present in the purely one-dimensional problems (see the discussion in the previous section).

3.2.1. Spherical explosion. The explosion is driven by a spherical region (with $r = 0.1$) with a large overpressure. The initial density and pressure are $\rho = 1$, $P_e = 1$, and the overpressure is $P_i = 100$. We have tried three different values of the initially uniform magnetic field: $B_y = 0$, $B_y = 10$, and $B_y = 100$. Our computation mesh is a cartesian uniform grid with 120 points in each direction. This problem is identical to the one described by Kössl, Müller, and Hillebrandt [14], and we have used their results as a qualitative comparison. In Figs. 6, 7, and 8, we show the contour plots of density, thermal pressure, magnetic pressure, and kinetic energy for the three cases in which $B_y = 0$, $B_y = 10$, and $B_y = 100$. For $B_y = 0$ (Fig. 6), there is one spherically symmetric hydrodynamical shock wave that propagates outward. This solution is essentially identical to the one obtained with other purely hydrodynamical codes, e.g., PPM. For $B_y = 10$, the shock wave is still mostly spherically symmetric, but it becomes slightly elongated in the direction of the magnetic field.

For $B_y = 100$, the explosion becomes highly anisotropic: there is essentially no displacement of gas in the direction transverse to the magnetic field, and two hydrodynamical shocks propagate in the parallel direction. In this highly magnetized fluid, the slow magnetosonic wave speed is almost equal to the sound speed, and the fast wave speed is almost equal to the Alfven velocity. Several weak magnetosonic waves are radiated transverse to the magnetic field as an initial transient until total pressure equilibrium is reached at the center. While

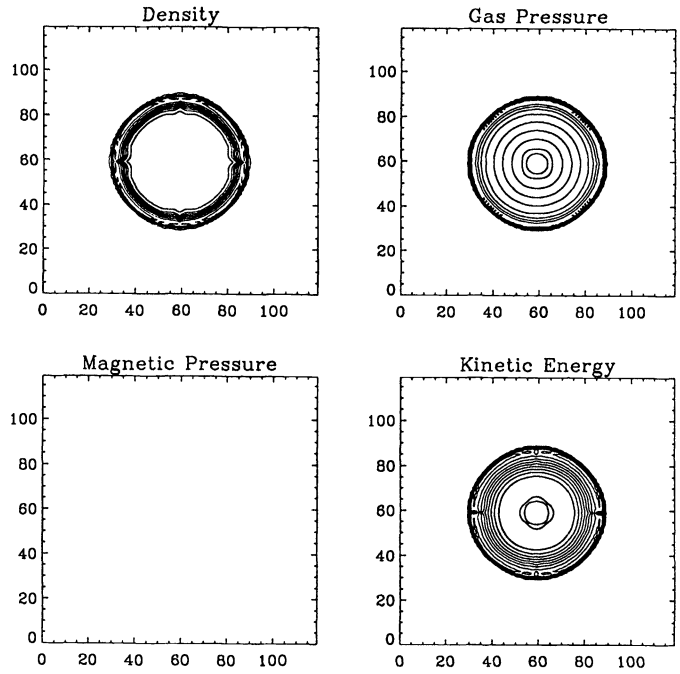


FIG. 6. *The solution to the spherically symmetric explosion problem in the hydrodynamic limit* $By_0 = 0$ *is shown on a mesh with* 120*-by-*120 *grid points after* 48 *timesteps. For each variable, there are* 20 *equally spaced contour levels. A* CFL *condition of* 0.8 *was used. No artificial viscosity or flattening was applied.*

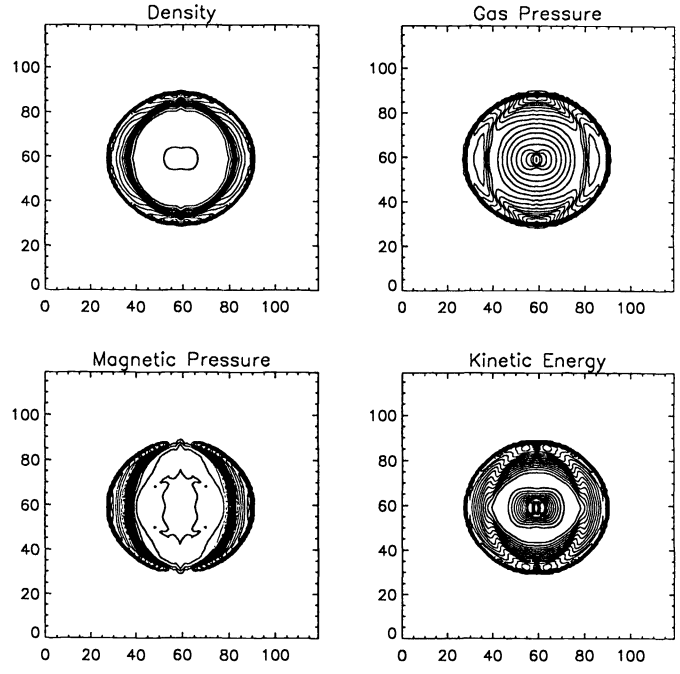


FIG. 7. *The solution to the spherically symmetric explosion problem with an initial field* $By_0 = 10$ *is shown after* 48 *timesteps. All the other conditions are the same as in Fig.* 6.

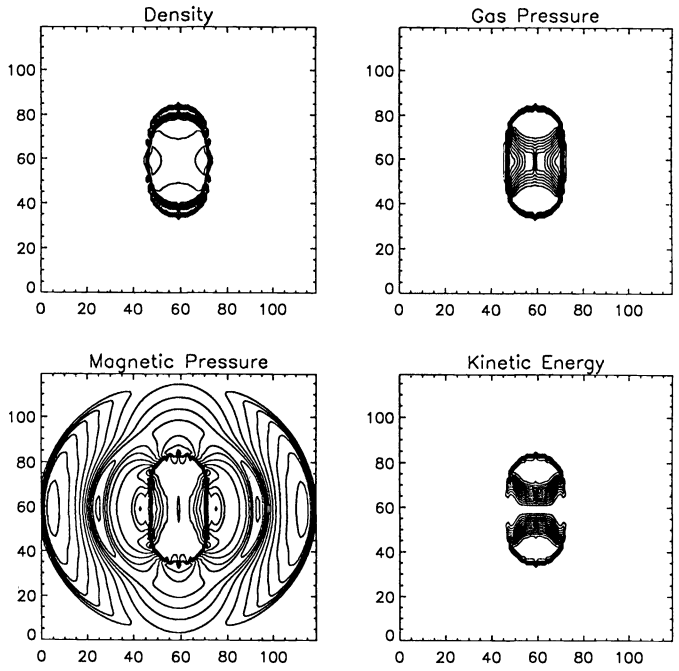


FIG. 8. *The solution to the spherically symmetric explosion problem with an initial field* $By_0 = 100$ *is shown after* 48 *timesteps. All the other conditions are the same as in Fig.* 6.

the results are in good agreement with the results of [14], we see that the waves and shocks look very sharp and clean with little or no oscillations. The strong steepening of the profiles generates the step-like irregularities that appear in the contour lines near the strong discontinuities. These irregularities are a well-known feature of higher-order Godunov methods (see Woodward and Colella [24]). A flattening algorithm and a quadratic artificial viscosity will cure this effect and prevent the appearance of low-amplitude nonphysical oscillations.

3.2.2. Orszag–Tang vortex. In the second numerical example, we studied the evolution of a compressible Orszag–Tang vortex system like the ones described by Dahlburg and Picone [10], [19]. This problem has been introduced by Orszag and Tang [18] as a simple model to study the evolution of MHD turbulence, and it has been generalized by Dahlburg and Picone [10] for the case of a fully compressible medium. We have selected this problem to test how our two-dimensional code treats the interactions between the several shock waves generated as the vortex system evolves. The initial conditions have a periodic structure containing X-points in both the velocity and the magnetic field. The velocity and magnetic fields have different modal structures. We selected an initially uniform pressure and density based on the value of the average Mach number. The initial configuration is as follows: we choose a cartesian uniform grid with periodic boundaries and 192-by-192 grid points. The fields are given as in [19]:

$$\rho(x, y, t = 0) = \rho_0,$$
$$P(x, y, t = 0) = P_0,$$
$$\mathbf{v}(x, y, t = 0) = -\sin(y)\hat{\mathbf{x}} + \sin(x)\hat{\mathbf{y}},$$
$$\mathbf{B}(x, y, t = 0) = -\sin(y)\hat{\mathbf{x}} + \sin(2x)\hat{\mathbf{y}},$$

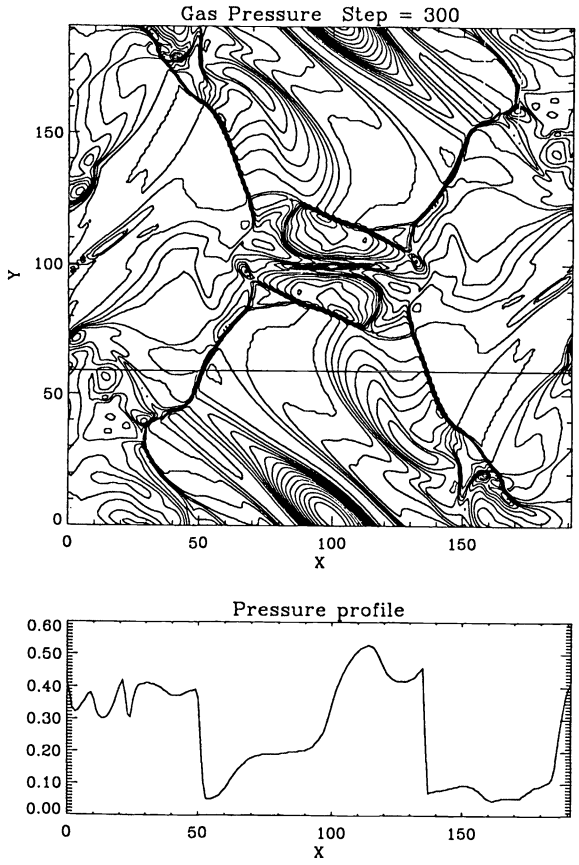


FIG. 9. *The evolution of the compressible Orszag–Tang vortex system for the case with initial Mach number* $M_0 = 1$ *and initial* $\beta = \frac{10}{3}$. *The solution was computed on a periodic cartesian mesh with* 192*-by-*192 *zones and is shown after* 300 *timesteps (approximately* 2 *Alfven transit times). The upper plot shows equally spaced contour levels of the thermal pressure. There are several shock fronts that propagate at transverse directions with respect to the grid and eventually interact. The lower plot shows the pressure profile along the solid line at* $Y = 60$. *There are two sharp jumps clearly visible where the solid line intersects the two strong shock fronts.*

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are unit vectors in the x and y directions. The initial average Mach number is given by $M^2 = \rho_0 |\mathbf{V}_0| / (\gamma P_0)$, where $|\mathbf{V}|_0$ is the initial root mean square (rms) value of the velocity. The initial average β, i.e., the ratio of thermal and magnetic pressure, is $\beta = 8\pi P_0 \, |\mathbf{B}^2|$.

We examined the case with $M = 1.0$ and $\beta = \frac{10}{3}$. In Fig. 9 we present the contour profiles of the thermal pressure after 300 timesteps. The plot shows the pressure profile along the solid line at $y = 60$, and it shows how the shocks remain sharply defined within a few grid points even in two dimensions. There are several shock fronts that propagate at different angles with respect to the grid, and which interact until the vortex system decays and gives rise to small-scale structures. The dynamics of the decay appears to closely match the one described in Dahlburg and Picone [10], even though we are not using exactly the same initial conditions. We note that in our code, the decay of the vortex system is determined by the numerical viscosity and resistivity that is built into the dissipation mechanism of the code. The similarity between our results and the results of Dahlburg and Picone, who use physical dissipation, therefore suggests that the numerical dissipation indeed provides a good model for the physical diffusive processes at subgrid scales. This conjecture, which deserves further

experiments, has been advanced also for the case of hydrodynamic turbulence by, for example, Cattaneo et al. [6].

4. Discussion and conclusions. We have derived an explicit, second-order Godunov method for the equations of ideal MHD in two and three spatial dimensions. Our approach contains many of the same elements applied successfully to ideal, compressible hydrodynamics. Most of the general techniques that improve the behavior of Godunov methods in hydrodynamics, such as flattening and artificial viscosity, carry over to our code. Our main innovation is the construction of an approximate, linearized Riemann solver. This Riemann solver detects and handles the degeneracies that occur in ideal MHD, and is well behaved for physically acceptable states. By well behaved we mean that the Riemann solver does not produce nonphysical states like, for example, negative pressures and densities in the presence of strong rarefaction waves.

To apply the one-dimensional characteristic analysis to the formulation of the Riemann problem, we rewrote the equations in quasi-conservative form. That is, we handle the terms containing derivatives of the magnetic field component parallel to each spatial direction, $\partial B_x/\partial x$ and $\partial B_y/\partial y$, as source terms, not as fluxes. In general coordinates, we split the terms in the equations into different components: some that must be treated as divergences, some as gradients, and some as pure source terms. Our particular quasi-conservative formulation is well suited to the operator-split methodology on which we based our code. At each timestep, we solve two sets of one-dimensional equations, one set for each spatial direction. These equations are not the same as the purely one-dimensional equations because they contain source terms arising from the Maxwell equation $\nabla\cdot\mathbf{B} = 0$. Furthermore, we wrote the momentum equations in such a way that the condition $\nabla\cdot\mathbf{B} = 0$ applies exactly to the continuum equations, even if it does not apply to the discretized version. For the examples we have considered, our approach appears to suppress the occurrence of monopole forces that would lead to unphysical dynamics. In cases where our basic difference approximation does not suffice to maintain the divergence-free condition of the magnetic field, we have the option of applying a Hodge decomposition to the magnetic field at each timestep to eliminate the monopole component.

The code achieves high resolution and accuracy in the presence of strongly discontinuous solutions. This high performance has some limitations, however. Because performing a Riemann solve is computationally expensive, and because the time advance of the solution is limited by the CFL condition, our code is best suited to studying transient phenomena that involve the propagation and interaction of shock discontinuities and MHD waves. While the current formulation of the code can already be used to study a wide variety of problems, there are good reasons to construct a fully unsplit Godunov scheme for the MHD equations. We expect that an unsplit method will be more appropriate when studying systems with strongly sheared fields, and that generally such an approach will improve the advection of the solenoidal magnetic field. This problem is analogous to the advection of the velocity field in an incompressible medium. Colella [8] discusses a general method of constructing unsplit methods that make use of the multidimensional wave propagation properties of the solution to construct the Godunov fluxes. We believe that the same approach will work also for the equations of MHD, and that our Riemann solver will be the base of one such scheme. As usual, the final answer must rely on numerical experiments.

Acknowledgments. Andrea Malagoli wishes to thank D. Spicer and the staff of the NASA Center for Computational Sciences (NCCS) for providing the supercomputing resources. We also thank Cray Research for generously allowing us access to its computer resources.

REFERENCES

[1] J. B. BELL, P. COLELLA, AND H. M. GLAZ, *A second-order projection method for the incompressible Navier–Stokes equations*, J. Comput. Phys., 85 (1989), pp. 257–283.

[2] J. B. BELL, P. COLELLA, AND L. HOWELL, *An efficient second-order projection method for viscous incompressible flow*, Proc. 10th AIAA Computational Fluid Dynamics Conf., Honolulu, HI, June 24–27, 1991, pp. 360–367.

[3] J. B. BELL, P. COLELLA, AND J. A. TRAGENSTEIN, *Higher-order Godunov methods for general systems of hyperbolic conservation laws*, J. Comput. Phys., 82 (1989), pp. 362–397.

[4] J. U. BRACKBILL AND D. C. BARNES, *The effect of nonzero $\nabla \mathbf{B}$ on the numerical solution of the magnetohydrodynamic equations*, J. Comput. Phys., 35 (1980), pp. 426–430.

[5] M. BRIO AND C. C. WU, *An upwind differencing scheme for the equations of ideal magnetohydrodynamics*, J. Comput. Phys., 75 (1988), pp. 400–422.

[6] F. CATTANEO, N. H. BRUMMELL, J. TOOMRE, A. MALAGOLI, AND N. E. HURLBURT, *Turbulent compressible convection*, Astrophys. J., 370 (1991), pp. 282–294.

[7] P. COLELLA, *A direct Eulerian* MUSCL *scheme for gas dynamics*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 104–117.

[8] ———, *Multidimensional upwind methods for hyperbolic conservation laws*, J. Comput. Phys, 87 (1990), pp. 171–200.

[9] P. COLELLA AND P. R. WOODWARD, *The piecewise parabolic method* (PPM) *for gas-dynamical simulations*, J. Comput. Phys., 54 (1984), pp. 174–201.

[10] R. B. DAHLBURG AND J. M. PICONE, *Evolution of the Orszag–Tang vortex system in a compressible medium. I. Initial average subsonic flow*, Phys. of Fluids B, 1 (1989), pp. 2153–2171.

[11] B. EINFELDT, C. D. MUNZ, P. L. ROE, AND B. SJÖGREEN, *On Godunov-type methods near low densities*, J. Comput. Phys., 92 (1991), pp. 273–295.

[12] B. ENGQUIST AND S. OSHER, *One-sided difference approximations for nonlinear conservation laws*, Math. Comp., 36 (1981), pp. 321–351.

[13] A. HARTEN, P. D. LAX, AND B. VAN LEER, *On upstream differencing and Godunov-type schemes for hyperbolic conservations laws*, SIAM Rev., 25 (1983), pp. 35–61.

[14] D. KÖSSL, E. MÜLLER, AND H. HILLEBRANDT, *Numerical simulations of axially symmetric magnetized jets*: I. *The influence of equipartition magnetic fields*, Astron. and Astrophys., 229 (1990), pp. 378–396.

[15] P. D. LAX, *Weak solutions of nonlinear hyperbolic equations and their numerical computation*, Comm. Pure Appl. Math, 7 (1954), pp. 159–193.

[16] ———, *Hyperbolic systems of conservation laws and the mathematical theory of shock waves*, Conference Board of the Mathematical Sciences Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1973.

[17] T. P. LIU, *The Riemann problem for general systems of conservation laws*, J. Differential Equations, 18 (1981), pp. 218–234.

[18] S. A. ORSZAG AND C. M. TANG, *Small-scale structure of two-dimensional magnetohydrodynamic turbulence*, J. Fluid. Mech., 90 (1979), pp. 129–143.

[19] J. M. PICONE AND R. B. DAHLBURG, *Evolution of the Orszag–Tang vortex system in a compressible medium. II. Supersonic flow*, Phys. of Fluids B, 3 (1991), pp. 29–44.

[20] P. L. ROE, *Approximate Riemann solvers, parameter vectors, and difference schemes*, J. Comput. Phys., 53 (1981), pp. 357–372.

[21] V. V. RUSANOV, *Calculation of interaction of non-steady shock waves with obstacles*, Zh. Vychisl. Mat. i Mat. Fiz., 1 (1961), pp. 267–279.

[22] G. SOD, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 27 (1978), pp. 1–31.

[23] B. VAN LEER, *Towards the ultimate conservative difference scheme* IV. *A new approach to numerical convection*, J. Comput. Phys., 23 (1977), pp. 276–299.

[24] P. R. WOODWARD AND P. COLELLA, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1984), pp. 115–173.

[25] A. ZACHARY AND P. COLELLA, *A higher-order Godunov method for the equations of ideal magnetohydrodynamics*, J. Comput. Phys., 99 (1992), pp. 341–347.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 285–294, March 1994

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.

COMPUTING LARGE SPARSE JACOBIAN MATRICES USING AUTOMATIC DIFFERENTIATION*

BRETT M. AVERICK†‡, JORGE J. MORÉ‡, CHRISTIAN H. BISCHOF‡, ALAN CARLE†§, AND ANDREAS GRIEWANK‡

Abstract. The computation of large sparse Jacobian matrices is required in many important large-scale scientific problems. Three approaches to computing such matrices are considered: hand-coding, difference approximations, and automatic differentiation using the ADIFOR (automatic differentiation in Fortran) tool. The authors compare the numerical reliability and computational efficiency of these approaches on applications from the MINPACK-2 test problem collection. The conclusion is that ADIFOR is the method of choice, leading to results that are as accurate as hand-coded derivatives, while at the same time outperforming difference approximations in both accuracy and speed.

Key words. optimization, derivatives, Jacobian, automatic differentiation, function differences, sparsity, large-scale, sensitivity analysis, nonlinear systems, ADIFOR

AMS subject classifications. 65-04, 39-04, 49-04, 65-05, 68-04, 68N20, 68Q52

1. Introduction. The solution of large-scale nonlinear problems often requires the computation of the Jacobian matrix $f'(x)$ of a mapping $f : \mathbb{R}^n \to \mathbb{R}^m$. This computation is required, for example, in constrained optimization, parameter identification, sensitivity analysis, and the solution of systems of stiff differential and algebraic equations. In this paper, we consider three approaches to computing large, sparse Jacobian matrices: function differences, hand coding, and automatic differentiation.

The most popular approach is to use *function differences* (FD) to approximate the Jacobian matrix. The ith column of $f'(x)$ can be approximated by first-order accurate *forward differences* and by second-order accurate *central differences*,

$$\frac{f(x + h_i e_i) - f(x)}{h_i}, \qquad \frac{f(x + h_i e_i) - f(x - h_i e_i)}{2h_i},$$

respectively, where h_i is a suitably chosen parameter and e_i is the ith unit vector. Computing derivatives by differences has the advantage that only the function is needed as a black box; however, the accuracy of such derivative approximations is hard to assess. The choice of the difference parameter h_i can be a source of difficulty for many problems, in particular if the problem is highly nonlinear or if the function is noisy. A small stepsize h_i is needed to suitably approximate the derivatives, yet it may lead to numerical cancellation and the loss of accuracy.

The potential inaccuracies of difference approximations can be avoided if one is able to supply derivative code. One approach is to *hand code* a subroutine to evaluate $f'(x)$.

*Received by the editors March 11, 1993; accepted for publication (in revised form) August 9, 1993. This work was supported by the Office of Scientific Computing, U.S. Department of Energy, Contract W-31-109-Eng-38.

†This work was supported in part by the Center for Research on Parallel Computation, National Science Foundation Cooperative Agreement CCR-8809615.

‡Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois 60439.

§Center for Research on Parallel Computation, Rice University, P.O. Box 1892, Houston, Texas 77251.

In addition to being accurate, hand coding usually produces efficient code. However, this approach is often time consuming and error prone, especially if the function is complicated. In particular, new coding effort is required whenever the function is modified.

Another way to obtain Jacobian matrices is by using symbolic manipulation packages such as Maple, Reduce, Macsyma, or Mathematica. Given a string describing the definition of a function, symbolic manipulation packages provide algebraic expressions for derivatives expressed in terms of the independent variables. Symbolic differentiation is a powerful technique, but quickly runs into resource limitations when applied to even moderately sized problems (described by 100 lines of code, say). Hence we will not consider it further.

Still another way to obtain derivative code is through *automatic differentiation* (AD). AD techniques rely on the fact that every function, no matter how complicated, is executed on a computer as a (potentially very long) sequence of elementary operations such as additions, multiplications, and elementary functions such as the trigonometric and exponential functions. By applying the chain rule to the composition of those elementary operations, one can compute derivative information for f exactly and in a completely mechanical fashion. The perceived disadvantage of AD techniques is that they are not able to handle large problems. In particular, there seems to be a perception that AD techniques are not able to compute large sparse Jacobian matrices either accurately or efficiently. The main purpose of this paper is to dispel this perception.

In §§2 and 3, we review three approaches to computing sparse Jacobian matrices: hand coding, difference approximations, and automatic differentiation using the ADIFOR tool. The rest of the paper uses several of the MINPACK-2 test problems to compare these approaches. The test problems are described in §4, and computational results are presented in §§5 and 6. We conclude that ADIFOR is the method of choice, leading to results that are as accurate as hand-coded derivatives, while at the same time outperforming difference approximations in both accuracy and speed.

2. Computing sparse Jacobian matrices. Computing the Jacobian matrix $f'(x)$ of a mapping $f : \mathbb{R}^n \to \mathbb{R}^m$ can be a difficult task when $f'(x)$ is large and sparse, particularly if a sparse data structure is used. One potential difficulty lies in the need to place elements of $f'(x)$ into the correct positions of the sparse data structure once they have been computed. For many problems, however, it is relatively easy to develop code for evaluating the Jacobian-vector product $f'(x)v$ for any $x, v \in \mathbb{R}^n$ since there is no dependence on data structure. In this section, we discuss the compressed Jacobian technique for computing sparse Jacobian matrices.

We assume that code is available for evaluating the Jacobian-vector product. It is often possible to write such code by hand, but as we shall see in the next section, it is also possible to generate this code with AD tools. If $f'(x)v$ code is not available, the Jacobian-vector product can be approximated by

$$f'(x)v \approx \frac{f(x + h_v v) - f(x)}{h_v}$$

for some difference parameter h_v.

Each column of $f'(x)$ can be obtained by choosing v to be the corresponding Cartesian basis vector. This can be extremely inefficient as it requires the computation of n Jacobian-vector products or in the case of a difference approximation, n function evaluations. To avoid this inefficiency, we *partition* the columns of $f'(x)$ into groups such that columns in a group do not have nonzeros in the same row position. For example, if a function $f : \mathbb{R}^4 \to \mathbb{R}^5$ has a

Jacobian matrix $f'(x)$ with the structure (symbols denote nonzeros, and zeros are not shown)

$$f'(x) = \begin{pmatrix} \bigcirc & & & \\ \bigcirc & & & \diamond \\ & \triangle & & \diamond \\ & \triangle & \square & \\ & \triangle & \square & \end{pmatrix},$$

then columns 1 and 2 can be placed in one group, while columns 3 and 4 can be placed in another group. This partitioning identifies *structurally orthogonal* columns of $f'(x)$, that is, columns whose inner product is zero, independent of x.

Given a partitioning of $f'(x)$ into p groups, each group consisting of structurally orthogonal columns, we can determine $f'(x)$ with p evaluations of $f'(x)v$. For each group we compute $f'(x)v$, where $v_i = 1$ if the i-th column is in the group, and $v_i = 0$ otherwise. In the above example, we would compute $f'(x)v_i$ for $v_1 = e_1 + e_2$ and $v_2 = e_3 + e_4$, and obtain

$$f'(x)v_1 = \begin{pmatrix} \bigcirc \\ \bigcirc \\ \triangle \\ \triangle \\ \triangle \end{pmatrix}, \qquad f'(x)v_2 = \begin{pmatrix} \diamond \\ \diamond \\ \square \\ \square \end{pmatrix},$$

at the cost of only two evaluations of $f'(x)v$ (versus four for the naive approach). Because of the structural orthogonality property, we can uniquely extract all entries of the Jacobian matrix from the Jacobian-vector products.

When Jacobian-vector products $f'(x)v_i$ are computed by hand-coded programs, it is usually advantageous to calculate them simultaneously to avoid the reevaluation of common expressions. This is the motivation for the *compressed Jacobian* approach where we assemble the vectors v_i into an $n \times p$ matrix V and compute $f'(x)V$. Clearly, the advantages of computing the compressed Jacobian tend to increase with p. Figure 2.1 illustrates the difference between the sparsity structure of $f'(x)$ and the sparsity structure of the compressed Jacobian $f'(x)V$ for the inverse elastic rod (IER) problem from the MINPACK-2 collection (see §4).

Curtis, Powell, and Reid (CPR) [11] were the first to note that a partitioning of the columns into p structurally orthogonal groups allows the approximation of the Jacobian matrix by function differences with p function evaluations. In the CPR algorithm, the groups are formed one at a time by scanning the columns in the natural order and including a column in the current group if it has not been included in a previous group and if it does not have a nonzero in the same row position as another column already in the group.

Coleman and Moré [10] showed that the partitioning problem could be analyzed as a *graph coloring* problem and that, by looking at the problem from the graph coloring point of view, it is possible to improve the CPR algorithm by scanning the columns in a carefully selected order. Coleman, Garbow, and Moré [9], [8] describe software for the partitioning problem. Given a representation of the sparsity structure of $f'(x)$, these algorithms produce a partitioning of the columns of $f'(x)$ into p structurally orthogonal groups. For many sparsity patterns, p is small and independent of n. For example, if the sparsity structure has bandwidth β, then $p \leq \beta$. We also note that discretization of an infinite-dimensional problem also leads to sparsity patterns where p is independent of the meshsize.

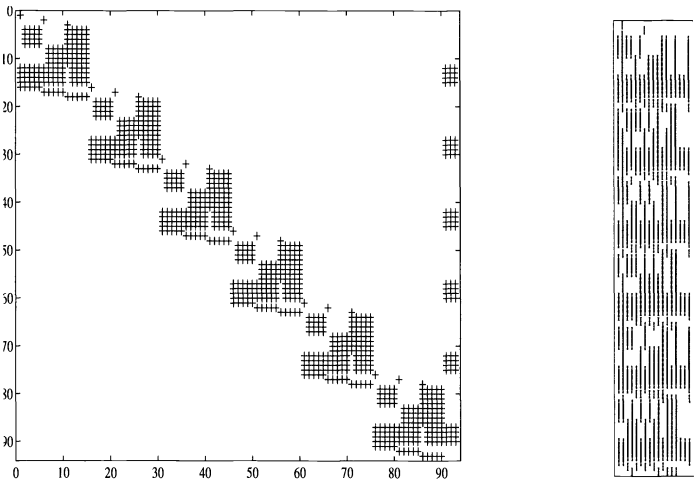


FIG. 2.1. IER *Jacobian sparsity (left) and* IER *compressed sparsity (right).*

3. AD and the ADIFOR tool. AD [17] is a chain rule-based technique for evaluating the derivatives of functions defined by computer programs with respect to their input variables. There are two basic modes of AD, which are usually referred to as *forward* and *reverse*. As discussed in [12], the reverse mode is closely related to adjoint methods and has a very low operation count for gradients. However, its potentially large memory requirement has been a serious impediment to its application in large-scale scientific computing. When there are several independent and dependent variables, the operation count for evaluating the Jacobian matrix may be lowest for certain mixed strategies [15] rather than for the forward or reverse mode. AD can also be extended to the accurate evaluation of second and higher derivatives [3], [7], [13]. A comprehensive collection on the theory, implementation, and some earlier applications can be found in [14].

In contrast to the approximation of derivatives by function differences, AD incurs no truncation error. Hence, at least for noniterative and branch-free codes, the resulting derivative values are usually obtained with the working accuracy of the original function evaluation. In contrast to fully symbolic differentiation, both operations count and storage requirement can be a priori bounded in terms of the complexity of the original function code.

The ADIFOR[1] tool [2], [4], [5] provides AD of programs written in Fortran 77. Given a Fortran subroutine (or collection of subroutines) describing a function, and an indication of which variables in parameter lists or common blocks correspond to independent and dependent variables with respect to differentiation, ADIFOR produces Fortran 77 code that allows the computation of the derivatives of the dependent variables with respect to the independent variables. ADIFOR employs a hybrid of the forward and reverse modes of AD. The resulting decrease in complexity compared with an implementation based entirely on the forward mode is usually substantial.

In contrast to some earlier AD implementations [16], the source translator ADIFOR was designed from the outset with large-scale codes in mind. The facilities of the ParaScope Fortran environment [6] control flow and data dependence flow information. ADIFOR produces portable Fortran 77 code and accepts almost all of Fortran 77; in particular, arbitrary calling sequences, nested subroutines, common blocks, and equivalences. The ADIFOR-generated

[1]For questions and information about ADIFOR e-mail `adifor-request@mcs.anl.gov`.

code tries to preserve vectorization and parallelism in the original code. It also employs a consistent subroutine naming scheme that allows for code tuning, the use of domain-specific knowledge, and the exploitation of vendor-supplied libraries. It should be stressed that ADIFOR uses data flow analysis information to determine the set of variables that require derivative information in addition to the dependent and independent ones. This approach allows for an intuitive interface and greatly reduces the storage requirements of the derivative code.

ADIFOR produces code to compute the Jacobian-vector product $f'(x)v$ or the compressed Jacobian matrix $f'(x)V$. The directional derivative $f'(x)v$ can be evaluated without forming the Jacobian matrix explicitly and thus potentially at a much lower computational cost. The operations count for this calculation is bounded by three times that of f; actual run time ratios vary depending on the implementation, the computing platform, and the nature of f. The compressed Jacobian matrices can be computed by exploiting the same graph coloring techniques discussed in §2. As already pointed out, the advantages of computing the compressed Jacobian matrix increase with p because it allows the reuse of expressions. ADIFOR-generated code reuses these expressions automatically; this is not possible with difference approximations and may not be done in a hand-coded subroutine to evaluate $f'(x)V$.

4. Test problems. In this section, we describe the problems used in comparing the different approaches for computing a sparse Jacobian matrix. The descriptions are brief since the problems are part of the MINPACK-2 test problem collection[2]; the current version of this collection is described by Averick, Carter, Moré, and Xue [1]. For each problem, the MINPACK-2 test collection contains subroutines that define the sparsity pattern of $f'(x)$ and evaluate $f(x)$ and $f'(x)V$ for any $V \in \mathbb{R}^{n\times p}$.

These problems are representative of computational problems found in applications. Collocation and finite differences are used to discretize these problems so as to obtain systems of nonlinear equations $f(x) = 0$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear mapping with a sparse Jacobian matrix $f'(x)$.

Flow in a channel (FIC). The analysis of fluid injection through one side of a long vertical channel leads to the boundary value problem

$$\begin{aligned} u'''' &= R[u'u'' - uu'''], \qquad 0 \le t \le 1, \\ u(0) = u'(0) &= 0, \quad u(1) = 1, \quad u'(1) = 0, \end{aligned}$$

where u is the potential function, u' is the tangential velocity, and R is the Reynolds number of the fluid.

Discretization of this problem by a k-stage collocation method, with $k = 4$, leads to a system of $n = 8n_h$ equations, where n_h is the number of subintervals in the collocation scheme. In this problem there is a maximum of nine nonzeros per row, independent of n_h.

Swirling flow between disks (SFD). The analysis of steady flow of a viscous, incompressible, axisymmetric fluid between two rotating, infinite coaxial disks, located at $t = 0$ and $t = 1$, yields the boundary value problem

$$\begin{aligned} \epsilon f'''' + ff''' + gg' = 0, \quad \epsilon g'' + fg' + f'g = 0, \qquad 0 \le t \le 1, \\ f(0) = f'(0) = f(1) = f'(1) = 0, \quad g(0) = \Omega_0, \quad g(1) = \Omega_1, \end{aligned}$$

where f' is radial velocity, g is angular velocity (Ω_0 and Ω_1 are the angular velocities of the infinite disks), and $0 \le \epsilon \ll 1$ is a viscosity parameter.

[2]Software for the MINPACK-2 test problem collection is available by anonymous ftp from `info.mcs.anl.gov`; current contents are in the file `pub/MINPACK-2/README`.

Discretization of this problem by a k-stage collocation method, with $k = 4$, leads to a system of $n = 14n_h$ equations, where n_h is the number of subintervals in the collocation scheme. In this problem there is a maximum of 14 nonzeros per row, independent of n_h.

Incompressible elastic rods (IER). The shape of a thin incompressible elastic rod, clamped at the origin and acted on by a vertical force Q, a horizontal force P, and torque M is described by the solution of the boundary value problem

$$\theta'(s) = Qx(s) - Py(s) + M, \quad x'(s) = \cos[\theta(s)], \quad y'(s) = \sin[\theta(s)],$$
$$x(0) = y(0) = \theta(0) = 0,$$

where θ is the local angle of inclination, and s is the arc length along the rod. We need to determine Q, P, and M such that $x(\cdot)$, $y(\cdot)$, and $\theta(\cdot)$ solve this boundary value problem and satisfy the boundary conditions $x(1) = a$, $y(1) = b$, $\theta(1) = c$.

Discretization by a k-stage collocation method, with $k = 4$, leads to a system of $n = 15n_h + 3$ equations, where n_h is the number of subintervals in the collocation scheme. The sparsity structure of the Jacobian and compressed Jacobian matrix for $n_h = 6$ are shown in Figure 2.1. The nonzeros in the last three columns correspond to the variables Q, P, and M. In this problem there is a maximum of 17 nonzeros per row, independent of n_h. Since there are 17 columns in the compressed Jacobian matrix, the coloring is optimal.

Solid fuel ignition (SFI). This problem arises in the analysis of a thermal reaction process dependent upon a balance between chemically generated heat addition and heat transfer by conduction in a rigid material. A steady-state model of this process can be described in terms of the solution u_λ of the boundary value problem

$$-\Delta u(x) = \lambda \exp[u(x)], \quad x \in \Omega, \qquad u(x) = 0, \quad x \in \partial\Omega,$$

where Δ is the Laplace operator, Ω is a domain in $\mathbb{R}^2$ with boundary $\partial\Omega$, and $\lambda \in \mathbb{R}$.

Discretization of this problem by finite differences on the unit square leads to a system of $n = n_x n_y$ equations, where n_x and n_y are the number of interior gridpoints in the coordinate directions, respectively. For this problem there is a maximum of five nonzeros per row, independent of n_x and n_y.

Flow in a driven cavity (FDC). The steady flow of a viscous incompressible fluid in a planar region Ω is described in terms of a stream function by the boundary value problem

$$\Delta^2\psi - R\Big[(\partial_y\psi)(\partial_x\Delta\psi) - (\partial_x\psi)(\partial_y\Delta\psi)\Big] = 0,$$
$$\psi(\xi_1, \xi_2) = \partial_x\psi(\xi_1, \xi_2) = 0, \qquad \partial_y\psi(\xi_1, \xi_2) = \begin{cases} 1 & \text{if } \xi_2 = 1, \\ 0 & \text{if } 0 \le \xi_2 < 1. \end{cases}$$

Discretization by finite differences on the unit square leads to a system of $n = n_x n_y$ equations, where n_x and n_y are the number of interior grid points in the coordinate directions, respectively. The Jacobian matrix has a maximum of 13 nonzeros per row independent of n_x and n_y.

5. Accuracy. We first compare the accuracy of the Jacobian matrix produced by ADIFOR, with the accuracy of the function difference approximation. As the standard, we take the Jacobian matrix included with the MINPACK-2 test problem collection. We measure both the absolute error and the relative error,

$$\max_{ij} \left\{ |\partial_{i,j} f(x) - a_{ij}(x)| \right\}, \qquad \max_{ij} \left\{ \frac{|\partial_{i,j} f(x) - a_{ij}(x)|}{\max\{|\partial_{i,j} f(x)|, |a_{ij}(x)|\}} \right\},$$

respectively, where $\partial_{i,j} f(x)$ is the derivative produced by the MINPACK-2 software, and $a_{i,j}(x)$ is either the difference approximation or the ADIFOR Jacobian matrix.

The results presented in Table 5.1 were obtained on a Solbourne 5E/902 in double-precision IEEE arithmetic. For these results, we evaluated the Jacobian matrices at a random vector with elements in the interval [0,1] and used a value of $h_i = 10^{-8}$ for all the variables. We do not claim that this choice is optimal, but for these problems, this choice produces reasonably accurate results. In general, the accuracy of an approximation with differences of function values depends on the choice of the difference parameters h_i, but even with an optimal choice of difference parameter, we can expect a difference approximation that is accurate only up to half the number of possible significant digits.

TABLE 5.1
Accuracy of ADIFOR *and function differences.*

		Absolute Error		Relative Error	
Problem	N	FD	ADIFOR	FD	ADIFOR
FIC	96	2.9×10^{-6}	1.4×10^{-14}	5.6×10^{-1}	9.5×10^{-16}
SFD	98	2.5×10^{-8}	0.0	1.0	0.0
IER	93	4.4×10^{-8}	0.0	1.6×10^{-1}	0.0
SFI	100	9.7×10^{-8}	0.0	3.8×10^{-8}	0.0
FDC	100	3.2×10^{-6}	3.5×10^{-14}	5.6×10^{-6}	5.4×10^{-14}

The results in Table 5.1 clearly indicate the superior accuracy of ADIFOR as compared with function differences. In terms of absolute error, ADIFOR and hand-coded Jacobians agree up to 16 significant digits, while function differences offer at most half that accuracy.

The same type of observation can be made for the relative error. However, relative error can be misleading if the Jacobian elements are sufficiently small. For FIC and IER, the relative error is usually of order 10^{-7}, but for small Jacobian entries, it may be of order 10^{-1}. In the case of SFD, function differences show a relative error of one. This is due to entries where the derivative value is of order 10^{-13} but function differences yield zero. Note, however, that ADIFOR accurately computes these small elements.

6. Timing. We now compare the time required to compute a sparse Jacobian matrix by all three approaches. The timing information was obtained for double-precision computations on a Solbourne 5E/902 and an IBM RS6000/580 workstation. Since we are interested in large problems, we used problems with n variables where n ranges between 14, 000 and 160, 000. Similar results were obtained on smaller problems.

Figure 6.1 shows the ratio of the run times for difference approximations to the Jacobian matrix to ADIFOR-generated derivative code. These results clearly show that the ADIFOR-generated code is faster than the difference approximations in all cases and that the performance advantage is more pronounced on the IBM than on the Solbourne. This can be explained by noting that the Solbourne has a true scalar processor, whereas the IBM employs a pipelined superscalar chip that performs well on the vector operations that constitute the bulk of the ADIFOR-generated derivative code.

In general, we expect ADIFOR to perform best if the number of groups p is large because the advantages of computing the compressed Jacobian matrix increase with p. This is borne out by our results. The performance of ADIFOR for small p tends to depend on the particular problem.

Note that the run time ratios in Figure 6.1 are independent of n. This can be explained by noting that the run time is proportional to the number of groups p associated with the sparsity structure; for these problems, p does not depend on the size n of the Jacobian matrix.

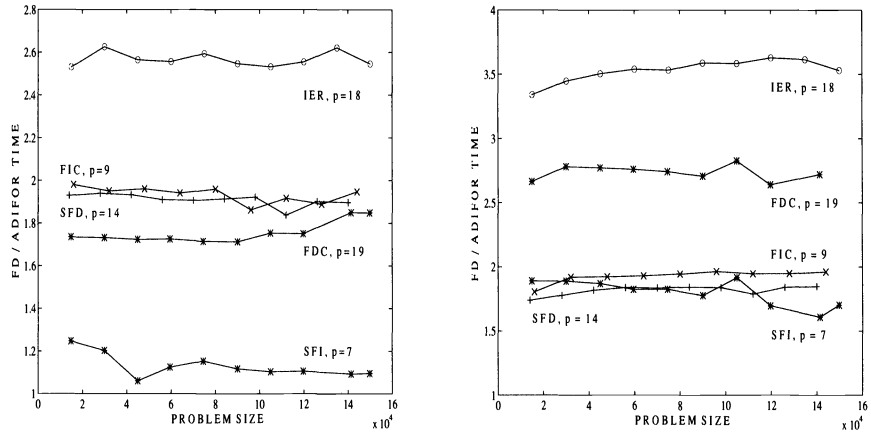


FIG. 6.1. FD *vs.* ADIFOR *timings on the Solbourne (left) and the* IBM *(right).*

The timing comparison of the ADIFOR-generated code with the hand-coded Jacobian matrix appears in Figure 6.2. We see that the ADIFOR-generated Jacobian code performs somewhat worse than the hand-coded Jacobian matrix, but not by a margin of more than roughly 2.5.

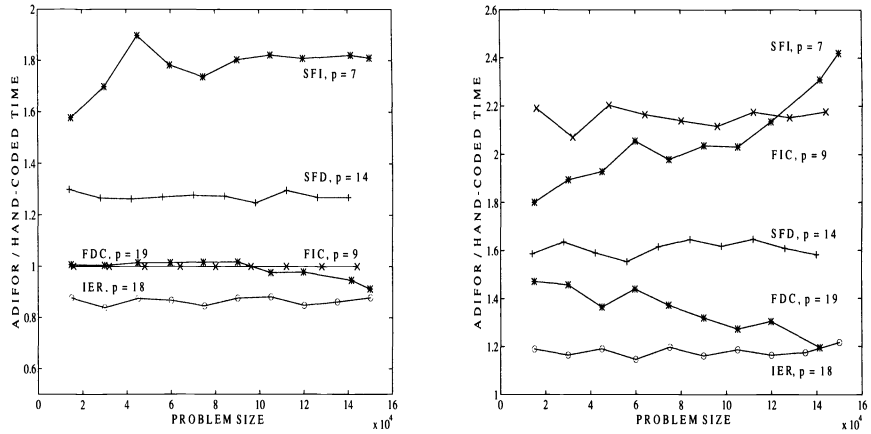


FIG. 6.2. ADIFOR *vs. hand-code timings on the Solbourne (left) and the* IBM *(right).*

Last, Table 6.1 puts these these timings in perspective by comparing the time for computing the Jacobian matrices (FD, ADIFOR, and MINPACK) with the time required to partition the columns into structurally orthogonal groups (DSM), and the time required to convert the compressed Jacobian matrix into a sparse matrix format (FDJS). Subroutines for these tasks are described by Coleman, Garbow, and Moré [8], [9]. Subroutine DSM takes the sparsity pattern of the Jacobian matrix and produces a partitioning of the columns of $f'(x)$ into p structurally orthogonal groups, while subroutine FDJS converts the compressed Jacobian matrix into a sparse matrix format. Columns N, NNZ, and P show the dimension, number of nonzeros in the Jacobian, and number of groups, respectively, for each of the problems.

Table 6.1 shows that the time required by DSM is significant when compared with the time required to compute a Jacobian matrix by either ADIFOR or the hand-coded MINPACK

subroutines. This is justified for most problems because DSM needs to be called only once for each sparsity pattern. Moreover, the runtime of DSM only depends on the sparsity pattern, not on the expense of evaluating the function or Jacobian matrix.

TABLE 6.1
Detailed Solbourne timings.

				Time (seconds)				
Problem	N	NNZ	P	DSM	FD	ADIFOR	MINPACK	FDJS
SFI	14884	73932	7	2.38	1.35	0.93	0.59	.19
FIC	16000	123987	9	2.62	5.23	2.55	1.46	.35
SFD	14000	154981	14	3.79	5.48	2.69	2.08	.43
IER	15003	158000	18	6.14	7.43	2.77	3.16	.48
FDC	14884	191056	19	9.11	11.71	6.53	6.50	.54

We also note that the runtime of FDJS is small compared with the other times in Table 6.1 and is proportional to the number of nonzeros in the Jacobian. This was to be expected because FDJS converts the compressed Jacobian matrix into a column-oriented sparse matrix format, but does not perform any arithmetic operations. We also note that the time required by FDJS becomes less significant as the number of groups p increases.

7. Conclusions. We conclude that ADIFOR-derived derivatives are certainly superior to difference approximations. Not only does the AD approach not suffer from truncation error, but its ADIFOR incarnation delivers code that outperforms divided differences by a factor of up to 3.5. The other attraction of ADIFOR is that one can generate derivative code at the touch of a button, whereas the development of a derivative code by hand is tedious and time consuming. Whether the effort involved in computing a Jacobian by hand is worth the modest speedup that we have observed here obviously depends on the application, but from our perspective it is not.

Acknowledgments. We would like to thank Richard Carter for many stimulating discussions and Jack Dongarra for the use of the IBM RS-6000.

REFERENCES

[1] B. M. AVERICK, R. G. CARTER, J. J. MORÉ, AND G.-L. XUE, *The MINPACK-2 test problem collection*, MCS-P153-0692, Argonne National Laboratory, Argonne, IL, 1992, preprint.

[2] C. BISCHOF, A. CARLE, G. CORLISS, A. GRIEWANK, AND P. HOVLAND, *ADIFOR: Generating derivative codes from Fortran programs*, Scientific Programming, 1 (1992), pp. 1–29.

[3] C. BISCHOF, G. CORLISS, AND A. GRIEWANK, *Computing second- and higher-order derivatives through univariate Taylor series*, MCS-P296-0392, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1992, preprint.

[4] C. BISCHOF AND A. GRIEWANK, *ADIFOR: A Fortran system for portable automatic differentiation*, in Proc. 4th Symposium on Multidisciplinary Analysis and Optimization, AIAA Paper 92-4744, American Institute of Aeronautics and Astronautics, 1992, pp. 433–441.

[5] C. BISCHOF AND P. HOVLAND, *Using ADIFOR to compute dense and sparse Jacobians*, Tech. Rep. MCS-TM-158, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1991.

[6] A. CARLE, K. D. COOPER, R. T. HOOD, K. KENNEDY, L. TORCZON, AND S. K. WARREN, *A practical environment for scientific programming*, IEEE Computer, 20 (1987), pp. 75–89.

[7] B. CHRISTIANSON, *Reverse accumulation and accurate rounding error estimates for Taylor series coefficients*, Optimization Methods Software, 1 (1992), pp. 81–94.

[8] T. F. COLEMAN, B. S. GARBOW, AND J. J. MORÉ, *Fortran subroutines for estimating sparse Jacobian matrices*, ACM Trans. Math. Software, 10 (1984), pp. 346–347.

[9] ———, *Software for estimating sparse Jacobian matrices*, ACM Trans. Math. Software, 10 (1984), pp. 329–345.

[10] T. F. COLEMAN AND J. J. MORÉ, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[11] A. R. CURTIS, M. J. D. POWELL, AND J. K. REID, *On the estimation of sparse Jacobian matrices*, J. Instute Math. Appl., 13 (1974), pp. 117–119.

[12] A. GRIEWANK, *On automatic differentiation*, in Mathematical Programming: Recent Developments and Applications, M. Iri and K. Tanabe, eds., Kluwer Academic Publishers, Norwell, MA, 1989, pp. 83–108.

[13] ———, *Automatic evaluation of first- and higher-derivative vectors*, in Proc. Conf. at Würzburg, Aug. 1990, Bifurcation and Chaos: Analysis, Algorithms, Applications, R. Seydel, F. W. Schneider, T. Küpper, and H. Troger, eds., Vol. 97, Birkhäuser-Verlag, Basel, Switzerland, 1991, pp. 135–148.

[14] A. GRIEWANK AND G. F. CORLISS, EDS., *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[15] A. GRIEWANK AND S. REESE, *On the calculation of Jacobian matrices by the Markowitz rule*, in Automatic Differentiation of Algorithms: Theory, Implementation, and Application, A. Griewank and G. Corliss, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 126–135.

[16] D. JUEDES, *A taxonomy of automatic differentiation tools*, in Automatic Differentiation of Algorithms: Theory, Implementation, and Application, A. Griewank and G. Corliss, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 315–329.

[17] L. B. RALL, *Automatic Differentiation: Techniques and Applications*, Vol. 120, Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1981.

SPECIAL SECTION ON ITERATIVE METHODS IN NUMERICAL LINEAR ALGEBRA

The 1992 Copper Mountain Conference on Iterative Methods in Numerical Linear Algebra was the seventh conference in the Copper Mountain series and the second devoted to general iterative methods. It was attended by approximately 190 mathematicians from all corners of the world. The meeting, held April 9–14, 1992, took place at the Copper Mountain Resort, which is located 70 miles west of Denver, in the heart of Colorado's famous Summit County ski region. The setting of the conference was a cluster of buildings nestled at the base of the resort's ski hill. Morning and evening sessions were scheduled, leaving afternoons open for informal discussions and recreation.

During the five days of the meeting, 109 talks on current research were presented. These talks were organized by content and grouped into sessions. Session topics included the Theory of Iterative Methods (two sessions), Nonsymmetric Systems (four), Preconditioning Strategies (two), Parallel Implementations (three), Applications (four), Multigrid and Multilevel Methods (three), Domain Decomposition (two), Eigenvalue Problems (two), Integral Equations, Nonlinear Systems (two), Indefinite and Complex Matrix Problems, Collocation Matrices, and Software. In addition, there were workshops devoted to Scientific Computing in C^{++} and Common Software Standards.

On the first day of the conference each participant was provided with a two-volume set of Preliminary Proceedings, which contained either a short paper or an abstract by each of the scheduled speakers. Of those, 49 speakers submitted complete papers to this special issue of the *SIAM Journal on Scientific Computing* (*SISC*). This and the next issue of *SISC* contain the results of the refereeing process. They represent a rich mix of papers on a wide variety of topics related to iterative methods.

The following twelve papers represent the conference. The first three deal with general iterative algorithms. The second and third involve quasi-minimal residual algorithms, a subject that inspired much discussion at the meeting. Papers 4 and 5 analyze interesting theoretical aspects of iterative methods. Papers 6 through 9 deal with precondition strategies. Papers 10 through 12 explore the implications for iterative methods of parallel computing environments.

In the next issue the last twelve papers again represent the conference. The first seven of these deal with multigrid or multilevel algorithms. The first of these is a very interesting paper that uses the tools of classical iterative methods to analyze and compare many well-known algorithms. The second and third papers also contribute to the development of the theory of multigrid and multilevel algorithms. Papers 4 through 6 are concerned with domain decomposition methods, which, incidentally, fall into the structure developed in the first paper. The final six papers in this group discuss iterative methods in the context of specific applications and represent the many fine application-specific talks presented at the conference.

A special effort was made to bring students to the meeting. The vehicle for this effort was a Student Paper Competition in which students were asked to submit an original research paper consisting primarily of their own work. We were fortunate enough to be able to provide all student authors participating in this competition with free lodging and registration. Out of thirteen submissions, three winners were selected. First place went to Xian-Zhong Guo from the University of Maryland for his work, *"The Algebraic Hierarchical Basis Multigrid Method."* Second place was awarded to Marlis Hochbruck from the University of Karlsruhe, Germany, for her paper, *"On the Use of Two QMR Algorithms for Solving Singular Systems and Applications in the Markov Chain Modeling."* Third place honors were awarded to Andrew Lumsdaine from MIT for the paper, *"Accelerating Dynamic Iteration Methods with Application to Semiconductor Device Simulation."* All winners were awarded a travel stipend,

and their papers were presented at a special session devoted to the Student Paper Competition.

We would like to thank the following members of the program committee for their help in editing this special issue. They are: Steve Ashby, Howard Elman, Roland Freund, Anne Greenbaum, Seymour Parter, Paul Saylor, Homer Walker, and Olof Widlund. Through their efforts, the articles contained in this special issue were carefully refereed and brought to print on schedule.

We would also 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.

As this issue goes to press, planning for the next conference in this series is in its final stages. It will be called the Colorado Conference on Iterative Methods and will be held April 4–9, 1994, in Breckenridge, Colorado. Plans again include a special journal issue in *SISC*. It is our hope that the lively interaction and the fine quality of presentations and papers that have marked the previous meetings can be duplicated at the upcoming meeting.

Conference Chairmen
Tom Manteuffel
Steve McCormick

Special Issue Editors
Steve Ashby
Howard Elman
Roland Freund
Anne Greenbaum
Seymour Parter
Paul Saylor
Homer Walker
Olof Widlund

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 297–312, March 1994

004

RESIDUAL SMOOTHING TECHNIQUES FOR ITERATIVE METHODS*

LU ZHOU† AND HOMER F. WALKER†

Abstract. An iterative method for solving a linear system $Ax = b$ produces iterates $\{x_k\}$ with associated residual norms that, in general, need not decrease "smoothly" to zero. "Residual smoothing" techniques are considered that generate a second sequence $\{y_k\}$ via a simple relation $y_k = (1-\eta_k)y_{k-1}+\eta_k x_k$. The authors first review and comment on a technique of this form introduced by Schönauer and Weiss that results in $\{y_k\}$ with monotone decreasing residual norms; this is referred to as *minimal residual smoothing*. Certain relationships between the residuals and residual norms of the biconjugate gradient (BCG) and quasi-minimal residual (QMR) methods are then noted, from which it follows that QMR can be obtained from BCG by a technique of this form; this technique is extended to generally applicable quasi-minimal residual smoothing. The practical performance of these techniques is illustrated in a number of numerical experiments.

Key words. iterative linear algebra methods, Krylov subspace methods, residual smoothing methods, GMRES, CGS, Bi-CGSTAB methods, QMR methods

AMS subject classification. 65F10

1. Introduction.

In recent years, there has been a great deal of interest in iterative methods for solving a general nonsymmetric linear system

$$Ax = b,$$

where $A \in \mathbb{R}^{n\times n}$ and $x, b \in \mathbb{R}^n$. The quality of the iterates $\{x_k\}$ produced by a method is often judged by the behavior of $\{\|r_k\|\}$, where $r_k = b - Ax_k$; in particular, it is usually desirable that $\{\|r_k\|\}$ converge "smoothly" to zero.

In the widely used generalized minimal residual (GMRES) method [13], each x_k is characterized by

$$\|b - Ax_k\|_2 = \min_{x\in x_0+\mathcal{K}_k(r_0,A)} \|b - Ax\|_2,$$

where $\|\cdot\|_2$ is the Euclidean norm and the *Krylov subspace* $\mathcal{K}_k(r_0, A)$ is defined by

$$\mathcal{K}_k(r_0, A) = \text{span}\left\{r_0, Ar_0, \ldots, A^{k-1}r_0\right\}.$$

For GMRES, then, $\{\|r_k\|_2\}$ converges to zero optimally among all *Krylov subspace methods*, for which $x_k \in x_0 + \mathcal{K}_k(r_0, A)$. Other methods, such as biconjugate gradient (BCG) [12], [4] and conjugate gradient squared (CGS) [15], have certain advantages over GMRES but often exhibit very irregular residual-norm behavior. This irregular behavior has provided an incentive for the development of methods that have similar advantages but produce better behaved residual norms, such as the biconjugate gradient stabilized (Bi-CGSTAB) methods [10], [17] and methods based on the quasi-minimal residual (QMR) approach [2], [3], [6]–[8].

Another approach to generating well-behaved residual norms has been pursued in [14] and [18]. In this approach, an auxiliary sequence $\{y_k\}$ is generated from $\{x_k\}$ by a relation

$$\begin{aligned} &y_0 = x_0, \\ &y_k = (1 - \eta_k)y_{k-1} + \eta_k x_k, \qquad k = 1, 2, \ldots, \end{aligned} \tag{1.1}$$

*Received by the editors May 18, 1992; accepted for publication (in revised form) December 27, 1992. This work was supported in part by United States Air Force Office of Scientific Research grant AFOSR-91-0294 and United States Department of Energy grant DE-FG02-92ER25136, both with Utah State University.

†Department of Mathematics and Statistics, Utah State University, Logan, Utah 84322-3900.

in which each η_k is chosen to minimize $\|b - A((1-\eta)y_{k-1} + \eta x_k)\|_2$ over $\eta \in \mathbb{R}$, i.e.,

$$\eta_k = -\frac{s_{k-1}^T(r_k - s_{k-1})}{\|r_k - s_{k-1}\|_2^2}, \tag{1.2}$$

where $s_{k-1} = b - Ay_{k-1}$. (A weighted Euclidean norm is allowed in [18], but this is not important here.) The resulting residuals $s_k = b - Ay_k$ clearly have monotone decreasing Euclidean norms, and $\|s_k\|_2 \leq \|r_k\|_2$ for each k.

Our purpose here is to explore residual smoothing techniques of the form (1.1). In §2, we elaborate briefly on the particular technique of [14] and [18] described above, which we refer to here as *minimal residual smoothing* (MRS). In §3, we first note that the residuals and residual norms of the QMR and BCG methods are related in certain ways, from which it follows that QMR can be obtained from BCG by a technique of the form (1.1). We then extend this to a *quasi-minimal residual smoothing* (QMRS) technique applicable to any iterative method.[1] We describe a number of illustrative numerical experiments in §4 and discuss conclusions in §5.

A notational convention: When helpful, we denote iterates and residuals associated with a particular method by superscripts indicating that method.

2. Minimal residual smoothing. Assuming we have some iterative method that generates iterates $\{x_k\}$ and corresponding residuals $\{r_k\}$, we formulate the MRS technique of [14], [18] as follows:

ALGORITHM 2.1. Minimal residual smoothing [14], [18].

INITIALIZE: SET $s_0 = r_0$ AND $y_0 = x_0$.

ITERATE: FOR $k = 1, 2, \ldots,$ DO:

COMPUTE x_k AND r_k.
COMPUTE η_k by (1.2).
SET $s_k = s_{k-1} + \eta_k(r_k - s_{k-1})$ AND $y_k = y_{k-1} + \eta_k(x_k - y_{k-1})$.

There is a potential numerical difficulty with Algorithm 2.1. In practice, when the performance of an iterative method has become degraded through numerical error, the computed value of r_k can differ significantly from $b - Ax_k$. When this happens, the value of s_k computed in Algorithm 2.1 can differ significantly from $b - Ay_k$. Algorithm 2.2 below is a mathematically equivalent variation that does not suffer this difficulty provided an accurate value of Ap_k is available for each $p_k = x_k - x_{k-1}$, which is very often the case. In Algorithm 2.2, both s_k and y_k are determined directly from p_k and Ap_k; in particular, r_k is not involved in the computation at all. The intermediate quantities u_k and v_k are maintained so that, after the updating in the final step, we have $r_k = s_k - u_k$ and $x_k = y_k + v_k$. The comparative practical behavior of Algorithms 2.1 and 2.2 is illustrated in §4.1.

ALGORITHM 2.2. Minimal residual smoothing.

INITIALIZE: SET $s_0 = r_0$, $y_0 = x_0$, AND $u_0 = v_0 = 0$.

ITERATE: FOR $k = 1, 2, \ldots,$ DO:

COMPUTE $p_k = x_k - x_{k-1}$ AND Ap_k.
SET $u_k = u_{k-1} + Ap_k$ AND $v_k = v_{k-1} + p_k$.
COMPUTE $\eta_k = s_{k-1}^T u_k / u_k^T u_k$.
SET $s_k = s_{k-1} - \eta_k u_k$ AND $y_k = y_{k-1} + \eta_k v_k$.
UPDATE $u_k \leftarrow (1 - \eta_k)u_k$ AND $v_k \leftarrow (1 - \eta_k)v_k$.

[1] This should not be confused with the QMR squared method, designated QMRS in [8].

In MRS, each s_k is the vector of minimal Euclidean norm along the line containing s_{k-1} and r_k. There is an obvious extension, viz., inductively determining s_k as the vector of minimal Euclidean norm in the affine subspace containing s_{k-1} and $r_k, \ldots, r_{k-\ell}$ for some ℓ. This extension can be implemented reasonably economically (in $O(\ell n) + O(\ell^2)$ arithmetic operations for each k). However, there is no guarantee of improvement, and, in experiments that we carried out, it was not more effective than basic MRS in reducing the residual norm.

3. Quasi-minimal residual smoothing.

3.1. How QMR smoothes BCG. We first develop certain relationships between the QMR and BCG residuals and their norms. We recall the BCG method as follows:

ALGORITHM 3.1.1. Biconjugate gradient method [12], [4].

INITIALIZE:

CHOOSE x_0 AND SET $q_0 = r_0 = b - Ax_0$.
CHOOSE $\tilde{r}_0 \neq 0$, AND SET $\tilde{q}_0 = \tilde{r}_0$.

ITERATE: FOR $k = 1, 2, \ldots,$ DO:

COMPUTE $\delta_k = \tilde{r}_{k-1}^T r_{k-1} / \tilde{q}_{k-1}^T A q_{k-1}$ AND SET $x_k = x_{k-1} + \delta_k q_{k-1}$.
SET $r_k = r_{k-1} - \delta_k A q_{k-1}$ AND $\tilde{r}_k = \tilde{r}_{k-1} - \delta_k A^T \tilde{q}_{k-1}$.
COMPUTE $\gamma_k = \tilde{r}_k^T r_k / \tilde{r}_{k-1}^T r_{k-1}$.
SET $q_k = r_k + \gamma_k q_{k-1}$ AND $\tilde{q}_k = \tilde{r}_k + \gamma_k \tilde{q}_{k-1}$.

We consider the basic QMR method obtained from the general method in [7] by omitting diagonal scaling and using the classical nonsymmetric Lanczos process [11] *without* look-ahead. To discuss this, we note that $r_0, \ldots, r_{k-1}$ generated by Algorithm 3.1.1 clearly form a basis of $\mathcal{K}_k(r_0, A)$. Set $V_k = (v_1, \ldots, v_k)$ with $v_i = r_{i-1}/\rho_{i-1}$ and $\rho_{i-1} = \|r_{i-1}\|_2$ for each i. The columns of V_k also form a basis of $\mathcal{K}_k(r_0, A)$, and any $x \in x_0 + \mathcal{K}_k(r_0, A)$ can be written as

$$x = x_0 + V_k z, \qquad z \in \mathbb{R}^k. \tag{3.1.1}$$

By the nonsymmetric Lanczos process, we have

$$AV_k = V_{k+1} H_k, \tag{3.1.2}$$

where $H_k \in \mathbb{R}^{(k+1)\times k}$ is tridiagonal. From (3.1.2), the residual for x in (3.1.1) is

$$\begin{aligned} b - Ax &= r_0 - AV_k z = r_0 - V_{k+1} H_k z \\ &= V_{k+1}(\rho_0 e_1 - H_k z), \end{aligned} \tag{3.1.3}$$

where $e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{k+1}$. The kth QMR iterate is defined by $x_k^{\mathrm{QMR}} = x_0 + V_k z_k$, where z_k satisfies

$$\|\rho_0 e_1 - H_k z_k\|_2 = \tau_k \equiv \min_{z \in \mathbb{R}^k} \|\rho_0 e_1 - H_k z\|_2.$$

For later reference, we recall from [7] that

$$\|r_k^{\mathrm{QMR}}\|_2 \leq \sqrt{k+1}\, \tau_k. \tag{3.1.4}$$

In [7], the upper bound $\sqrt{k+1}\, \tau_k$ is used in a preliminary test for termination in QMR. We comment further on it below.

To develop the desired relationships between QMR and BCG, we note that $q_0, \ldots, q_{k-1}$ generated by Algorithm 3.1.1 also form a basis of $\mathcal{K}_k(r_0, A)$. Set

$$Y_k = (\delta_1 q_0, \ldots, \delta_k q_{k-1})$$

and

$$G_{k+1} = \begin{pmatrix} \rho_0 & & & \\ & \rho_1 & & \\ & & \ddots & \\ & & & \rho_k \end{pmatrix}.$$

We assume that Algorithm 3.1.1 can successfully carry out the kth step, in which case $\delta_1, \ldots, \delta_k$ are all nonzero and the columns of Y_k form a basis of $\mathcal{K}_k(r_0, A)$. From Algorithm 3.1.1, we have

$$AY_k = V_{k+1} G_{k+1} \bar{H}_k, \tag{3.1.5}$$

where

$$\bar{H}_k = \begin{pmatrix} 1 & & & \\ -1 & 1 & & \\ & \ddots & \ddots & \\ & & -1 & 1 \\ & & & -1 \end{pmatrix} \in \mathbb{R}^{(k+1)\times k}.$$

Since the columns of Y_k are a basis for $\mathcal{K}_k(r_0, A)$, we can write $x \in x_0 + \mathcal{K}_k(r_0, A)$ as $x = x_0 + Y_k \bar{z}$ for $\bar{z} \in \mathbb{R}^k$. Then from (3.1.5), we have

$$\begin{aligned} b - Ax &= r_0 - AY_k \bar{z} = r_0 - V_{k+1} G_{k+1} \bar{H}_k \bar{z} \\ &= V_{k+1} \left[G_{k+1} (e_1 - \bar{H}_k \bar{z}) \right]. \end{aligned} \tag{3.1.6}$$

Comparing (3.1.3) and (3.1.6), one easily verifies that

$$\tau_k = \min_{z \in \mathbb{R}^k} \| \rho_0 e_1 - H_k z \|_2 = \min_{\bar{z} \in \mathbb{R}^k} \| G_{k+1} (e_1 - \bar{H}_k \bar{z}) \|_2. \tag{3.1.7}$$

The least-squares problem on the right-hand side of (3.1.7) is easily solved. Setting $\bar{z} = (\zeta_1, \zeta_2, \ldots, \zeta_k)^T$ gives

$$\tau_k^2 = \min_{\bar{z} \in \mathbb{R}^k} \rho_0^2 (1 - \zeta_1)^2 + \rho_1^2 (\zeta_1 - \zeta_2)^2 + \cdots + \rho_{k-1}^2 (\zeta_{k-1} - \zeta_k)^2 + \rho_k^2 \zeta_k^2.$$

With a change of variables $\xi_0 = 1 - \zeta_1, \xi_1 = \zeta_1 - \zeta_2, \ldots, \xi_{k-1} = \zeta_{k-1} - \zeta_k, \xi_k = \zeta_k$, we have

$$\tau_k^2 = \min_{\sum_{i=0}^k \xi_i = 1} \sum_{i=0}^k \rho_i^2 \xi_i^2.$$

The unique minimizer is given by

$$\xi_i = \frac{\frac{1}{\rho_i^2}}{\sum_{j=0}^k \frac{1}{\rho_j^2}}, \quad i = 0, 1, \ldots, k,$$

and so

$$\tau_k = \sqrt{\frac{1}{\sum_{j=0}^{k} \frac{1}{\rho_j^2}}}.$$

Then the upper bound in (3.1.4) is

$$\sqrt{k+1}\,\tau_k = \sqrt{\frac{1}{\frac{1}{k+1}\sum_{j=0}^{k} \frac{1}{\rho_j^2}}}, \tag{3.1.8}$$

and it follows from (3.1.6) that the QMR residual vector is

$$r_k^{\mathrm{QMR}} = V_{k+1}(\rho_0\xi_0, \ldots, \rho_k\xi_k)^T = \frac{1}{\sum_{j=0}^{k} \frac{1}{\rho_j^2}} \sum_{i=0}^{k} \frac{1}{\rho_i^2} r_i^{\mathrm{BCG}}. \tag{3.1.9}$$

These results are of interest in their own right. We summarize them in the following theorem and then discuss some consequences.

THEOREM 3.1. *With* $\rho_i = \|r_i^{\mathrm{BCG}}\|_2$ *for* $i = 0, \ldots, k$, *the* QMR *residual* r_k^{QMR} *is given by* (3.1.9) *as a convex combination of the* BCG *residuals* $r_0^{\mathrm{BCG}}, \ldots, r_k^{\mathrm{BCG}}$. *The upper bound* $\sqrt{k+1}\,\tau_k$ *on* $\|r_k^{\mathrm{QMR}}\|_2$ *in* (3.1.4) *is given by* (3.1.8) *as the square root of the harmonic mean of* $\rho_0^2, \ldots, \rho_k^2$.

Equation (3.1.9) gives considerable insight into how QMR produces relatively smoothly decreasing (if not monotonically decreasing) residual norms. If $\|r_k^{\mathrm{BCG}}\|_2$ is small for some k, then r_k^{BCG} is given large weight in the convex combination (3.1.9) and $\|r_k^{\mathrm{QMR}}\|_2$ is small. If subsequent BCG residual norms increase, then the effect of this increase is mollified by relatively small weights in (3.1.9) and any increase in $\|r_k^{\mathrm{QMR}}\|_2$ is correspondingly small. These observations are borne out in Experiments 3 and 4 in §4.2 below, in which each QMR residual norm is roughly comparable to the best BCG residual norm obtained so far.

It follows from (3.1.8) that the upper bound $\sqrt{k+1}\,\tau_k$ on $\|r_k^{\mathrm{QMR}}\|_2$ is greater than or equal to $\min_{j=0,\ldots,k} \|r_j^{\mathrm{BCG}}\|_2$, with equality if and only if $\|r_0^{\mathrm{BCG}}\|_2 = \cdots = \|r_k^{\mathrm{BCG}}\|_2$. In fact, this upper bound can be a significant overestimate of $\|r_k^{\mathrm{QMR}}\|_2$, e.g., when A is nearly symmetric. Indeed, if A were symmetric, then the nonsymmetric Lanczos process would become the symmetric process, the vectors r_i^{BCG} would be mutually orthogonal, and we would have $\|r_k^{\mathrm{QMR}}\|_2 = \tau_k$ from (3.1.9).

From (3.1.9), we immediately obtain

$$\begin{aligned} r_k^{\mathrm{QMR}} &= \frac{\tau_k^2}{\tau_{k-1}^2} r_{k-1}^{\mathrm{QMR}} + \frac{\tau_k{}^2}{\rho_k^2} r_k^{\mathrm{BCG}}, \\ x_k^{\mathrm{QMR}} &= \frac{\tau_k^2}{\tau_{k-1}^2} x_{k-1}^{\mathrm{QMR}} + \frac{\tau_k^2}{\rho_k^2} x_k^{\mathrm{BCG}}. \end{aligned} \tag{3.1.10}$$

This has two useful consequences: First, since $\tau_k^2/\tau_{k-1}^2 + \tau_k^2/\rho_k^2 = 1$, we conclude that *the* QMR *method is obtained from the* BCG *method by a smoothing technique of the form* (1.1).[2] We generalize this technique in §3.2 below. Second, (3.1.10) gives a convenient and economical way of obtaining the QMR iterates and residuals from BCG. We note, however, that

[2]This is also implicit in results in [8] derived in a different manner. A different but equivalent way of determining the QMR iterates from BCG is also given in [8, §3].

if r_k^{BCG} and $b - Ax_k^{\mathrm{BCG}}$ differ significantly because of numerical error, then, as with Algorithm 2.1, r_k^{QMR} and $b - Ax_k^{\mathrm{QMR}}$ computed by (3.1.10) can also differ significantly. In this case, an analogue of Algorithm 2.2 obtained by applying the general Algorithm 3.2.2 below should perform better. In §4.2, we describe an experiment in which (3.1.10) performs poorly while the mathematically equivalent implementation of Algorithm 3.2.2 performs satisfactorily.

3.2. General quasi-minimal residual smoothing. Suppose we have some iterative method that generates iterates $\{x_k\}$ and corresponding residuals $\{r_k\}$. Since the difference of any two residuals is in the range of A, we can find for each $k \geq 1$ a w_k such that $r_{k-1} - r_k = Aw_k$. Define $Y_k = (w_1, \ldots, w_k)$ and $\rho_i = \|r_i\|_2$ for $i = 0, \ldots, k$. With V_{k+1}, G_{k+1}, and $\bar{H}_k$ as in §3.1, we again have (3.1.5), and so (3.1.6) holds for $x = x_0 + Y_k\bar{z}$, $\bar{z} \in \mathbb{R}^k$. It follows as before that

$$\tau_k \equiv \min_{\bar{z}\in\mathbb{R}^k} \|G_{k+1}(e_1 - \bar{H}_k\bar{z})\|_2 = \sqrt{\frac{1}{\sum_{j=0}^{k} \frac{1}{\rho_j^2}}}, \tag{3.2.1}$$

and, for the minimizing $\bar{z}$, we have, as in (3.1.9),

$$s_k \equiv V_{k+1}\left[G_{k+1}(e_1 - \bar{H}_k\bar{z})\right] = \frac{1}{\sum_{j=0}^{k} \frac{1}{\rho_j^2}} \sum_{i=0}^{k} \frac{1}{\rho_i^2} r_i. \tag{3.2.2}$$

In analogy with (3.1.10), we can use (3.2.2) to define "smoothed" residuals and iterates by

$$s_k = \frac{\tau_k^2}{\tau_{k-1}^2} s_{k-1} + \frac{\tau_k^2}{\rho_k^2} r_k, \qquad y_k = \frac{\tau_k^2}{\tau_{k-1}^2} y_{k-1} + \frac{\tau_k^2}{\rho_k^2} x_k,$$

and we can determine τ_k simply by $1/\tau_k^2 = 1/\tau_{k-1}^2 + 1/\rho_k^2$. This leads to the following algorithm.

ALGORITHM 3.2.1. Quasi-minimal residual smoothing.

INITIALIZE: SET $s_0 = r_0$, $y_0 = x_0$, AND $\tau_0 = \rho_0 = \|r_0\|_2$.

ITERATE: FOR $k = 1, 2, \ldots,$ DO:

COMPUTE x_k AND r_k.

SET $\rho_k = \|r_k\|_2$ AND DEFINE τ_k by $1/\tau_k^2 = 1/\tau_{k-1}^2 + 1/\rho_k^2$.

SET $s_k = (\tau_k^2/\tau_{k-1}^2)s_{k-1} + (\tau_k^2/\rho_k^2)r_k$ AND $y_k = (\tau_k^2/\tau_{k-1}^2)y_{k-1} + (\tau_k^2/\rho_k^2)x_k$.

In this algorithm, as in Algorithm 2.1, the divergence of r_k and $b - Ax_k$ through numerical error can cause s_k and $b - Ay_k$ to differ significantly. We formulate a mathematically equivalent algorithm analogous to Algorithm 2.2 that should avoid this difficulty provided an accurate value of Ap_k is available for each $p_k = x_k - x_{k-1}$.

ALGORITHM 3.2.2. Quasi-minimal residual smoothing.

INITIALIZE: SET $s_0 = r_0$, $y_0 = x_0$, $u_0 = v_0 = 0$, AND $\tau_0 = \rho_0 = \|r_0\|_2$.

ITERATE: FOR $k = 1, 2, \ldots,$ DO:

COMPUTE $p_k = x_k - x_{k-1}$ AND Ap_k.

SET $u_k = u_{k-1} + Ap_k$ AND $v_k = v_{k-1} + p_k$.

SET $\rho_k = \|s_{k-1} - u_k\|_2$ AND DEFINE τ_k BY $1/\tau_k^2 = 1/\tau_{k-1}^2 + 1/\rho_k^2$.

SET $s_k = s_{k-1} - (\tau_k^2/\rho_k^2)u_k$ AND $y_k = y_{k-1} + (\tau_k^2/\rho_k^2)v_k$.

UPDATE $u_k \leftarrow (1 - (\tau_k^2/\rho_k^2))u_k$ AND $v_k \leftarrow (1 - (\tau_k^2/\rho_k^2))v_k$.

It would be trivial to incorporate a diagonal scaling matrix

$$\Omega_{k+1} = \operatorname{diag}(\omega_0, \ldots, \omega_k) \in \mathbb{R}^{(k+1)\times(k+1)}$$

into the above developments, as is typically done in deriving QMR-type methods. In place of (3.2.1) and (3.2.2), we would then have

$$\tau_k \equiv \min_{\bar{z}\in\mathbb{R}^k} \|\Omega_{k+1}G_{k+1}(e_1 - \bar{H}_k\bar{z})\|_2 = \sqrt{\frac{1}{\sum_{j=0}^{k}\frac{1}{\omega_j^2\rho_j^2}}},$$

and, for the minimizing $\bar{z}$,

$$s_k \equiv V_{k+1}\Omega_{k+1}^{-1}\left[\Omega_{k+1}G_{k+1}(e_1 - \bar{H}_k\bar{z})\right] = \frac{1}{\sum_{j=0}^{k}\frac{1}{\omega_j^2\rho_j^2}}\sum_{i=0}^{k}\frac{1}{\omega_i^2\rho_i^2}r_i.$$

Algorithms 3.2.1 and 3.2.2 would be modified by replacing each occurrence of ρ_k^2 by $\omega_k^2\rho_k^2$. Similar remarks hold for the developments in §3.1.

As in Theorem 3.1, each s_k generated by Algorithms 3.2.1 and 3.2.2 is given by (3.2.2) as a convex combination of $r_0, \ldots, r_k$. Also, (3.2.1) and (3.2.2) imply immediately an analogue of (3.1.4) and (3.1.8), i.e.,

$$\|s_k\|_2 \le \sqrt{k+1}\,\tau_k = \sqrt{\frac{1}{\frac{1}{k+1}\sum_{j=0}^{k}\frac{1}{\rho_j^2}}}.$$

The remarks in the two paragraphs following Theorem 3.1, with appropriate changes, are valid here.

We mention the possibility of applying Algorithms 3.2.1 and 3.2.2 repeatedly to produce increasingly "smoothed" residuals and iterates. However, in an experiment that we performed, applying these algorithms twice to a BCG sequence (i.e., applying them once to a QMR sequence) showed no practical advantages.

In the case of the CGS method, Algorithms 3.2.1 and 3.2.2 can be applied in a straightforward way; however, we note an interesting alternative in this specific case. We write the CGS method as follows:

ALGORITHM 3.2.3. Conjugate gradient squared method [15].

INITIALIZE:
CHOOSE x_0 AND SET $p_0 = u_0 = r_0 = b - Ax_0$ AND $v_0 = Ap_0$.
CHOOSE $\tilde{r}_0$ SUCH THAT $\rho_0 = \tilde{r}_0^T r_0 \neq 0$.

ITERATE: FOR $k = 1, 2, \ldots,$ DO:
COMPUTE $\sigma_{k-1} = \tilde{r}_0^T v_{k-1}$, $\alpha_{k-1} = \rho_{k-1}/\sigma_{k-1}$, AND
$q_k = u_{k-1} - \alpha_{k-1}v_{k-1}$.
SET $x_k = x_{k-1} + \alpha_{k-1}(u_{k-1} + q_k)$ AND $r_k = r_{k-1} - \alpha_{k-1}A(u_{k-1} + q_k)$.
SET $\rho_k = \tilde{r}_0^T r_k$, $\beta_k = \rho_k/\rho_{k-1}$, $u_k = r_k + \beta_k q_k$,
$p_k = u_k + \beta_k(q_k + \beta_k p_{k-1})$, AND $v_k = Ap_k$.

We define auxiliary iterates and residuals as follows: For $k = 0, 1, \ldots,$ set

$$\begin{aligned} \hat{x}_{2k} &= x_k, & \hat{x}_{2k+1} &= x_k + \alpha_k u_k, \\ \hat{r}_{2k} &= r_k, & \hat{r}_{2k+1} &= r_k - \alpha_k A u_k. \end{aligned} \tag{3.2.3}$$

It is not hard to verify that, if we apply Algorithms 3.2.1 and 3.2.2 (modified, if necessary, to incorporate diagonal scaling) to the iterates $\{\hat{x}_k\}$ and residuals $\{\hat{r}_k\}$, then the resulting method is equivalent to the "transpose-free" QMR-like method derived from CGS in [6]. This is *not* the same as the method obtained by the straightforward application of Algorithms 3.2.1 and 3.2.2 to CGS; in §4.3, we illustrate the practical performance of these two ways of applying QMRS to CGS. This equivalence may help to explain experiments in [3] and also Experiments 5 and 6 in §4.3 below, in which residual norms from the method of [6] are roughly comparable to the best CGS residual norms obtained so far. (However, an example is given in [8] in which CGS diverges while the method of [6] converges.)

One can similarly obtain the QMRCGSTAB method of [3] by applying Algorithms 3.2.1 and 3.2.2 to Bi-CGSTAB [17]. Indeed, in addition to iterates $\{x_k\}$ and residuals $\{r_k\}$, Bi-CGSTAB produces $\{s_k, p_k, \alpha_k, \omega_k\}$ such that

$$x_k = x_{k-1} + \alpha_k p_k + \omega_k s_k,$$
$$s_k = r_{k-1} - \alpha_k A p_k, \qquad r_k = s_k - \omega_k A s_k.$$

In this case, we define auxiliary iterates and residuals by

$$\hat{x}_{2k} = x_k, \qquad \hat{x}_{2k+1} = x_k + \alpha_{k+1} p_{k+1},$$
$$\hat{r}_{2k} = r_k, \qquad \hat{r}_{2k+1} = s_{k+1},$$

for $k = 0, 1, \ldots$. Then the method obtained by applying Algorithms 3.2.1 and 3.2.2 (modified to incorporate diagonal scaling, if necessary) to $\{\hat{x}_k\}$ and $\{\hat{r}_k\}$ is equivalent to QMRCGSTAB.

4. Numerical experiments. We report on numerical experiments that illustrate the performance of the algorithms discussed previously. The test problem used in all but one of the experiments is a discretization of

$$\begin{aligned} \Delta u + cu + d\frac{\partial u}{\partial x} &= f \quad \text{in } D, \\ u &= 0 \quad \text{on } \partial D, \end{aligned} \tag{4.1}$$

where $D = [0, 1] \times [0, 1]$ and c and d are constants. In the experiments outlined here, we took $f \equiv 1$ and used a 100×100 mesh of equally spaced discretization points in D, so that the resulting linear systems were of dimension 10,000. Discretization was by the usual second order centered differences. Preconditioning, when used, was with a fast Poisson solver from FISHPACK [16]. In all experiments, computing was done in double precision on Sun Microsystems workstations.

4.1. Comparing Algorithms 2.1 and 2.2. We compared the numerical performance of the mathematically equivalent MRS Algorithms 2.1 and 2.2 in the following two experiments.

Experiment 1. This was a controlled experiment in which we artificially simulated the numerical breakdown of a convergent algorithm through exponential error growth. In each simulation, we first generated random[3] A and b and computed $x_* = A^{-1}b$ using a direct method; then, for $k = 0, \ldots, k_{\max}$, we generated $x_k = x_* + 2^{-k}u_k$ for random u_k, computed $r_k = b - Ax_k$, and perturbed $x_k \leftarrow x_k + \epsilon(k/k_{\max})^\nu v_k$ and $r_k \leftarrow r_k + \epsilon(k/k_{\max})^\nu w_k$ for random

[3]Here, "random" means having components that are sampled independently from a normal distribution with zero mean and unit variance. Random normal components were generated using the URAND subroutine from [5] followed by a Box–Muller transformation; see [5, p. 247]. In the particular experiment reported in Fig. 1, the seed 2468 was used in URAND.

v_k and w_k and fixed $\epsilon > 0$ and positive integer ν. We plotted $\log_{10} \|s_k\|_2$ and $\log_{10} \|b - Ay_k\|_2$ versus k for both Algorithms 2.1 and 2.2 in a number of trials. Typical results (with $n = 10$, $\epsilon = 10^2$, $\nu = 10$, and $k_{\max} = 50$) are shown in Fig. 1. Note that in Fig. 1, the solid curve is actually a superposition of the curves for $\log \|s_k\|_2$ and $\log \|b - Ay_k\|_2$ generated by Algorithm 2.2; they are visually indistinguishable. In contrast, the corresponding curves for Algorithm 2.1 diverge strongly.

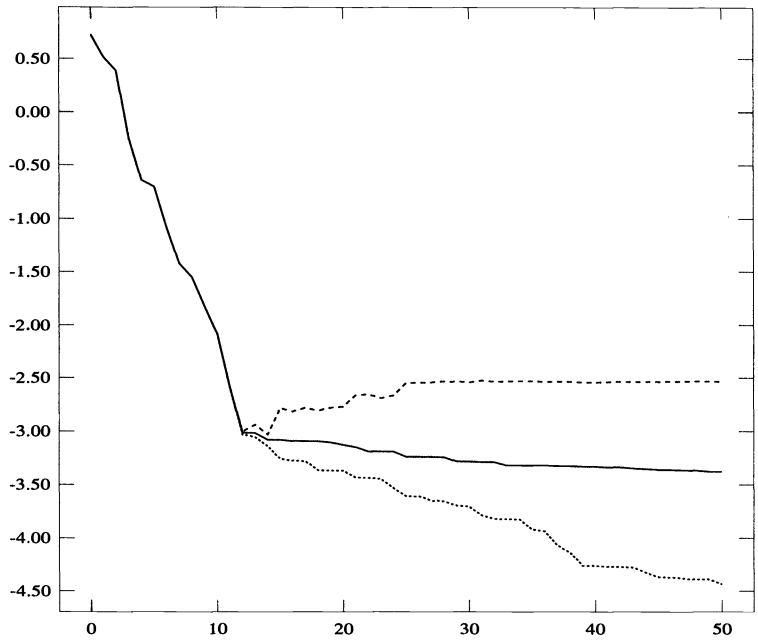


FIG. 1. *Residual norms versus the number of iterations for Experiment* 1. *Solid curve*: $\log_{10} \|s_k\|_2$ *and* $\log_{10} \|b - Ay_k\|_2$ *for Algorithm* 2.2; *dotted curve*: $\log_{10} \|s_k\|_2$ *for Algorithm* 2.1; *dashed curve*: $\log_{10} \|b - Ay_k\|_2$ *for Algorithm* 2.1.

Experiment 2. In this, we applied BCG without preconditioning to problem (4.1) with $c = d = 50$. For insight here and in Experiment 3 below, we first plotted $\log_{10} \|r_k^{\mathrm{BCG}}\|_2$ and $\log_{10} \|b - Ax_k^{\mathrm{BCG}}\|_2$ versus k in Fig. 2. Note the divergence of $\|r_k^{\mathrm{BCG}}\|_2$ and $\|b - Ax_k^{\mathrm{BCG}}\|_2$ beyond about $k = 280$; this is apparently due to rounding errors introduced in earlier iterations when $\|r_k^{\mathrm{BCG}}\|_2$ was very large. We next plotted $\log_{10} \|s_k\|_2$ and $\log_{10} \|b - Ay_k\|_2$ versus k for both Algorithms 2.1 and 2.2. The results are shown in Fig. 3, in which the graphs of $\log_{10} \|b - Ay_k\|_2$ for Algorithm 2.1 (solid curve) and for Algorithm 2.2 (long-dashed curve) are nearly superimposed. Note that for Algorithm 2.2, the values of $\log_{10} \|s_k\|_2$ stay fairly close to the values of $\log_{10} \|b - Ay_k\|_2$ in the later iterations, while for Algorithm 2.1, the values of $\log_{10} \|s_k\|_2$ are wrongly forced downward by the (inaccurate) decreasing values of $\|r_k^{\mathrm{BCG}}\|_2$. In view of the superior performance of Algorithm 2.2, it was used to implement MRS in the remaining experiments discussed below.

4.2. Smoothing methods for BCG. We compared the performance of BCG, QMR, and MRS applied to BCG on problem (4.1) in the two experiments below. In these, MRS was implemented as in Algorithm 2.2, and the QMR iterates and their residuals were generated from the BCG iterates and residuals using either (3.1.10) or a mathematically equivalent implementation of Algorithm 3.2.2. Caution: Comments below regarding the numerical

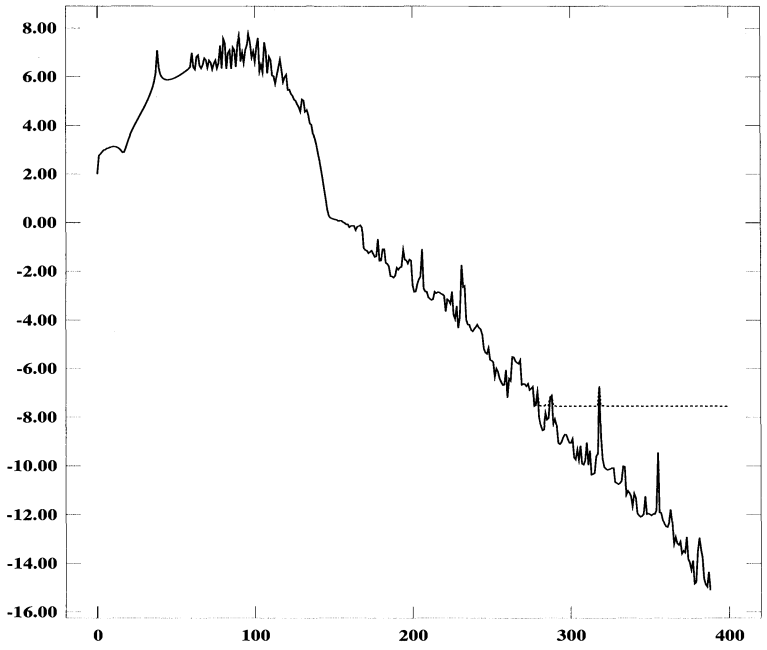


FIG. 2. *Recursive and true* BCG *residual norms versus the number of iterations for Experiments* 2 *and* 3. *Solid curve*: $\log_{10} \|r_k^{\mathrm{BCG}}\|_2$; *dotted curve*: $\log_{10} \|b - Ax_k^{\mathrm{BCG}}\|_2$.

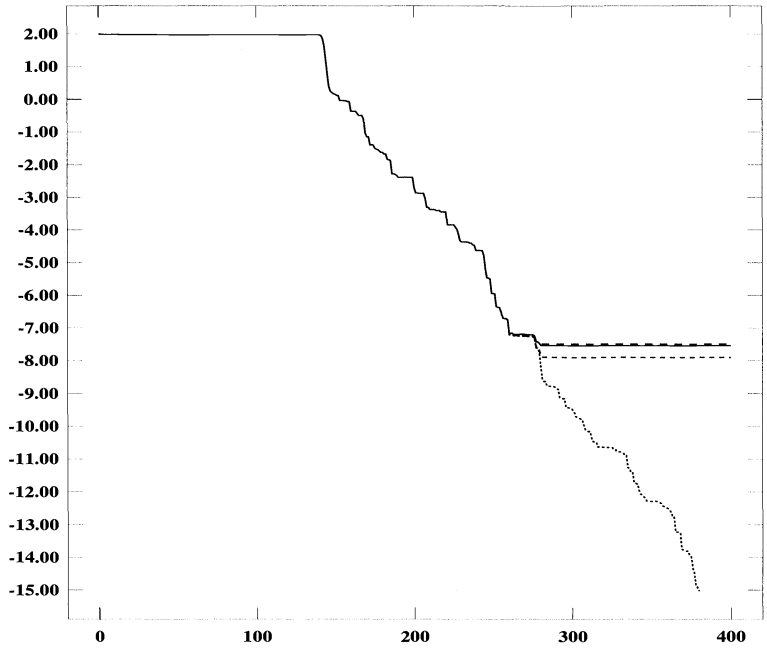


FIG. 3. *Residual norms versus the number of iterations for Experiment* 2. *Solid curve and nearly superimposed long-dashed curve*: $\log_{10} \|b - Ay_k\|_2$ *for Algorithms* 2.1 *and* 2.2; *dotted curve*: $\log_{10} \|s_k\|_2$ *for Algorithm* 2.1; *short-dashed curve*: $\log_{10} \|s_k\|_2$ *for Algorithm* 2.2.

accuracy of QMR iterates are *not* intended to apply to the general QMR method of [7], which uses the look-ahead Lanczos process; they only pertain to the soundness of (3.1.10) and Algorithm 3.2.2 applied to the BCG iterates.

Experiment 3. As in Experiment 2, we considered problem (4.1) with $c = d = 50$, without preconditioning. We first compared the accuracy of (3.1.10) and Algorithm 3.2.2 for obtaining QMR iterates from BCG iterates. The results, which are analogous to those in Fig. 3, are shown in Fig. 4, in which the graphs of $\log_{10} \|b - Ax_k^{\mathrm{QMR}}\|_2$ for (3.1.10) (solid curve) and for Algorithm 3.2.2 (long-dashed curve) are nearly indistinguishable. The remarks made in Experiment 2 about Fig. 3 are valid here, with the appropriate changes. In view of the superior performance of Algorithm 3.2.2, it was used instead of (3.1.10) or Algorithm 3.2.1 in the remaining experiments reported below, except where noted.

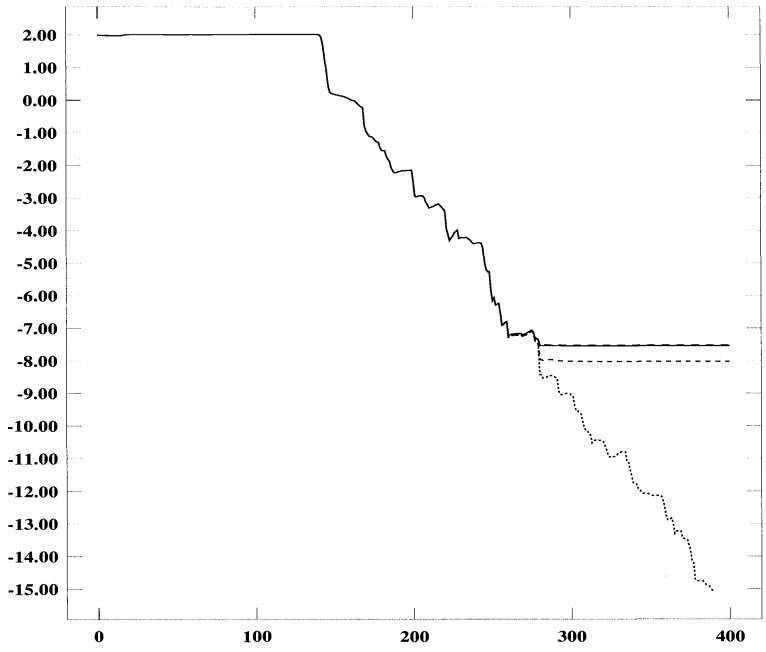


FIG. 4. *Residual norms versus the number of iterations for Experiment* 3. *Solid curve and nearly indistinguishable long-dashed curve*: $\log_{10} \|b - Ax_k^{\mathrm{QMR}}\|_2$ *for* (3.1.10) *and Algorithm* 3.2.2; *dotted curve*: $\log_{10} \|r_k^{\mathrm{QMR}}\|_2$ *for* (3.1.10); *short-dashed curve*: $\log_{10} \|r_k^{\mathrm{QMR}}\|_2$ *for Algorithm* 3.2.2.

We next compared the performance of BCG, QMR, and MRS applied to BCG on this problem. The results are shown in Fig. 5. Note that the solid curve in Fig. 5 is the *recursive* residual value $\log_{10} \|r_k^{\mathrm{BCG}}\|_2$, rather than the *true* value $\log_{10} \|b - Ax_k^{\mathrm{BCG}}\|_2$; cf. Fig. 2. Both QMR and MRS were very effective in smoothing the BCG iterates. Although fine details are hard to make out in parts of Fig. 5, the performance of QMR and MRS was, in fact, very similar over all iterations; of course, the MRS residual norms were monotone decreasing while those of QMR, although trending downward fairly smoothly, were not. Note that both QMR and MRS very effectively stabilized the iterates and residuals at about the point of greatest *true* residual reduction of BCG.

Experiment 4. We applied the three methods with preconditioning to problem (4.1) with $c = d = 100$. The preconditioning resulted in relatively well-behaved BCG residual norms, and QMR and MRS applied to BCG performed very similarly. There was no evidence of numerical error. The results are shown in Fig. 6.

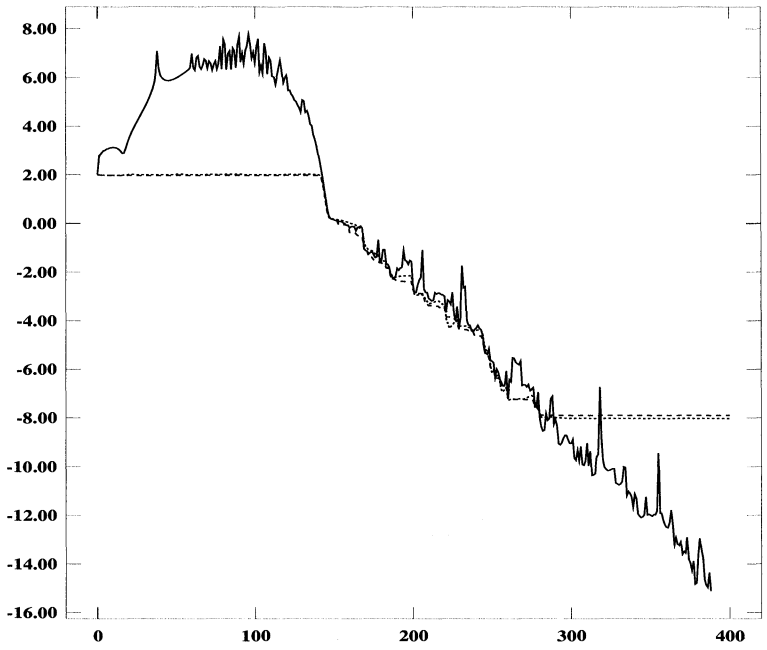


FIG. 5. *Residual norms versus the number of iterations for Experiment* 3. *Solid curve*: $\log_{10} \|r_k^{\mathrm{BCG}}\|_2$; *dotted curve*: $\log_{10} \|r_k^{\mathrm{QMR}}\|_2$; *dashed curve*: $\log_{10} \|s_k\|_2$ *for* MRS *Algorithm* 2.2 *applied to the* BCG *iterates*.

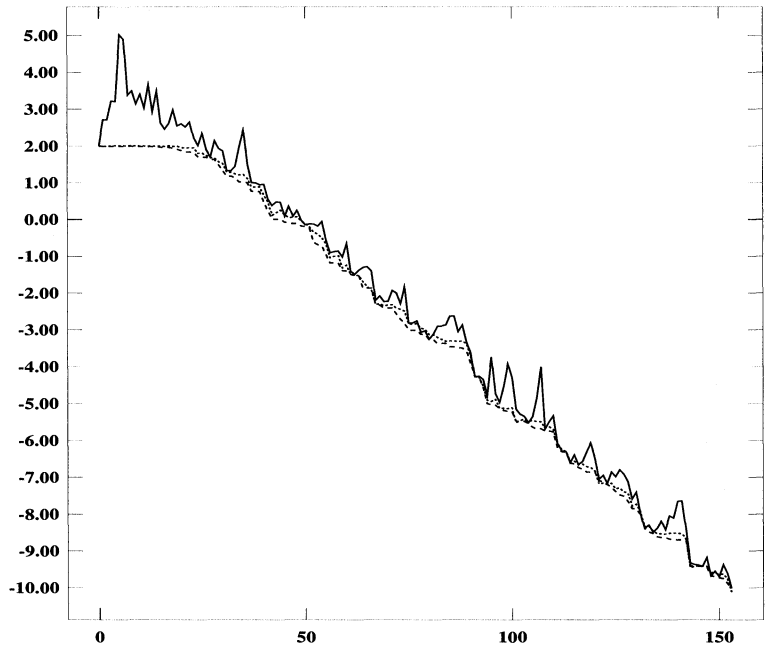


FIG. 6. *Residual norms versus the number of iterations for Experiment* 4. *Solid curve*: $\log_{10} \|r_k^{\mathrm{BCG}}\|_2$; *dotted curve*: $\log_{10} \|r_k^{\mathrm{QMR}}\|_2$; *dashed curve*: $\log_{10} \|s_k\|_2$ *for* MRS *Algorithm* 2.2 *applied to the* BCG *iterates*.

4.3. Smoothing methods for CGS. In the two experiments below, we compared the performance on problem (4.1) of CGS, the "transpose-free" QMR-like method of [6] (referred to as TFQMR below), CGS with QMRS, and CGS with MRS. We implemented TFQMR by applying QMRS to $\{\hat{x}_k\}$ and $\{\hat{r}_k\}$ defined in (3.2.3); see §3.2. The implementation of CGS with QMRS was straightforward.

Experiment 5. We applied the methods without preconditioning to problem (4.1) with $c = d = 5$. The results are given in Fig. 7. The three smoothing methods very effectively smoothed the very ill-behaved CGS residuals and performed roughly the same, although MRS tended to give slightly smaller residual norms than the other two methods. Although not shown in Fig. 7, significant divergence of $\|r_k^{\mathrm{CGS}}\|_2$ and $\|b - Ax_k^{\mathrm{CGS}}\|_2$ began at about iteration 262 and, as in Experiment 3 and Fig. 5, all three smoothing methods effectively stabilized the iterates and residuals at about the point of greatest *true* CGS residual reduction.

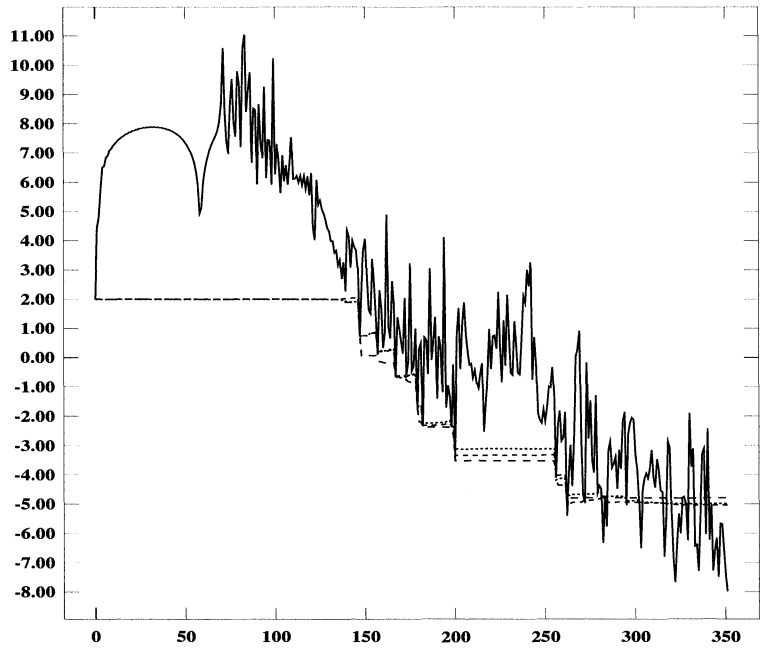


FIG. 7. *Residual norms versus the number of iterations for Experiment* 5. *Solid curve*: $\log_{10}\|r_k^{\mathrm{CGS}}\|_2$; *dotted curve*: $\log_{10}$ *of the* TFQMR *residual norms*; *short-dashed curve*: $\log_{10}\|s_k\|_2$ *for* QMRS *Algorithm* 3.2.2 *applied to the* CGS *iterates*; *long-dashed curve*: $\log_{10}\|s_k\|_2$ *for* MRS *Algorithm* 2.2 *applied to the* CGS *iterates*.

Experiment 6. We applied the methods with preconditioning to problem (4.1) with $c = d = 50$. Because r_k^{CGS} and $b - Ax_k^{\mathrm{CGS}}$ remained very close throughout this experiment, we used Algorithm 3.2.1 rather than Algorithm 3.2.2. The results are given in Fig. 8. It is evident that all three smoothing methods worked very effectively, especially in stabilizing the iterates and residuals once the limits of residual reduction were reached. It is also notable how similarly QMRS and MRS behaved, although MRS ultimately gave the smallest residual norms of all methods.

5. Summary and conclusions. We have focused on two residual smoothing techniques of the general form (1.1), minimum residual smoothing (MRS) and quasi-minimal residual smoothing (QMRS). The former generates from given iterates a sequence of auxiliary iterates with monotone decreasing residual norms that are no greater than those of the original sequence. The latter generates auxiliary iterates with residual norms that typically decrease

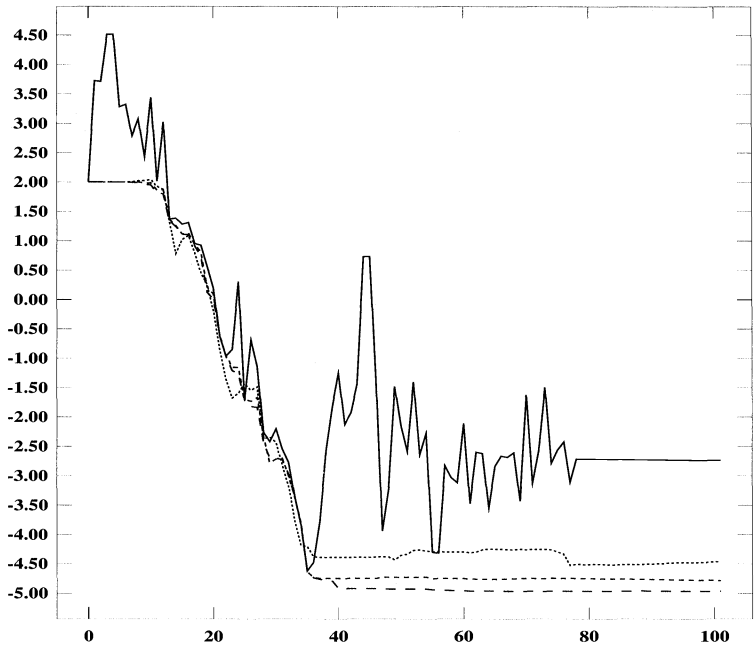


FIG. 8. *Residual norms versus the number of iterations for Experiment* 6. *Solid curve*: $\log_{10} \|r_k^{\mathrm{CGS}}\|_2$; *dotted curve*: $\log_{10}$ *of the* TFQMR *residual norms*; *short-dashed curve*: $\log_{10} \|s_k\|_2$ *for* QMRS *Algorithm* 3.2.1 *applied to the* CGS *iterates*; *long-dashed curve*: $\log_{10} \|s_k\|_2$ *for* MRS *Algorithm* 2.2 *applied to the* CGS *iterates*.

fairly smoothly, if not necessarily monotonically. It also provides insight into the workings of QMR-type methods and, in particular, indicates how they are related to and smooth the residual norms of underlying methods such as BCG and CGS.

These residual smoothing techniques provide important practical tools. Their performance is illustrated in a number of numerical experiments in §4; these indicate that both MRS and QMRS are reliable and very effective in smoothing the residuals of the underlying method when applied as in Algorithms 2.2 and 3.2.2, respectively. The performance of MRS and QMRS was roughly similar in our experiments. However, the MRS residual norms were monotone decreasing while those of QMRS, although typically trending fairly smoothly downward, were not; also, the MRS residual norms were often, although not always, slightly smaller than the QMRS residual norms and, in some cases, tended to remain a little more stable in the final iterations. On the basis of work to date, we have some preference for MRS over QMRS for general use.

Having smoothly decreasing or monotone decreasing residual norms may be of real importance or just a nicety, depending on the application. In fact, this can always be achieved trivially just by saving the best iterate obtained so far; in some of our experiments, MRS and QMRS did not produce significantly smaller residual norms than this simple technique. Thus the good behavior of the MRS and QMRS residual norms alone may not always be enough to justify their use. However, a strongly compelling reason for using MRS or QMRS with methods such as BCG and CGS is their effectiveness in stabilizing the iterates and residuals once the limits of residual reduction have been reached. This is perhaps most clearly seen in Figs. 5 and 7, in which the recursive residuals generated by the underlying method continue to decrease long after they have lost accuracy, while the MRS and QMRS residuals become stable and remain fairly accurate. In addition to helping to avoid misleading results, this stabilizing effect could be useful in obtaining further accuracy from the method. Indeed, one might be

able to detect the onset of stability and restart the method with a fresh, more accurate residual, thereby making further accurate residual reduction possible. We have successfully carried out this strategy in experiments.

We conclude by noting recent related work. In [9] it is shown that iterates produced by certain pairs of "orthogonal error" methods can be related through (1.1), extending Theorem 4.2 of [18, p. 78]. In [1], it is assumed that two sequences $\{x'_k\}$ and $\{x''_k\}$ are given, and an auxiliary sequence is generated by

$$y_k = (1 - \eta_k)x'_k + \eta_k x''_k,$$

where η_k is chosen to minimize the residual at y_k. If a single sequence $\{x_k\}$ is given, then the choice $x'_k = y_{k-1}, x''_k = x_k$ gives MRS as a special case. Many other possibilities are explored in [1], and the possibility is raised of combining given $\{x_k^{(1)}\}, \ldots, \{x_k^{(m)}\}$ to produce $\{y_k\}$ by

$$y_k = \eta_k^{(1)} x_k^{(1)} + \cdots + \eta_k^{(m)} x_k^{(m)}, \qquad \sum_{i=1}^{m} \eta_k^{(i)} = 1. \tag{5.1}$$

In [1], it is suggested that the $\eta_k^{(i)}$'s be chosen to minimize the residual at y_k, but more general choices may be useful. For example, the QMR squared method of [8] can be obtained from the CGS iterates and certain auxiliary quantities through a relation of the form (5.1) in which $x_k^{(1)} = y_{k-1}$; see (4.11) of [8, p. 9].

REFERENCES

[1] C. Brezinski and M. Redivo Zaglia, *A hybrid procedure for solving linear systems*, Tech. Rep. ANO-283, Laboratoire d'Analyse Numerérique et d'Optimization, Univ. des Sciences et Technologies de Lille, Sept. 1992.

[2] T. F. Chan, L. de Pillis, and H. van der Vorst, *A transpose-free squared Lanczos algorithm and application to solving non-symmetric linear systems*, Tech. Rep. CAM 91-17, Dept. of Mathematics, Univ. of California at Los Angeles, Sept. 1991.

[3] T. F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, and C. H. Tong, QMRCGSTAB: *a quasi-minimal residual variant of the* Bi-CGSTAB *algorithm for nonsymmetric systems*, Tech. Rep. CAM 92-26, Dept. of Mathematics, Univ. of California at Los Angeles, June 1992.

[4] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis Dundee 1975, Scotland, Lecture Notes in Math. 506, Springer-Verlag, Berlin, Heidelberg, New York, 1976, pp. 73–89.

[5] G. E. Forsythe, M. A. Malcolm, and C. B. Moler, *Computer Methods for Mathematical Computations*, Series in Automatic Computation, Prentice–Hall, Englewood Cliffs, NJ, 1977.

[6] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, Tech. Rep. 91.18, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, CA, Sept. 1991; SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[7] 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.

[8] R. W. Freund and T. Szeto, *A quasi-minimal residual squared algorithm for non-Hermitian linear systems*, Tech. Rep. 91.26, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, CA, Dec. 1991.

[9] M. H. Gutknecht, *Changing the norm in conjugate gradient type algorithms*, Tech. Rep. 91-15, Interdisziplinäres Projektzentrum für Supercomputing, Eidgenössische Technische Hochschule, Zürich, Aug. 1991.

[10] ———, *Variants of* BICGSTAB *for matrices with complex spectrum*, Tech. Rep. 91-14, Interdisziplinäres Projektzentrum für Supercomputing, Eidgenössische Technische Hochschule, Zürich, Aug. 1991.

[11] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Stand., 45 (1950), pp. 255–282.

[12] ———, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Stand., 49 (1952), pp. 33–53.

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

[14] W. SCHÖNAUER, *Scientific Computing on Vector Computers*, North–Holland, Amsterdam, New York, Oxford, Tokyo, 1987.

[15] P. SONNEVELD, CGS, *a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[16] P. N. SWARZTRAUBER AND R. A. SWEET, *Efficient* FORTRAN *subprograms for the solution of elliptic partial differential equations*, ACM Trans. Math. Software, 5 (1979), pp. 352–364.

[17] 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.

[18] R. WEISS, *Convergence Behavior of Generalized Conjugate Gradient Methods*, Ph.D. thesis, Univ. of Karlsruhe, Germany, 1990.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 313–337, March 1994

005

AN IMPLEMENTATION OF THE QMR METHOD BASED ON COUPLED TWO-TERM RECURRENCES*

ROLAND W. FREUND† AND NOËL M. NACHTIGAL‡

Abstract. Recently, the authors proposed a new Krylov subspace iteration, the quasi-minimal residual (QMR) algorithm, for solving non-Hermitian linear systems. In the original implementation of the QMR method, the Lanczos process with look-ahead is used to generate basis vectors for the underlying Krylov subspaces. In the Lanczos algorithm, these basis vectors are computed by means of three-term recurrences. It has been observed that, in finite-precision arithmetic, vector iterations based on three-term recursions are usually less robust than mathematically equivalent coupled two-term vector recurrences.

This paper presents a look-ahead algorithm that constructs the Lanczos basis vectors by means of coupled two-term recursions. Some implementation details are given, and the look-ahead strategy is described. A new implementation of the QMR method, based on this coupled two-term algorithm, is proposed. A simplified version of the QMR algorithm without look-ahead is also presented, and the special case of QMR for complex symmetric linear systems is considered. Results of numerical experiments comparing the original and the new implementations of the QMR method are reported.

Key words. Krylov subspace iteration, quasi-minimal residual method, non-Hermitian matrices, coupled two-term recurrences, look-ahead techniques, complex symmetric matrices

AMS subject classifications. 65F10, 65N22

1. Introduction.

Recently, we proposed a new Krylov subspace method, the quasi-minimal residual (QMR) algorithm [9], for the iterative solution of general nonsingular non-Hermitian systems of linear equations

$$Ax = b. \tag{1.1}$$

The QMR method is closely related to the classical biconjugate gradient (BCG) algorithm due to Lanczos [17]. The BCG method aims at generating approximate solutions for (1.1) that satisfy a Galerkin condition. Unfortunately, for non-Hermitian matrices A, such iterates need not always exist, and this is the source of one of the two possible breakdowns—triggered by division by 0—that can occur during each iteration step of BCG. The second breakdown is equivalent to the possible breakdown—also triggered by division by 0—of the nonsymmetric Lanczos process [16]. In finite-precision arithmetic, it is unlikely that one encounters exact breakdowns in the BCG algorithm. However, near-breakdowns can occur, which can cause a buildup of round-off in successive iterations. Another problem with BCG is the lack of a residual minimization property for its iterates, which leads to a typically erratic convergence behavior, with wild oscillations in the residual norm.

The QMR method offers remedies for these problems. It generates iterates that are defined by a quasi minimization of the residual norm, rather than a Galerkin condition. This eliminates the oscillations and leads to a smooth and nearly monotone convergence behavior. In contrast to BCG, a QMR iterate always exists at each iteration step, and this excludes breakdowns caused by nonexistent iterates. Moreover, possible breakdowns in the underlying Lanczos

*Received by the editors June 4, 1992; accepted for publication (in revised form) January 27, 1993. This work was supported by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

†AT&T Bell Laboratories, Room 2C-420, 600 Mountain Ave., Murray Hill, New Jersey 07974-0636 (freund@research.att.com). This research was performed while this author was in residence at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, California 94035.

‡Research Institute for Advanced Computer Science, Mail Stop T041-5, NASA Ames Research Center, Moffett Field, California 94035-1000 (na.nachtigal@na-net.ornl.gov).

process are prevented by using look-ahead techniques. Therefore, except for the rare event of an incurable breakdown, breakdowns cannot occur in the QMR method.

In the original QMR algorithm [9], an implementation of the Lanczos method with look-ahead is used to generate basis vectors for the underlying Krylov subspaces. In the Lanczos process, these basis vectors are generated by means of three-term recurrences. It turns out that, in finite-precision arithmetic, these three-term vector iterations are usually less robust than mathematically equivalent coupled two-term vector recurrences. In this paper, we present a general look-ahead algorithm based on coupled two-term recursions for constructing basis vectors of Krylov subspaces. Based on this algorithm we then propose a new implementation of the QMR method.

We stress that the better numerical behavior of coupled two-term recurrences is not a characteristic of QMR only, and that this phenomenon was also observed for other Krylov subspace methods. For instance, Joubert [15] remarked that BCG, which is based on coupled two-term recurrences, is more robust than the mathematically equivalent Lanczos/Orthodir and Lanczos/Orthores algorithms, which are both based on three-term vector recursions. Indeed, Lanczos/Orthodir and Lanczos/Orthores often converge considerably slower than BCG, or they even diverge, while BCG converges; we refer the reader to [18] for numerical examples that illustrate this phenomenon. Finally, we remark that the standard implementation of the classical conjugate gradient (CG) method of Hestenes and Stiefel [14] is also based on coupled two-term recurrences. Reid [20] mentions that this version of CG is "preferable on computational grounds" to an equivalent three-term implementation of CG. Woźniakowski [24] presents numerical examples for which an implementation of CG based on two-term recurrences generates approximate solutions to a higher accuracy than does a three-term version of CG.

The remainder of the paper is organized as follows. In §2, we briefly review the Lanczos process and the original QMR algorithm. In §3, we present a sketch of the proposed look-ahead procedure for constructing Lanczos vectors by means of coupled two-term recurrences. In §4, we discuss the look-ahead strategy for this algorithm, and in §5, we give some implementation details. Next we combine the coupled two-term procedure with the QMR approach. In §6, we outline the resulting implementation for the general case of QMR with look-ahead. In §7, we present a simplified version of the QMR algorithm without look-ahead. In §8, we consider a variant of QMR for the special case of complex symmetric linear systems. In §9, we report results of numerical experiments comparing the original and the new implementations of the QMR method. Finally, in §10, we make some concluding remarks.

Throughout the paper, all vectors and matrices are allowed to have real or complex entries. As usual, $M^T = [m_{kj}]$ and $M^H = [\overline{m}_{kj}]$ denote the transpose and the conjugate transpose, respectively, of the matrix $M = [m_{jk}]$. We use $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ for the largest and smallest singular value of M, respectively. The vector norm $\|x\| := \sqrt{x^H x}$ is always the Euclidean norm and $\|M\| := \sigma_{\max}(M)$ is the corresponding matrix norm. We denote by

$$\mathcal{P}_n := \{\varphi(\lambda) \equiv \gamma_0 + \gamma_1\lambda + \cdots + \gamma_n\lambda^n \mid \gamma_0, \gamma_1, \ldots, \gamma_n \in \mathbb{C}\}$$

the set of all complex polynomials of degree at most n. The nth Krylov subspace of $\mathbb{C}^N$ generated by $c \in \mathbb{C}^N$ and the $N \times N$ matrix B is defined by

$$K_n(c, B) := \operatorname{span}\{c, Bc, \ldots, B^{n-1}c\},$$

and we will make use of the fact that

$$K_n(c, B) = \{\varphi(B)c \mid \varphi \in \mathcal{P}_{n-1}\}.$$

Furthermore, it is always assumed that A is an $N \times N$ matrix, singular or nonsingular.

Finally, we remark that, for complex matrices, there are two equivalent formulations of the Lanczos process, using either A^T or A^H. In this paper, we have chosen the formulation with A^T, for two reasons. First, it avoids complex conjugation of the scalars in some of the recurrence relations, and, second, the recursions reduce immediately for the special case of complex symmetric matrices.

2. The QMR algorithm. In this section, we briefly describe the QMR method and its original implementation [9]. We remark that, in [9], QMR was proposed for nonsingular linear systems (1.1). Freund and Hochbruck [8] showed that the QMR method can also be applied to singular square systems, and that it always generates well-defined iterates. However, as discussed in [8], these iterates converge to a meaningful solution of (1.1) only for consistent systems with coefficient matrices of index 1. An important special case for which these conditions are satisfied is consistent singular systems with a one-dimensional null space, i.e.,

$$b \in \{Ax \mid x \in \mathbb{C}^N\} \quad \text{and} \quad \dim\{x \in \mathbb{C}^N \mid Ax = 0\} = 1. \tag{2.1}$$

In this paper, we always consider the QMR method for the general case of $N \times N$ linear systems, with singular or nonsingular coefficient matrices A.

2.1. Krylov subspace methods. Let $x_0 \in \mathbb{C}^N$ be an arbitrary initial guess for the linear system (1.1), and denote by $r_0 := b - Ax_0$ the corresponding residual vector. An iterative scheme for solving (1.1) is called a *Krylov subspace method* if, for any choice of x_0, it produces approximate solutions of the form

$$x_n \in x_0 + K_n(r_0, A), \qquad n = 1, 2, \ldots . \tag{2.2}$$

Clearly, the design of a Krylov subspace algorithm consists of two main parts: the construction of suitable basis vectors for the Krylov subspaces $K_n(r_0, A)$ in (2.2) and the choice of the actual iterates x_n. The QMR method is an example of a Krylov subspace iteration, where the basis vectors are generated by means of the nonsymmetric Lanczos process, and the iterates are characterized by a QMR property. Next, we describe these two main ingredients of QMR.

2.2. The Lanczos process. The Lanczos method is started with two vectors

$$v_1 = r_0/\rho_1, \quad \text{where } \rho_1 = \|r_0\|, \tag{2.3}$$

and an arbitrary second starting vector

$$w_1 \in \mathbb{C}^N \quad \text{with } \|w_1\| = 1. \tag{2.4}$$

It then produces two sequences of vectors

$$\{v_j\}_{j=1}^n \quad \text{and} \quad \{w_j\}_{j=1}^n, \tag{2.5}$$

such that, for $n = 1, 2, \ldots,$

$$\begin{aligned} \operatorname{span}\{v_1, v_2, \ldots, v_n\} &= K_n(v_1, A), \\ \operatorname{span}\{w_1, w_2, \ldots, w_n\} &= K_n(w_1, A^T), \end{aligned} \tag{2.6}$$

and the two sets are biorthogonal or block biorthogonal, i.e.,

$$W_n^T V_n = D_n, \tag{2.7}$$

where D_n is a diagonal or block-diagonal matrix. Here, and in the sequel, we denote by

$$V_n := [\, v_1 \quad v_2 \quad \cdots \quad v_n \,] \quad \text{and} \quad W_n := [\, w_1 \quad w_2 \quad \cdots \quad w_n \,] \tag{2.8}$$

the matrices containing the Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ as columns. We remark that the conditions (2.6)–(2.7) determine the vectors (2.5) only up to scaling. Throughout this paper, we always scale the Lanczos vectors to have unit length

$$\|v_n\| = \|w_n\| = 1, \qquad n = 1, 2, \ldots . \tag{2.9}$$

The crucial point of the Lanczos process is that vectors satisfying (2.6)–(2.7) can be constructed by means of short vector recursions. In the classical Lanczos algorithm [16], the vectors are generated using simple three-term recurrences:

$$\begin{aligned} \tilde{v}_{n+1} &= Av_n - v_n\mu_n - v_{n-1}\nu_n, \\ \rho_{n+1} &= \|\tilde{v}_{n+1}\|, \qquad v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \end{aligned} \tag{2.10}$$

$$\begin{aligned} \tilde{w}_{n+1} &= A^T w_n - w_n\mu_n - w_{n-1}(\nu_n\rho_n/\xi_n), \\ \xi_{n+1} &= \|\tilde{w}_{n+1}\|, \qquad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \end{aligned} \tag{2.11}$$

where

$$\mu_n = w_n^T A v_n / w_n^T v_n, \qquad \nu_n = \xi_n w_n^T v_n / w_{n-1}^T v_{n-1}. \tag{2.12}$$

In this case, the matrix D_n in (2.7) is diagonal and nonsingular:

$$D_n = \operatorname{diag}(\delta_1, \delta_2, \ldots, \delta_n), \quad \text{where } \delta_j := w_j^T v_j \neq 0. \tag{2.13}$$

Furthermore, we note that, by using the notation introduced in (2.8), the recurrence relations (2.10)–(2.11) for the first $n+1$ Lanczos vectors $\{v_j\}_{j=1}^{n+1}$ and $\{w_j\}_{j=1}^{n+1}$ can be written compactly in matrix form:

$$AV_n = V_{n+1}H_n, \tag{2.14}$$

$$A^T W_n = W_{n+1}\Gamma_{n+1}^{-1} H_n \Gamma_n. \tag{2.15}$$

Here, H_n is an $(n+1) \times n$ tridiagonal matrix, and

$$\Gamma_n := \operatorname{diag}(\gamma_1, \gamma_2, \ldots, \gamma_n), \quad \text{where } \gamma_j := \begin{cases} 1 & \text{if} \quad j = 1, \\ \gamma_{j-1}\rho_j/\xi_j & \text{if} \quad j > 1, \end{cases} \tag{2.16}$$

is a diagonal scaling matrix with positive diagonal entries. Finally, for later use, we note that all subdiagonal elements of H_n are nonzero, and therefore

$$\operatorname{rank} H_n = n. \tag{2.17}$$

Unfortunately, in the classical Lanczos algorithm, breakdowns cannot be excluded. Indeed, by (2.12), division by 0 will occur during the construction of v_{n+1} and w_{n+1} if $w_n^T v_n = 0$, but $w_n \neq 0$ and $v_n \neq 0$. Parlett, Taylor, and Liu [19] were the first to devise a practical modification of the Lanczos procedure that uses look-ahead to skip over possible *exact breakdowns* ($w_n^T v_n = 0$) or *near-breakdowns* ($w_n^T v_n$ is nonzero, but small in some sense). The QMR algorithm is based on a different implementation of the look-ahead Lanczos method, recently

developed by Freund, Gutknecht, and Nachtigal [7]. Next, we briefly sketch this look-ahead Lanczos procedure.

As in the classical Lanczos algorithm, two sequences of Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ are generated, starting with (2.3) and (2.4). Again, we will use the matrix notation V_n and W_n defined in (2.8). As before, these vectors satisfy (2.6)–(2.7), but now D_n is generally only a block-diagonal matrix, with $l := l(n)$ square blocks of dimension h_j, $j = 1, 2, \ldots, l$, on the diagonal. More precisely, we have

$$D_n = \operatorname{diag}(D^{(1)}, D^{(2)}, \ldots, D^{(l)}), \quad \text{where } D^{(j)} := (W^{(j)})^T V^{(j)}. \tag{2.18}$$

Here, the matrices $V^{(j)}$ and $W^{(j)}$, $j = 1, 2, \ldots, l$, are defined by partitioning V_n and W_n into blocks, according to the look-ahead steps taken:

$$V_n = \begin{bmatrix} V^{(1)} & V^{(2)} & \cdots & V^{(l)} \end{bmatrix} \quad \text{and} \quad W_n = \begin{bmatrix} W^{(1)} & W^{(2)} & \cdots & W^{(l)} \end{bmatrix}. \tag{2.19}$$

We remark that the matrices $V^{(j)}$ and $W^{(j)}$ are of size $N \times h_j$, and they contain as their columns the Lanczos vectors constructed in the jth look-ahead step. The integer h_j is called the length of the jth look-ahead step, and l in (2.18)–(2.19) is the number of look-ahead steps that were performed during the first n steps of the Lanczos process. For later use, we introduce some further notation. For $j = 1, 2, \ldots$, we denote by n_j the index of the first vectors of the blocks $V^{(j)}$ and $W^{(j)}$ in (2.19); hence, we have

$$V^{(j)} = \begin{bmatrix} v_{n_j} & v_{n_j+1} & \cdots \end{bmatrix} \quad \text{and} \quad W^{(j)} = \begin{bmatrix} w_{n_j} & w_{n_j+1} & \cdots \end{bmatrix}. \tag{2.20}$$

Note that the indices n_j satisfy

$$1 =: n_1 < n_2 < \cdots < n_l \leq n < n_{l+1}. \tag{2.21}$$

The vectors v_{n_j} and w_{n_j} are called *regular*, while the remaining vectors in the blocks $V^{(j)}$ and $W^{(j)}$ are called *inner*. We remark that, in view of (2.7) and (2.18), the regular vectors are biorthogonal to all previous Lanczos vectors, i.e.,

$$w_i^T v_{n_j} = w_{n_j}^T v_i = 0 \quad \text{for all } i = 1, 2, \ldots, n_j - 1, \tag{2.22}$$

while the inner vectors in the jth blocks $V^{(j)}$ and $W^{(j)}$ are biorthogonal to all Lanczos vectors from the previous blocks, but not necessarily to the Lanczos vectors in the jth blocks. Finally, we note that, in (2.18), the blocks $D^{(j)}$, $j = 1, 2, \ldots, l-1$, are all nonsingular, while the last block $D^{(l)}$ is nonsingular if $n_{l+1} = n + 1$, i.e., if v_{n+1} and w_{n+1} are constructed as regular vectors.

In the look-ahead algorithm, the Lanczos vectors (2.5) are again generated using only short vector recurrences, which now involve vectors from the last two blocks $V^{(l)}$, $V^{(l-1)}$ and $W^{(l)}$, $W^{(l-1)}$, instead of just v_n, v_{n-1} and w_n, w_{n-1}, as in the classical algorithm. For example, v_{n+1} is computed by means of the relations

$$\begin{aligned} \tilde{v}_{n+1} &= A v_n - V^{(l)} \mu_n - V^{(l-1)} \nu_n, \\ \rho_{n+1} &= \|\tilde{v}_{n+1}\|, \qquad v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \end{aligned} \tag{2.23}$$

where $\mu_n \in \mathbb{C}^{h_l}$ and $\nu_n \in \mathbb{C}^{h_{l-1}}$ are suitably chosen coefficient vectors. The second Lanczos vector w_{n+1} is obtained similarly. As before, the recurrence relations for the Lanczos vectors can be summarized in the form (2.14)–(2.15). Here, H_n is now a block-tridiagonal matrix with l square blocks on the diagonal, where the jth block has dimension $h_j \times h_j$, $j = 1, 2, \ldots, l$.

In addition, H_n is also an upper Hessenberg matrix. Furthermore, (2.16) and (2.17) remain valid, and ρ_j and ξ_j in (2.16) are scaling factors used to ensure the normalization (2.9) of the Lanczos vectors; see (2.23).

We would like to stress that the look-ahead strategy (see [7] and also §4 below) is such that the algorithm performs mostly standard Lanczos steps, i.e., look-ahead steps of length $h_j = 1$ with blocks $V^{(j)}$ and $W^{(j)}$ that consist of only single Lanczos vectors. True look-ahead steps, i.e., steps of size $h_j > 1$, are only used to avoid exact and near-breakdowns. Typically, except for contrived examples, only few true look-ahead steps occur, and their size is usually small, mostly $h_j = 2$. We note that, if only steps of length $h_j = 1$ are performed, then the look-ahead Lanczos algorithm reduces to the classical algorithm. Finally, we remark that so-called incurable breakdowns [23], [13] can occur in the Lanczos process. Such breakdowns cannot be remedied by look-ahead, and indeed, in such a case, the look-ahead Lanczos algorithm would build a block of size $h_l = \infty$. Fortunately, incurable breakdowns are extremely rare, and they do not present a problem in practice.

The look-ahead Lanczos algorithm is intimately connected with formally orthogonal polynomials; see, e.g., [13], [7]. In particular, we will use the fact that each pair of Lanczos vectors v_j and w_j can be expressed in the form

$$v_j = \phi_{j-1}(A)v_1 \quad \text{and} \quad w_j = \gamma_j \phi_{j-1}(A^T)w_1, \tag{2.24}$$

where $\phi_{j-1} \in \mathcal{P}_{j-1}$ is of exact degree $j-1$ and $\gamma_j > 0$ is defined in (2.16).

For further details and properties of the look-ahead Lanczos algorithm, we refer the reader to [7].

2.3. The QMR property. In the QMR method, the vectors $\{v_j\}_{j=1}^n$ generated by the look-ahead Lanczos algorithm are used as a basis for the Krylov subspace $K_n(r_0, A)$ in (2.2). The nth QMR iterate x_n is then defined by

$$x_n = x_0 + V_n z_n, \tag{2.25}$$

where $z_n \in \mathbb{C}^n$ is the unique solution of the least-squares problem

$$\| f_{n+1} - \Omega_{n+1} H_n z_n \| = \min_{z \in \mathbb{C}^n} \| f_{n+1} - \Omega_{n+1} H_n z \|. \tag{2.26}$$

Here

$$f_{n+1} := \omega_1 \rho_1 \cdot [1 \quad 0 \quad \cdots \quad 0]^T \in \mathbb{R}^{n+1}, \tag{2.27}$$

with ρ_1 given in (2.3), and

$$\Omega_{n+1} := \operatorname{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1}), \quad \omega_j > 0, \quad j = 1, 2, \ldots, n+1, \tag{2.28}$$

is an arbitrary diagonal weighting matrix. Note that, in view of (2.17) and (2.28), the $(n+1) \times n$ matrix $\Omega_{n+1} H_n$ has full rank n. This guarantees that there always exists a unique solution of (2.26). Furthermore, we remark that the standard choice for the weights in (2.28) is

$$\omega_j = 1 \quad \text{for all } j. \tag{2.29}$$

However, there are instances (see [11]) where the use of different weights is crucial, and therefore we formulate the QMR method in the general setting (2.28).

From (2.25), (2.14), (2.27), and (2.3), it follows that the residual vector $r_n := b - Ax_n$ corresponding to x_n satisfies

$$r_n = V_{n+1} \Omega_{n+1}^{-1} \left(f_{n+1} - \Omega_{n+1} H_n z_n \right). \tag{2.30}$$

Hence, in view of (2.26), the nth QMR iterate x_n is characterized by a minimization of the second factor in (2.30); this is just the *quasi-minimal residual* property. The relation (2.30) shows that the scaling (2.29) is very natural, in the sense that all columns of $V_{n+1}\Omega_{n+1}^{-1}$ in the representation (2.30) of r_n are treated equally. We remark that the QMR iterates x_n can be easily updated from step to step. Due to the block-tridiagonal structure of H_n, this update can be implemented with only short recurrences; see [9] for details. Finally, we note that the QMR property can be used to derive convergence results for the QMR method; we refer the reader to [9] and [6].

3. A coupled two-term procedure with look-ahead. In this section, we consider a different approach to constructing the Lanczos vectors. The basic idea is to break up the three-term recurrences in the Lanczos process into coupled two-term recurrences, by using (in addition to the Lanczos vectors) a suitable second set of basis vectors for the underlying Krylov subspaces. In §9, we will illustrate that QMR based on this coupled two-term procedure has better numerical properties than the original implementation of QMR based on three-term recurrences.

3.1. The general setting. In the following, let $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ always denote the sequence of vectors generated by the look-ahead Lanczos algorithm as described in §2.2. We assume that we are also given a second set of basis vectors

$$\{p_j\}_{j=1}^n \quad \text{and} \quad \{q_j\}_{j=1}^n \tag{3.1}$$

for the Krylov subspaces $K_n(v_1, A)$ and $K_n(w_1, A^T)$. More precisely, we consider vectors (3.1) that, in analogy to (2.24), are of the form

$$p_j = \psi_{j-1}(A)v_1 \quad \text{and} \quad q_j = \gamma_j\psi_{j-1}(A^T)w_1, \tag{3.2}$$

where $\gamma_j > 0$ is given by (2.16), and $\psi_{j-1} \in \mathcal{P}_{j-1}$ is of exact degree $j-1$ with the same leading coefficient as the polynomial ϕ_{j-1} in (2.24). To distinguish between the two bases, we will often refer to the Lanczos vectors $\{v_j\}_{j=1}^n$ and $\{w_j\}_{j=1}^n$ as the *V–W sequence* and to the vectors (3.1) as the *P–Q sequence*.

From (2.24) and (3.2), we conclude that, for each $n = 1, 2, \ldots,$

$$p_n = v_n - \sum_{i=1}^{n-1} p_i u_{in}, \tag{3.3}$$

$$q_n = w_n - \sum_{i=1}^{n-1} q_i u_{in}(\gamma_n/\gamma_i), \tag{3.4}$$

with suitable coefficients $u_{in} \in \mathbb{C}$, $i = 1, 2, \ldots, n-1$. Similarly, in view of (2.23), (2.24), (3.2), and (2.16), we have

$$\tilde{v}_{n+1} = Ap_n - \sum_{i=1}^{n} v_i l_{in}, \quad \rho_{n+1} = \|\tilde{v}_{n+1}\|, \quad v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \tag{3.5}$$

$$\tilde{w}_{n+1} = A^T q_n - \sum_{i=1}^{n} w_i l_{in}(\gamma_n/\gamma_i), \quad \xi_{n+1} = \|\tilde{w}_{n+1}\|, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1}, \tag{3.6}$$

with suitable coefficients $l_{in} \in \mathbb{C}$, $i = 1, 2, \ldots, n$. Note that (3.3)–(3.6) are coupled recurrences for generating the P–Q and V–W sequences: first, p_n and q_n are computed by means

of (3.3)–(3.4), and then, the next Lanczos pair v_{n+1} and w_{n+1} is obtained from (3.5)–(3.6). Of course, it remains to specify the actual choice of the P–Q sequence. In order to minimize work and storage of previous vectors, the goal here is to select these vectors such that the recurrences (3.3)–(3.6) are as short as possible.

In addition to (2.8), it will be convenient to use the notation

$$P_n := [\, p_1 \quad p_2 \quad \cdots \quad p_n \,] \quad \text{and} \quad Q_n := [\, q_1 \quad q_2 \quad \cdots \quad q_n \,].$$

The recurrences (3.3)–(3.6) for the vectors $\{p_j\}_{j=1}^{n}$, $\{q_j\}_{j=1}^{n}$, $\{v_j\}_{j=1}^{n+1}$, and $\{w_j\}_{j=1}^{n+1}$ can then be written compactly in matrix form:

$$V_n = P_n U_n, \qquad A P_n = V_{n+1} L_n, \tag{3.7}$$

$$W_n = Q_n \Gamma_n^{-1} U_n \Gamma_n, \quad A^T Q_n = W_{n+1} \Gamma_{n+1}^{-1} L_n \Gamma_n. \tag{3.8}$$

Here, U_n is an upper triangular matrix and L_n is an upper Hessenberg matrix given by

$$U_n := \begin{bmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & u_{n-1,n} \\ 0 & \cdots & 0 & 1 \end{bmatrix} \quad \text{and} \quad L_n := \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1n} \\ \rho_2 & l_{22} & & \vdots \\ 0 & \rho_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & l_{nn} \\ 0 & \cdots & 0 & \rho_{n+1} \end{bmatrix}, \tag{3.9}$$

respectively, and Γ_n is the diagonal matrix defined in (2.16). Note that, by eliminating P_n in (3.7), we obtain

$$A V_n = V_{n+1} L_n U_n. \tag{3.10}$$

By comparing (3.10) with (2.14), it follows that

$$H_n = L_n U_n, \tag{3.11}$$

i.e., the matrices (3.9) define a factorization of the block-tridiagonal Hessenberg matrix H_n generated by the look-ahead Lanczos algorithm.

Recall from (2.7) and (2.18) that the Lanczos vectors are block biorthogonal. These biorthogonality relations determine the coefficients l_{in} in (3.5)–(3.6). For example, consider the case that v_{n+1} and w_{n+1} are constructed as regular vectors. Then, in view of (2.22), we have the condition $W_n^T v_{n+1} = 0$, which, by (3.5), is equivalent to

$$0 = W_n^T A p_n - \sum_{i=1}^{n} W_n^T v_i l_{in}. \tag{3.12}$$

Using (2.7), (2.8), and the first equation in (3.8), we deduce from (3.12) that

$$\begin{bmatrix} l_{1n} \\ \vdots \\ l_{nn} \end{bmatrix} = D_n^{-1} \Gamma_n U_n^T \Gamma_n^{-1} Q_n^T A p_n. \tag{3.13}$$

Recall from (2.18), (2.16), and (3.9) that the matrices D_n^{-1}, Γ_n, and U_n^T are block diagonal, diagonal, and lower triangular, respectively. Hence the relation (3.13) implies that the vector $Q_n^T A p_n$ determines the length of the recurrences (3.5)–(3.6). In particular, in order to obtain recursions that are as short as possible, the P–Q sequence should be chosen such that the vector

$Q_n^T A p_n$ has as many leading zeros as possible. The same conclusion also holds for the case that v_{n+1} and w_{n+1} are constructed as inner vectors.

Motivated by this discussion, we require that the vectors in the P–Q sequence be A-biorthogonal or block A-biorthogonal, in the sense that the matrix

$$E_n := Q_n^T A P_n \tag{3.14}$$

should be diagonal or block diagonal. Note that the vector $Q_n^T A p_n$ is just the nth column of E_n. Furthermore, we remark that $q_n^T A P_n$, the nth row of E_n, has the same zero structure as $Q_n^T A p_n$. This is a consequence of relation (3.16) in the following lemma, which we will also need later on.

LEMMA 3.1. *Let* $\{v_i\}_{i=1}^n$, $\{w_i\}_{i=1}^n$ *and* $\{p_i\}_{i=1}^n$, $\{q_i\}_{i=1}^n$ *be vectors satisfying* (2.24) *and* (3.2), *respectively, and let* $\Gamma_n = \operatorname{diag}(\gamma_1, \gamma_2, \ldots, \gamma_n)$. *Let* D_n *and* E_n *be the matrices given by* (2.7) *and* (3.14), *respectively. Then*

$$D_n \Gamma_n = (D_n \Gamma_n)^T, \tag{3.15}$$

$$E_n \Gamma_n = (E_n \Gamma_n)^T. \tag{3.16}$$

Proof. Using the polynomial representation (2.24) for the V–W vectors and the fact that polynomials in a matrix commute, we obtain

$$w_i^T v_j \gamma_j = \gamma_i w_1^T \phi_{i-1}(A) \phi_{j-1}(A) v_1 \gamma_j = \gamma_j w_1^T \phi_{j-1}(A) \phi_{i-1}(A) v_1 \gamma_i = w_j^T v_i \gamma_i$$

and thus (3.15). The relation (3.16) follows similarly. □

3.2. The coupled algorithm without look-ahead. Next, we briefly consider the case that the V–W sequence consists of the vectors generated by the classical Lanczos algorithm without look-ahead. Recall from (2.13) that here the matrix D_n in (2.7) is diagonal, and that the Lanczos vectors are biorthogonal:

$$w_j^T v_n = \begin{cases} 0 & \text{if } j \neq n, \\ \delta_n \neq 0 & \text{if } j = n. \end{cases} \tag{3.17}$$

Suppose that it is possible to construct the vectors in the P–Q sequence such that the matrix E_n in (3.14) is also diagonal:

$$E_n = \operatorname{diag}(\epsilon_1, \epsilon_2, \ldots, \epsilon_n), \quad \text{where } \epsilon_j := q_j^T A p_j \neq 0. \tag{3.18}$$

By (3.14) and (3.18), the P–Q vectors are then A-biorthogonal:

$$q_j^T A p_n = \begin{cases} 0 & \text{if } j \neq n, \\ \epsilon_n \neq 0 & \text{if } j = n. \end{cases} \tag{3.19}$$

With (2.13), (3.9), and (3.19), we deduce from (3.13) that

$$l_{in} = 0, \quad i = 1, 2, \ldots, n-1, \quad \text{and} \quad l_{nn} = \beta_n := \epsilon_n / \delta_n. \tag{3.20}$$

Furthermore, by multiplying (3.3) from the left by $q_j^T A$ and by using (3.19), (3.6), and (3.17), we obtain that, for $j = 1, 2, \ldots, n-1$,

$$\begin{aligned} 0 = q_j^T A p_n &= q_j^T A v_n - \epsilon_j u_{jn} \\ &= (A^T q_j)^T v_n - \epsilon_j u_{jn} \\ &= \xi_{j+1} w_{j+1}^T v_n - \epsilon_j u_{jn}. \end{aligned} \tag{3.21}$$

With (3.17), it follows from (3.21) that

$$u_{in} = 0, \quad i = 1, 2, \ldots, n-2, \quad \text{and} \quad u_{n-1,n} = \xi_n \delta_n / \epsilon_{n-1}. \tag{3.22}$$

In view of (3.20) and (3.22), all but the last terms vanish in each of the sums in (3.3)–(3.6), and hence (3.3)–(3.6) reduces to a coupled two-term procedure for generating the P–Q and V–W sequences.

The nth iteration of the resulting algorithm can be summarized as follows.

ALGORITHM 3.2 (nth iteration of the coupled algorithm without look-ahead).

1. If $\epsilon_{n-1} = 0$, then stop.
   Otherwise, compute $\delta_n = w_n^T v_n$.
   If $\delta_n = 0$, then stop.
2. Compute

$$\begin{aligned} p_n &= v_n - p_{n-1}(\xi_n \delta_n / \epsilon_{n-1}), \\ q_n &= w_n - q_{n-1}(\rho_n \delta_n / \epsilon_{n-1}). \end{aligned}$$

3. Compute $\epsilon_n = q_n^T A p_n$, $\beta_n = \epsilon_n / \delta_n$, and set

$$\begin{aligned} \tilde{v}_{n+1} &= A p_n - v_n \beta_n, \quad \rho_{n+1} = \|\tilde{v}_{n+1}\|, \\ \tilde{w}_{n+1} &= A^T q_n - w_n \beta_n, \quad \xi_{n+1} = \|\tilde{w}_{n+1}\|. \end{aligned}$$

4. If $\rho_{n+1} = 0$ or $\xi_{n+1} = 0$, then stop.
   Otherwise, set

$$v_{n+1} = \tilde{v}_{n+1} / \rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1} / \xi_{n+1}.$$

We remark that the vectors in the P–Q and V–W sequences are, up to scaling, just the search directions and the residual vectors generated by the BCG method [17], [4]. In particular, Algorithm 3.2 can be viewed as the nth iteration of a rescaled version of BCG, where the computation of the BCG iterates is omitted.

In exact arithmetic, one of the termination checks in steps 1 or 4 of the coupled two-term procedure will be satisfied after at most N iterations. Normally, the algorithm stops due to $\rho_{n+1} = 0$ or $\xi_{n+1} = 0$, and then the procedure has constructed a basis for the invariant subspaces $K_n(v_1, A)$ or $K_n(w_1, A^T)$, respectively. This is called regular termination. If $\epsilon_{n-1} = 0$ or $\delta_n = 0$ occurs, then the algorithm has to be stopped to avoid division by 0. This is referred to as an exact breakdown. Recall from §2.2 that $\delta_n = 0$ signals a breakdown in the classical Lanczos algorithm. Note that, like BCG, the coupled two-term procedure now has a second source of breakdown, namely, the case $\epsilon_{n-1} = 0$. It can be shown that the condition $\epsilon_{n-1} = 0$ corresponds to a breakdown in the BCG algorithm due to an nth iterate not being defined by the Galerkin condition.

In finite-precision arithmetic, exact breakdowns are rather unlikely. However, near-breakdowns, where δ_n or ϵ_{n-1} is nonzero, but small in some sense, may occur, leading to numerical instabilities in subsequent iterations. Next, we sketch a coupled two-term procedure that uses look-ahead in the construction of both the V–W and P–Q sequences to avoid exact and near-breakdowns.

3.3. The general algorithm with look-ahead. For describing the look-ahead in the V–W sequence, we will use the notations (2.18)–(2.21) introduced in §2.2. In particular, the integer $l := l(n)$ denotes the number of look-ahead steps that were performed during the construction of the first n vectors $\{v_i\}_{i=1}^n$ and $\{w_i\}_{i=1}^n$ in the V–W sequence, and the n_j's in (2.21) are the indices of the regular vectors. Recall that, by (2.7) and (2.18), the V–W vectors satisfy the block-biorthogonality conditions

$$(W^{(i)})^T V^{(j)} = \begin{cases} 0 & \text{if } i \neq j, \\ D^{(j)} & \text{if } i = j, \end{cases} \qquad i, j = 1, 2, \ldots, l. \tag{3.23}$$

Here, the blocks $D^{(j)}$ are all nonsingular, except for possibly $D^{(l)}$. However, we have that necessarily

$$D^{(l)} \text{ is nonsingular, if } n_{l+1} = n + 1. \tag{3.24}$$

Next, we introduce similar notations for describing the look-ahead in the P–Q sequence. We denote by $k := k(n)$ the number of look-ahead steps that were performed during the construction of the first $n-1$ vectors $\{p_i\}_{i=1}^{n-1}$ and $\{q_i\}_{i=1}^{n-1}$ in the P–Q sequence. In analogy to (2.19), we partition these vectors into blocks, according to the look-ahead steps taken:

$$P_{n-1} = \left[P^{(1)} \quad P^{(2)} \quad \cdots \quad P^{(k)} \right] \quad \text{and} \quad Q_{n-1} = \left[Q^{(1)} \quad Q^{(2)} \quad \cdots \quad Q^{(k)} \right]. \tag{3.25}$$

Recall from (3.14) that the P–Q vectors are constructed to be block A-biorthogonal. More precisely, we have

$$E_{n-1} = \operatorname{diag}(E^{(1)}, E^{(2)}, \ldots, E^{(k)}),$$

or, equivalently,

$$(Q^{(i)})^T A P^{(j)} = \begin{cases} 0 & \text{if } i \neq j, \\ E^{(j)} & \text{if } i = j, \end{cases} \qquad i, j = 1, 2, \ldots, k. \tag{3.26}$$

Here, the blocks $E^{(j)}$ are nonsingular, except for possibly the last block $E^{(k)}$. In analogy to (2.21), we denote by m_j the indices of the first vectors of the blocks $P^{(j)}$ and $Q^{(j)}$ in (3.25). Hence, for $j = 1, 2, \ldots, k$, we have

$$P^{(j)} = \left[p_{m_j} \quad p_{m_j+1} \quad \cdots \quad \right] \quad \text{and} \quad Q^{(j)} = \left[q_{m_j} \quad q_{m_j+1} \quad \cdots \quad \right]. \tag{3.27}$$

Furthermore, the indices m_j satisfy

$$1 =: m_1 < m_2 < \cdots < m_k < n \leq m_{k+1}. \tag{3.28}$$

We remark that, by (3.26) and (3.27), the vectors p_{m_j} and q_{m_j} are A-biorthogonal to all previous P–Q vectors, i.e.,

$$q_i^T A p_{m_j} = q_{m_j}^T A p_i = 0 \quad \text{for all } i = 1, 2, \ldots, m_j - 1.$$

Therefore, using the same notation as for the V–W sequence, we refer to the vectors p_{m_j} and q_{m_j} as *regular* vectors, while the remaining vectors in (3.27) are called *inner*. Finally, it turns out that, in (3.26), the last block $E^{(k)}$ has to be nonsingular, if p_n and q_n are constructed as regular vectors. This means that, in analogy to (3.24),

$$E^{(k)} \text{ is nonsingular, if } m_{k+1} = n. \tag{3.29}$$

After these preliminaries, we can now sketch the actual algorithm. Let $n \geq 1$, and assume that we have already generated the first $n-1$ vectors $\{p_i\}_{i=1}^{n-1}$ and $\{q_i\}_{i=1}^{n-1}$ of the P–Q sequence, and the first n vectors $\{v_i\}_{i=1}^{n}$ and $\{w_i\}_{i=1}^{n}$ of the V–W sequence.

First, the next pair of P–Q vectors, p_n and q_n, is constructed, using the recurrences (3.3)–(3.4). Here, the coefficients u_{in} need to be chosen such that p_n satisfies the corresponding A-biorthogonality conditions (3.26). We note that, in view of (3.16), the A-biorthogonality relations for the vector q_n are then also fulfilled. With (3.6) and (3.23), one readily verifies that

$$q_i^T A v_n = (A^T q_i)^T v_n = 0 \quad \text{for all } i = 1, 2, \ldots, n_l - 2.$$

By rewriting this in terms of the blocks $Q^{(j)}$, we obtain that

$$(Q^{(j)})^T A v_n = 0 \quad \text{for all } j = 1, 2, \ldots, k^\star - 1, \tag{3.30}$$

where $k^\star := k^\star(n)$ is given by

$$k^\star = \max\left\{ j \;\middle|\; 1 \leq j \leq k \text{ and } m_j \leq \max\{1, n_l - 1\} \right\}. \tag{3.31}$$

The orthogonality conditions (3.30), together with (3.26), imply that, in (3.3), we have $u_{in} = 0$ for all $i < m_{k^\star}$. Thus the recurrences (3.3)–(3.4) reduce to

$$p_n = v_n - \sum_{i=m_{k^\star}}^{n-1} p_i u_{in}, \tag{3.32}$$

$$q_n = w_n - \sum_{i=m_{k^\star}}^{n-1} q_i u_{in} (\gamma_n / \gamma_i). \tag{3.33}$$

Now we need to determine the coefficients u_{in} for $m_{k^\star} \leq i \leq n-1$. This is done by enforcing the remaining A-biorthogonality conditions (3.26) for p_n and the vectors q_i, namely,

$$q_i^T A p_n = 0 \quad \text{for all } i = m_{k^\star}, m_{k^\star} + 1, \ldots, m^\star. \tag{3.34}$$

Here, $m^\star := n-1$ if p_n and q_n are constructed as regular vectors, and $m^\star := m_k - 1$ otherwise. Note that, in view of (3.28), we always have $m_k - 1 \leq m^\star$. First, we consider (3.34) for the indices i in the range $m_{k^\star} \leq i \leq m_k - 1$. Using (3.32) and (3.26), we deduce from (3.34) that

$$U_{m_{k^\star}:m_k-1,n} = (\operatorname{diag}(E^{(k^\star)}, \ldots, E^{(k-1)}))^{-1} \left[Q^{(k^\star)} \quad \cdots \quad Q^{(k-1)} \right]^T A v_n. \tag{3.35}$$

Here, and in the sequel, we use the notation

$$M_{i:j,n} := [\, m_{in} \quad m_{i+1,n} \quad \cdots \quad m_{jn} \,]^T$$

for vectors consisting of successive elements of the nth column of the matrix $M = [\, m_{ij} \,]$. If p_n and q_n are constructed as regular vectors, then we also have to ensure that (3.34) holds for i with $m_k \leq i \leq n-1$, and this gives

$$U_{m_k:n-1,n} = (E^{(k)})^{-1} (Q^{(k)})^T A v_n. \tag{3.36}$$

We remark that, by (3.29), the matrix $E^{(k)}$ in (3.36) is necessarily nonsingular since the vectors p_n and q_n are regular. If p_n and q_n are constructed as inner vectors, then $m^\star = m_k - 1$,

and (3.34) yields no conditions for the choice of the coefficients u_{in} with $m_k \le i < n-1$. In this case, we set

$$u_{in} = \zeta_{in} \quad \text{for } i = m_k, m_k+1, \ldots, n-1, \tag{3.37}$$

where $\zeta_{in} \in \mathbb{C}$ can be chosen arbitrarily. Finally, if p_n and q_n are regular, we update the "regular" indices (3.28) by setting $m_{k+1} := n$ and $k := k+1$. This completes the construction of the p_n and q_n vectors.

In a second step, we now compute the next pair of V–W vectors, v_{n+1} and w_{n+1}, using the recurrences (3.5)–(3.6). Here, we have to determine the coefficients l_{in} such that v_{n+1} satisfies the corresponding biorthogonality conditions (3.23). Note that, in view of (3.15), the relations (3.23) for w_{n+1} are then fulfilled automatically. We proceed similar to the construction of p_n and q_n. By using (3.26) and the fact that the columns of the matrices Q_i and W_i span the same space, it is easily verified that

$$(W^{(j)})^T A p_n = 0 \quad \text{for all } j = 1, 2, \ldots, l^\star - 1, \tag{3.38}$$

where $l^\star := l^\star(n)$ is given by

$$l^\star = \max\left\{ j \;\middle|\; 1 \le j \le l \text{ and } n_j \le m_k \right\}. \tag{3.39}$$

From (3.38), we conclude that the recurrences (3.5)–(3.6) reduce as follows:

$$\tilde{v}_{n+1} = A p_n - \sum_{i=n_{l^\star}}^{n} v_i l_{in}, \tag{3.40}$$

$$\tilde{w}_{n+1} = A^T q_n - \sum_{i=n_{l^\star}}^{n} w_i l_{in} (\gamma_n/\gamma_i). \tag{3.41}$$

The recurrence coefficients l_{in} in (3.40)–(3.41) are determined by enforcing the remaining biorthogonality conditions (3.23) for the vectors v_{n+1} and w_i, $n_{l^\star} \le i \le n$. This gives

$$L_{n_{l^\star}:n_l-1,n} = (\operatorname{diag}(D^{(l^\star)}, \ldots, D^{(l-1)}))^{-1} \left[W^{(l^\star)} \quad \ldots \quad W^{(l-1)} \right]^T A p_n. \tag{3.42}$$

Moreover, if v_{n+1} and w_{n+1} are constructed as regular vectors, then

$$L_{n_l:n,n} = (D^{(l)})^{-1} (W^{(l)})^T A p_n. \tag{3.43}$$

Note that, by (3.24), the matrix $D^{(l)}$ in (3.43) is necessarily nonsingular since v_{n+1} and w_{n+1} are regular. If v_{n+1} and w_{n+1} are built as inner vectors, then we set

$$l_{in} = \eta_{in} \quad \text{for } i = n_l, n_l+1, \ldots, n, \tag{3.44}$$

where $\eta_{in} \in \mathbb{C}$ can be chosen arbitrarily. Finally, if v_{n+1} and w_{n+1} are regular, then we update the indices (2.21) by setting $n_{l+1} := n+1$ and $l := l+1$.

The resulting coupled procedure for generating the P–Q and V–W sequences can be sketched as follows.

ALGORITHM 3.3 (Coupled algorithm with look-ahead).

0. Choose $v_1, w_1 \in \mathbb{C}^N$ with $\|v_1\| = \|w_1\| = 1$.
   Set $V^{(1)} = v_1$, $W^{(1)} = w_1$, $D^{(1)} = w_1^T v_1$.
   Set $k = 1, m_k = 1, l = 1, n_l = 1$.

For $n = 1, 2, \ldots,$ do:

1. Determine $k^\star$ from (3.31).
2. Decide whether to construct p_n and q_n as regular or inner vectors and go to step 3 or 4, respectively.
3. Compute p_n and q_n by means of (3.35)–(3.36) and (3.32)–(3.33).
   Set $m_{k+1} = n$, $k = k + 1$, $P^{(k)} = Q^{(k)} = \emptyset$ and go to step 5.
4. Compute p_n and q_n by means of (3.35), (3.37), and (3.32)–(3.33).
5. Set
$$P^{(k)} = \left[P^{(k)} \quad p_n \right], \quad Q^{(k)} = \left[Q^{(k)} \quad q_n \right], \quad E^{(k)} = (Q^{(k)})^T A P^{(k)}.$$
6. Determine $l^\star$ from (3.39).
7. Decide whether to construct v_{n+1} and w_{n+1} as regular or inner vectors and go to step 8 or 9, respectively.
8. Compute $\tilde{v}_{n+1}$ and $\tilde{w}_{n+1}$ by means of (3.42)–(3.43) and (3.40)–(3.41).
   Set $n_{l+1} = n + 1$, $l = l + 1$, $V^{(l)} = W^{(l)} = \emptyset$ and go to step 10.
9. Compute $\tilde{v}_{n+1}$ and $\tilde{w}_{n+1}$ by means of (3.42), (3.44), and (3.40)–(3.41).
10. Compute $\rho_{n+1} = \|\tilde{v}_{n+1}\|$ and $\xi_{n+1} = \|\tilde{w}_{n+1}\|$.
    If $\rho_{n+1} = 0$ or $\xi_{n+1} = 0$, then stop.
    Otherwise, set
$$v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1}/\xi_{n+1},$$
$$V^{(l)} = \left[V^{(l)} \quad v_{n+1} \right], \quad W^{(l)} = \left[W^{(l)} \quad w_{n+1} \right], \quad D^{(l)} = (W^{(l)})^T V^{(l)}.$$

We remark that Algorithm 3.3 reduces to the coupled two-term procedure described in § 3.2 if all vectors in the P–Q and V–W sequences are built as regular vectors. Note that in this case, we have $k(n) = n - 1$, $n_{k^\star} = n - 1$, $l(n) = n$, and $n_{l^\star} = n$ for all n.

4. The look-ahead strategy. As described in §3.2, there are two possible breakdowns in the coupled two-term procedure without look-ahead: one associated with the V–W sequence, and another associated with the P–Q sequence. In particular, Algorithm 3.2 will encounter an exact breakdown in the V–W sequence if $w_n^T v_n = 0$, or in the P–Q sequence if $q_{n-1}^T A p_{n-1} = 0$. The exact breakdowns of the two sequences are not independent of each other, as was pointed out by Gutknecht in [12]. For a full description of the structure and coupling of the exact breakdowns, we refer the reader to [12] and [2], and the references given there. However, in practice one is also concerned with avoiding near-breakdowns; that is, situations when $w_n^T v_n$ or $q_{n-1}^T A p_{n-1}$ are not exactly zero, but are small in some sense.

In the coupled procedure with look-ahead, which we sketched in §3.3, exact and near-breakdowns in the P–Q, respectively V–W, sequence are prevented by building the next pair of vectors p_n and q_n, respectively v_{n+1} and w_{n+1}, as inner vectors. In this section, we describe the look-ahead strategy that is used to decide in steps 2 and 7 of Algorithm 3.3 whether vectors are constructed as regular or inner vectors.

Recall from §2.2 that the vectors $\{v_i\}_{i=1}^{n+1}$ and $\{w_i\}_{i=1}^{n+1}$ in the look-ahead Lanczos algorithm satisfy a block three-term recurrence that can be written compactly as (2.14)–(2.15). By eliminating V_n and W_n in (3.7) and (3.8), one obtains a similar recurrence relation for the vectors $\{p_i\}_{i=1}^{n}$ and $\{q_i\}_{i=1}^{n}$ of the P–Q sequence:

$$A P_{n-1} = P_n U_n L_{n-1} \quad \text{and} \quad A^T Q_{n-1} = Q_n \Gamma_n^{-1} U_n L_{n-1} \Gamma_{n-1}. \tag{4.1}$$

By using the A-biorthogonality of p_n and q_n, it is easy to show that the recurrences for the P–Q sequence are also three-term recurrences, or, in the general case, block three-term recurrences. We note that, in (4.1), the matrix $U_n L_{n-1}$ of recurrence coefficients is obtained by multiplying the factors L_{n-1} and U_n from the decompositions (3.11) of H_{n-1} and H_n, respectively, in reverse order. This was first remarked by Rutishauser [21] for the special case of no look-ahead, and recently by Gutknecht [12], for the general case.

The look-ahead strategy consists of monitoring breakdowns in the two sequences independently. For the V–W sequence, the criteria used are the same as those proposed in [7]:

$$\sigma_{\min}(D^{(l)}) \geq \text{ eps}, \tag{4.2}$$

$$n(A) \geq \sum_i \left|(L_n U_n)_{in}\right|, \tag{4.3}$$

$$n(A) \geq \sum_i \frac{\gamma_n}{\gamma_i} \left|(L_n U_n)_{in}\right|, \tag{4.4}$$

where eps is machine epsilon, and $n(A)$ is an estimate for the norm of A. The Lanczos vectors v_{n+1} and w_{n+1} are built as regular vectors only if all three of the above checks are satisfied. The check (4.2) ensures that the diagonal blocks $D^{(j)}$ are nonsingular, while the checks (4.3)–(4.4) ensure that the size of the coefficients μ_n and ν_n in (2.23) and in the corresponding relation for $\tilde{w}_{n+1}$ do not exceed an estimate $n(A)$ for the norm of A. The first condition is needed since the inverse of $D^{(l)}$ appears in μ_n, while the second and third conditions attempt to ensure that the components from $K_n(r_0, A)$ and from $K_n(w_1, A^T)$ do not dominate the Av_n and $A^T w_n$ terms, respectively. Another motivation for these checks is as follows. The symmetric Lanczos process for Hermitian matrices A generates tridiagonal matrices H_n that satisfy

$$\|H_n\| \leq \|A\| \quad \text{for all } n. \tag{4.5}$$

For the classical nonsymmetric Lanczos algorithm, the relation (4.5) does not hold in general. Indeed, formally, we have $\|H_n\| = \infty$ if a breakdown occurs. As David Day has pointed out to us,[1] an "ideal" look-ahead Lanczos procedure would ensure that (4.5) also holds for non-Hermitian matrices. The criteria (4.3)–(4.4) can be viewed as a cheap way of modeling the conditions (4.5). We remark that the checks (4.3)–(4.4) take advantage of the normalization (2.9) of the Lanczos vectors.

For the P–Q sequence, the criteria are similar: the diagonal blocks $E^{(j)}$ must be nonsingular, and the size of the last columns of $U_n L_{n-1}$ and of $\Gamma_n^{-1} U_n L_{n-1} \Gamma_{n-1}$ must not exceed the estimate $n(A)$ for the norm of A. Singularity of $E^{(k)}$ is once again checked from its smallest singular value:

$$\sigma_{\min}(E^{(k)}) \geq \text{ eps}.$$

However, for the second and third checks, it is no longer sufficient to compute just the norm of the last column of the matrices of recurrence coefficients, as the vectors p_n and q_n are not normalized to unit length. Instead, one must check

$$n(A)\|p_n\| \geq \sum_i \left|(U_n L_{n-1})_{i,n-1}\right| \|p_i\| \tag{4.6}$$

and

$$n(A)\|q_n\| \geq \sum_i \frac{\gamma_{n-1}}{\gamma_i} \left|(U_n L_{n-1})_{i,n-1}\right| \|q_i\|. \tag{4.7}$$

[1] Private communication, Berkeley, March 1992.

This means that the look-ahead strategy for the P–Q sequence requires the computation of the two norms $\|p_n\|$ and $\|q_n\|$ at each step, work that would otherwise not be needed by the algorithm. Once again, the vectors p_n and q_n are built as regular vectors only if all three of the above checks are satisfied. We remark that the look-ahead strategy presented here builds regular vectors in preference to inner vectors and will therefore build as few inner vectors as possible.

Finally, we note that other look-ahead strategies are also possible. For example, Gutknecht [12] proposed a look-ahead strategy that assumes that the near-breakdowns encountered in the two sequences have the same structure as the exact breakdowns. We have chosen to monitor the two sequences independently; nevertheless, our strategy will recover the exact-breakdown structure if only exact breakdowns are considered.

5. Implementation details. In this section, we discuss some of the details of an implementation of Algorithm 3.3. The nth iteration of Algorithm 3.3 updates the matrices E_{n-1}, L_{n-1}, U_{n-1}, P_{n-1}, Q_{n-1}, V_n, W_n, and D_n, to E_n, L_n, U_n, P_n, Q_n, V_{n+1}, W_{n+1}, and D_{n+1}, respectively. We are interested in obtaining an implementation that requires only two inner products per iteration to compute all the coefficients of the recurrence formulas. Recall that the look-ahead strategy (4.6)–(4.7) for the P–Q sequence and the normalization (2.9) require a total of four norm computations, so that the implementation will require two inner products and four norms per iteration.

Let us introduce the auxiliary matrices

$$F_n = W_n^T A P_n \quad \text{and} \quad \tilde{F}_n = Q_n^T A V_n,$$

whose columns are needed in (3.42)–(3.43) and (3.35)–(3.36). It turns out that these two matrices are essentially the transpose of each other.

LEMMA 5.1. *The matrices F_n and $\tilde{F}_n$ satisfy $F_n \Gamma_n = (\tilde{F}_n \Gamma_n)^T$.*

Proof. The proof is similar to the proof of Lemma 3.1. □

At the beginning of each iteration, we will have available the $n \times (n-1)$ matrix $F_{1:n,1:n-1}$, which we will update to $F_{1:n+1,1:n}$.

To compute the coefficients u_{in} needed in (3.32)–(3.33), $\tilde{F}_{1:n-1,n}$ is obtained by Lemma 5.1. The vectors p_n and q_n are then computed from (3.32)–(3.33). To obtain E_n from E_{n-1}, the diagonal term $q_n^T A p_n$ is computed directly, requiring one inner product. Then, using

$$F_n = W_n^T A P_n = \Gamma_n U_n^T \Gamma_n^{-1} Q_n^T A P_n = \Gamma_n U_n^T \Gamma_n^{-1} E_n, \tag{5.1}$$

the remainder of the last row of E_n is computed from E_{n-1}, $F_{n,1:n-1}$, and U_{n-1}. The last column of E_n is obtained by symmetry, using (3.16) from Lemma 3.1. One then computes $F_{1:n,n}$, using (5.1) and the new column $E_{1:n,n}$. The vectors v_{n+1} and w_{n+1} are then computed from (3.40)–(3.41).

Next, we consider the update of D_{n+1} from D_n. The diagonal term $w_{n+1}^T v_{n+1}$ is once again computed directly, thus requiring the second inner product per iteration. Next, using

$$\begin{aligned} F_n &= W_n^T A P_n = W_n^T V_{n+1} L_n \\ &= D_n L_{1:n,1:n} + l_{n+1,n} D_{1:n,n+1} [\,0 \quad \cdots \quad 0 \quad 1\,], \end{aligned} \tag{5.2}$$

the remainder of the last column of D_{n+1} is computed from D_n, F_n, and L_n. The last row of D_{n+1} is obtained by symmetry, using (3.15) from Lemma 3.1. One then computes $F_{n+1,1:n}$, using (5.2) and the new row $D_{n+1,1:n+1}$.

Thus, the coupled Lanczos Algorithm 3.3 requires the computation of two inner products and four vector norms per iteration. We conclude this section by noting that, in Algorithm 3.3,

the choice of the inner recurrence coefficients (3.37) and (3.44) is arbitrary. In our implementation of the algorithm, we used

$$\begin{aligned} u_{n-1,n} &= 1, \\ u_{n-2,n} &= 1 \quad \text{when } m_k \leq n-2, \\ u_{in} &= 0 \quad \text{for } i = m_k, \ldots, n-3, \\ l_{nn} &= 1, \\ l_{n-1,n} &= 1 \quad \text{when } n_l \leq n-1, \\ l_{in} &= 0 \quad \text{for } i = n_l, \ldots, n-2, \end{aligned}$$

for the inner-vector recurrence coefficients.

6. An implementation of QMR with look-ahead. We now return to linear systems (1.1) and the QMR method. In this section, we propose an implementation of the QMR method based on the coupled two-term look-ahead Algorithm 3.3.

Recall that the nth QMR iterate x_n is defined by (2.25)–(2.26) in terms of the matrices V_n and H_n generated by the look-ahead Lanczos algorithm. In the original implementation of QMR, the solution z_n of the least-squares problem (2.26) is computed by means of a QR decomposition of the matrix $\Omega_{n+1}H_n$.

Here we consider the case that the Lanczos vectors are constructed using the coupled two-term Algorithm 3.3. Recall that Algorithm 3.3 yields as a by-product the factors L_n and U_n in the decomposition (3.11) of H_n. Using the factorization (3.11) and setting $y_n := U_n z_n$, we can rewrite the definition (2.25)–(2.26) of x_n as follows:

$$x_n = x_0 + V_n U_n^{-1} y_n, \tag{6.1}$$

where y_n is the unique solution of the least-squares problem

$$\| f_{n+1} - \Omega_{n+1} L_n y_n \| = \min_{y \in \mathbb{C}^n} \| f_{n+1} - \Omega_{n+1} L_n y \|. \tag{6.2}$$

Here, as before, f_{n+1} is given by (2.27) and Ω_{n+1} is defined in (2.28). We remark that the least-squares problem (6.2) is actually cheaper to solve than the original one (2.26). The reason is that the matrix L_n in (6.2) has fewer nonzero elements than H_n in (2.26). For example, if no look-ahead steps are taken, then H_n is tridiagonal, while L_n is a lower bidiagonal matrix. This special case will be considered in more detail in §7. We remark that, typically, the coefficient matrix $\Omega_{n+1}L_n$ of (6.2) is better conditioned than the matrix $\Omega_{n+1}H_n$ in (2.26). Consequently, the least-squares problem (6.2) usually can be solved to higher accuracy than the original one (2.26); see the examples in §9. This is another advantage of the coupled two-term implementation of QMR.

As discussed in [9], solutions of least-squares problems of the type (6.2) can be easily updated from step to step, using the QR decomposition of $\Omega_{n+1}L_n$,

$$\Omega_{n+1} L_n = Q_n^H \begin{bmatrix} R_n \\ 0 \end{bmatrix}, \tag{6.3}$$

where Q_n is a unitary $(n+1) \times (n+1)$ matrix, and R_n is a nonsingular upper triangular $n \times n$ matrix. With this, the least-squares problem (6.2) becomes

$$\begin{aligned} \min_{y \in \mathbb{C}^n} \| f_{n+1} - \Omega_{n+1} L_n y \| &= \min_{y \in \mathbb{C}^n} \left\| Q_n^H \left(Q_n f_{n+1} - \begin{bmatrix} R_n \\ 0 \end{bmatrix} y \right) \right\| \\ &= \min_{y \in \mathbb{C}^n} \left\| Q_n f_{n+1} - \begin{bmatrix} R_n \\ 0 \end{bmatrix} y \right\|. \end{aligned}$$

For y_n this gives:

$$y_n = R_n^{-1} t_n, \quad \text{where } t_n = \begin{bmatrix} \tau_1 \\ \vdots \\ \tau_n \end{bmatrix}, \quad \begin{bmatrix} t_n \\ \tilde{\tau}_{n+1} \end{bmatrix} := Q_n f_{n+1}. \tag{6.4}$$

Finally, we note that it is possible to update the QMR iterate at each step, as was done in the original QMR algorithm. Full implementation details were given in [9, §4], and we will not repeat them here. The point is that the QMR iterates have an update formula of the form

$$x_n = x_{n-1} + d_n \tau_n. \tag{6.5}$$

Here, τ_n is given by (6.4), and d_n is an auxiliary search direction defined as the last column of the matrix $V_n U_n^{-1} R_n^{-1} = P_n R_n^{-1}$, which appears in (6.1) after inserting y_n from (6.4). The vectors d_n are also updated with short recurrences: the recurrence for d_n involves only as many vectors as the recurrence for v_{n+1}. For full details of the update procedure for d_n, we refer the reader to [9].

The basic outline of the resulting implementation of QMR based on the coupled two-term Algorithm 3.3 is then as follows.

ALGORITHM 6.1 (QMR based on coupled recurrences).

0. Choose $x_0 \in \mathbb{C}^N$ and set $r_0 = b - Ax_0$, $\rho_1 = \|r_0\|$, $v_1 = r_0/\rho_1$.
   Choose $w_1 \in \mathbb{C}^N$ with $\|w_1\| = 1$.

For $n = 1, 2, \ldots$, do:

1. Perform the nth iteration of the coupled two-term Algorithm 3.3.
   This yields matrices L_n, P_n, U_n, and V_{n+1}, which satisfy (3.7).
2. Update the QR factorization (6.3) of $\Omega_{n+1} L_n$ and the vector t_n in (6.4).
3. Update the QMR iterate x_n by means of (6.5).
4. If x_n has converged, then stop.

In [9], various properties of the QMR method are given. For example, it is shown how existing BCG iterates can be easily recovered from the QMR process and how estimates for the QMR residual norms can be obtained at no extra costs. We would like to stress that these properties also hold true for the particular implementation of QMR sketched in Algorithm 6.1. Finally, recall from (2.29) that we recommend the use of unit weights $\omega_j = 1$ in $\Omega_{n+1} = \operatorname{diag}(\omega_1, \omega_2, \ldots, \omega_{n+1})$.

7. An implementation of QMR without look-ahead. In this section, we present the simplification of Algorithm 6.1 to the case where no look-ahead is used. We also briefly address the issue of preconditioning.

Let M be a given nonsingular $N \times N$ matrix that approximates in some sense the coefficient matrix A of (1.1). Suppose further that M is decomposed as

$$M = M_1 M_2. \tag{7.1}$$

Then, one applies the QMR algorithm to the system

$$A' y' = b', \tag{7.2}$$

where $A' = M_1^{-1} A M_2^{-1}$, $b' = M_1^{-1} b$, and $x' = M_2 x$. It is easy to see that the linear systems (1.1) and (7.2) are equivalent, and that one can transform back from the iterates x'_n and the

residuals r'_n of the system (7.2) to the iterates x_n and the residuals r_n of the system (1.1). For example, while applying QMR to the preconditioned system (7.2), it is possible to write the resulting algorithm in terms of the quantities corresponding to the original system (1.1); this is what is done below.

We now present a version of the QMR algorithm based on the coupled Algorithm 3.2, which does not have look-ahead. We remark that in this case, by (3.22) and (3.20), the matrix U_n in (3.9) is upper bidiagonal, and L_n is a lower bidiagonal matrix. We also implement preconditioning, as discussed above. The resulting QMR algorithm is as follows.

ALGORITHM 7.1 (QMR based on coupled recurrences without look-ahead).

0. Choose $x_0 \in \mathbb{C}^N$ and set $r_0 = b - Ax_0$.
   Compute $\rho_1 = \|M_1^{-1} r_0\|$ and set $v_1 = r_0/\rho_1$.
   Choose $w_1 \in \mathbb{C}^N$ with $\|M_2^{-T} w_1\| = 1$.
   Set $p_0 = q_0 = d_0 = 0$, $c_0 = \epsilon_0 = \xi_1 = 1$, $\vartheta_0 = 0$, $\eta_0 = -1$.

For $n = 1, 2, \ldots,$ do:

1. If $\epsilon_{n-1} = 0$, then stop.
   Compute $\delta_n = w_n^T M^{-1} v_n$. If $\delta_n = 0$, then stop.

2. Compute

$$p_n = M^{-1} v_n - p_{n-1}(\xi_n \delta_n / \epsilon_{n-1}),$$
$$q_n = M^{-T} w_n - q_{n-1}(\rho_n \delta_n / \epsilon_{n-1}).$$

3. Compute $\epsilon_n = q_n^T A p_n$, $\beta_n = \epsilon_n / \delta_n$, and set

$$\tilde{v}_{n+1} = Ap_n - v_n \beta_n, \qquad \rho_{n+1} = \|M_1^{-1} \tilde{v}_{n+1}\|,$$
$$\tilde{w}_{n+1} = A^T q_n - w_n \beta_n, \qquad \xi_{n+1} = \|M_2^{-T} \tilde{w}_{n+1}\|.$$

4. Compute

$$\vartheta_n = \frac{\omega_{n+1} \rho_{n+1}}{\omega_n c_{n-1} |\beta_n|}, \quad c_n = \frac{1}{\sqrt{1 + \vartheta_n^2}}, \quad \eta_n = -\eta_{n-1} \frac{\rho_n c_n^2}{\beta_n c_{n-1}^2},$$
$$d_n = p_n \eta_n + d_{n-1} (\vartheta_{n-1} c_n)^2, \quad x_n = x_{n-1} + d_n.$$

5. If $\rho_{n+1} = 0$ or $\xi_{n+1} = 0$, then stop.
   Otherwise, set

$$v_{n+1} = \tilde{v}_{n+1} / \rho_{n+1}, \quad w_{n+1} = \tilde{w}_{n+1} / \xi_{n+1}.$$

As pointed out in §3.2, the vectors computed in steps 2 and 3 of Algorithm 7.1 are rescaled versions of vectors used in BCG. Freund and Szeto [11] used this connection to derive an implementation of QMR without look-ahead that is directly based on BCG. Their Algorithm 3.1 in [11] and Algorithm 7.1 are mathematically equivalent.

8. QMR for complex symmetric matrices. In this section, we briefly discuss the application of the QMR Algorithm 7.1 to the solution of complex symmetric linear systems, i.e., systems with

$$A = A^T \in \mathbb{C}^{N \times N}.$$

We note that the QMR approach was originally proposed by Freund in [5] for exactly this class of linear systems. We stress that the implementation of complex symmetric QMR in [5] is based on the three-term Lanczos recurrence. Here we present a different implementation based on coupled two-term recursions.

The benefit of applying a Lanczos method to complex symmetric systems is that the underlying Lanczos algorithm simplifies naturally. Recall that in the coupled Algorithm 3.3, the second starting vector w_1 is arbitrary. In the case of a symmetric matrix, if w_1 is chosen equal to v_1, then it is easy to show that $w_n = v_n$ and $q_n = p_n$ for all n. Thus, the recurrences for w_n and q_n can be eliminated, saving roughly half the amount of work and storage.

However, we remark that, in contrast to the Lanczos algorithm for Hermitian matrices where breakdowns are excluded, the Lanczos process for complex symmetric also requires look-ahead to avoid exact and near-breakdowns. This is discussed in detail in [5].

The only other issue in the case of complex symmetric systems is that the preconditioner M in (7.1) must also be symmetric, i.e.,

$$M = M_1 M_2 = (M_1 M_2)^T = M^T. \tag{8.1}$$

For example, this is always guaranteed if the decomposition (7.1) is "symmetric" in the sense that

$$M_2 = M_1^T. \tag{8.2}$$

However, we stress that the condition (8.2) is not necessary, and M_1 and M_2 can be arbitrary matrices satisfying (8.1). We remark that standard preconditioning techniques, such as incomplete factorization, yield symmetric preconditioners (8.1) when applied to symmetric matrices A.

In this case, one can apply the QMR Algorithm 7.1 to the resulting preconditioned system. Once again writing everything in terms of the quantities corresponding to the original system (1.1), one obtains the following iteration.

ALGORITHM 8.1 (QMR without look-ahead for complex symmetric systems).

0. Choose $x_0 \in \mathbb{C}^N$ and set $r_0 = b - Ax_0$.
   Compute $\rho_1 = \|M_1^{-1} r_0\|$ and set $v_1 = r_0/\rho_1$.
   Set $p_0 = d_0 = 0$, $c_0 = \epsilon_0 = 1$, $\vartheta_0 = 0$, $\eta_0 = -1$.

For $n = 1, 2, \ldots$, do:

1. If $\epsilon_{n-1} = 0$, then stop.
   Compute $\delta_n = v_n^T M^{-1} v_n$. If $\delta_n = 0$, then stop.
2. Compute
$$p_n = M^{-1} v_n - p_{n-1}(\rho_n \delta_n / \epsilon_{n-1}).$$
3. Compute $\epsilon_n = p_n^T A p_n$, $\beta_n = \epsilon_n/\delta_n$, and
$$\tilde{v}_{n+1} = Ap_n - v_n \beta_n, \quad \rho_{n+1} = \|M_1^{-1} \tilde{v}_{n+1}\|.$$
4. Compute
$$\vartheta_n = \frac{\omega_{n+1}\rho_{n+1}}{\omega_n c_{n-1} |\beta_n|}, \quad c_n = \frac{1}{\sqrt{1+\vartheta_n^2}}, \quad \eta_n = -\eta_{n-1} \frac{\rho_n c_n^2}{\beta_n c_{n-1}^2},$$
$$d_n = p_n \eta_n + d_{n-1}(\vartheta_{n-1} c_n)^2, \quad x_n = x_{n-1} + d_n.$$

5. If $\rho_{n+1} = 0$, then stop.
   Otherwise, set

$$v_{n+1} = \tilde{v}_{n+1}/\rho_{n+1}.$$

9. Numerical experiments. In this section, we present a few numerical examples. We compare the original and the new implementation of the QMR algorithm, as well as illustrate an application of the coupled QMR algorithm to the solution of complex symmetric linear systems.

In Figs. 9.1–9.3 below, we always show the true relative residual norm $\|r_n\|/\|r_0\|$ plotted versus the iteration index n. All examples were run on a Sun SparcStation 2 using double precision, with machine epsilon of order $\mathcal{O}(10^{-16})$. In all cases, we used unit weights $\omega_j = 1$ for all j in the least-squares problems (6.2), respectively (2.26).

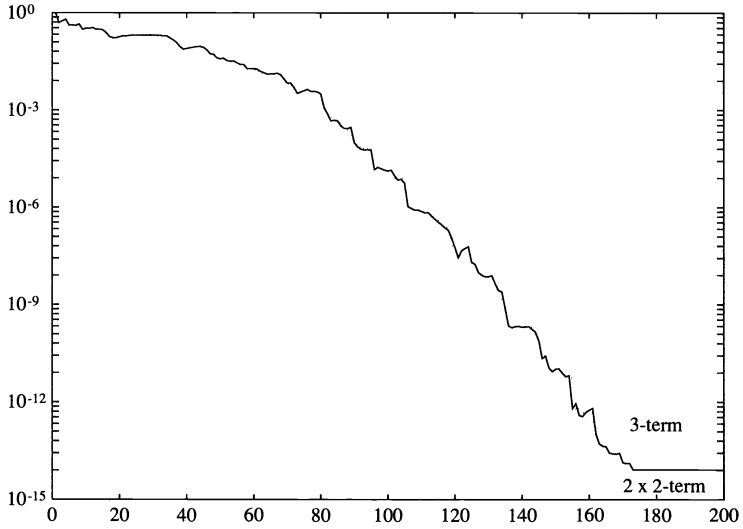


FIG. 9.1. *Convergence curves for Example* 9.1.

Example 9.1. This example is taken from [1] and is meant to illustrate the typical behavior that can be expected from the new implementation of QMR when compared to the original implementation. We consider the partial differential equation

$$Lu = f \quad \text{on } (0, 1) \times (0, 1), \tag{9.1}$$

where

$$Lu := -\frac{\partial}{\partial x}\left(e^{-xy}\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(e^{xy}\frac{\partial u}{\partial y}\right)$$
$$+20(x+y)\frac{\partial u}{\partial x} + 20\frac{\partial}{\partial x}((x+y)u) + \frac{1}{1+x+y}u,$$

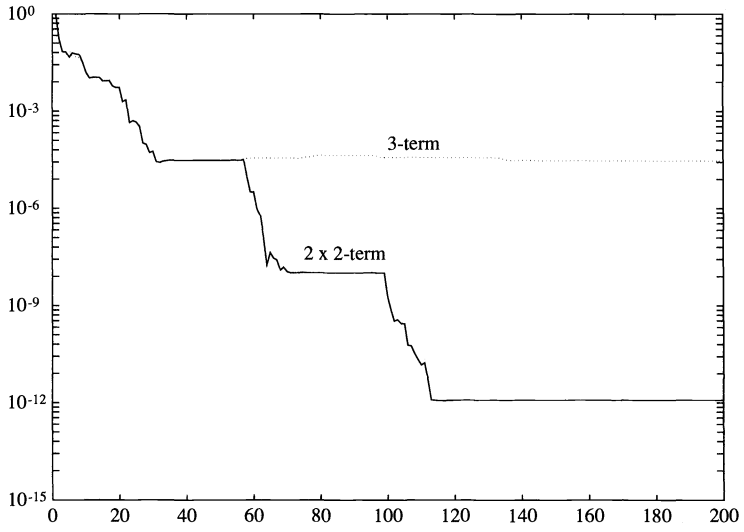


FIG. 9.2. *Convergence curves for Example* 9.2.

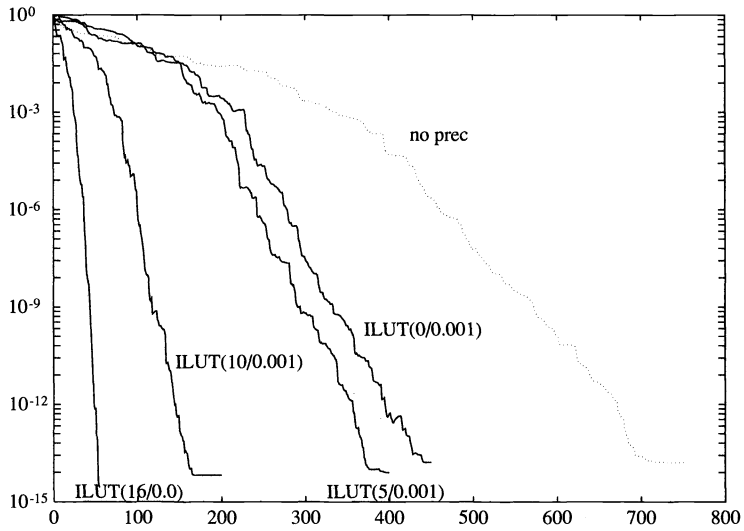


FIG. 9.3. *Convergence curves for Example* 9.3.

with Dirichlet boundary conditions $u = 0$. We discretized (9.1) using centered differences on a 30×30 grid with mesh size $h = \frac{1}{31}$. This leads to a linear system $Ax = b$, where A is a nonsymmetric matrix of order $N = 900$ with 4380 nonzero entries. We ran the original and the new implementations of QMR, both with look-ahead and with the same starting conditions, until the true residual norm $\|r_n\|$ was not reduced any further. The vectors b and w_1 were

random vectors, the initial guess x_0 was zero, and the example was run without preconditioning. The original QMR algorithm is plotted in Fig. 9.1 with a dotted line; it stagnated at `6.7e-14`, and it built six look-ahead blocks of size 2. The coupled QMR algorithm is plotted in Fig. 9.1 with a solid line; it stagnated at `8.3e-15`, and it built four blocks of size 2 in the V–W sequence and seven blocks of size 2 in the P–Q sequence. Recall from §6 that the coupled two-term QMR algorithm solves the least-squares problem (6.2) with coefficient matrix L_n, while the original three-term implementation is based on the least-squares problem (2.26) with coefficient matrix H_n. The Euclidean condition numbers of these matrices at step $n = 165$, respectively $n = 173$, when the original QMR algorithm, respectively the coupled QMR algorithm, begins to stagnate, are $\mathrm{cond}(L_n) =$ `5.2e+03` and $\mathrm{cond}(H_n) =$ `7.3e+03` for both $n = 165$ and $n = 173$. Thus the least-squares problem (6.2) is slightly better conditioned than (2.26).

The behavior in Example 9.1 seems to be fairly typical, in that usually the new implementation is better than the original implementation, but the difference is not very large. However, there are cases where the coupled implementation is significantly better than the original QMR implementation. The next example is of this type.

Example 9.2. This is a linear system that arises in performance modeling of multiprocessor systems, using Petri-net analysis. In such applications, one obtains large sparse singular matrices A with null spaces of dimension 1, and one needs to compute a nontrivial basis vector for this null space. This leads to a linear system of the form $Ax = 0$, and thus condition (2.1) is satisfied. We used a matrix A of size $N = 3663$ with 23397 nonzero elements. The vector b was zero, while the initial guess x_0 and the starting vector w_1 were both random. The linear system is a difficult one, and iterative methods do not converge easily without preconditioning. We used the variant described in [9] of Saad's ILUT preconditioner [22], with no additional fill-in allowed and a drop tolerance of `0.001`, which generated a preconditioning matrix M with 23397 elements. The original QMR algorithm is plotted in Fig. 9.2 with a dotted line; it stagnated at `2.9e-05`, and it built two blocks of size 2. On the other hand, the new implementation, plotted in Fig. 9.2 with a solid line, stagnated at `1.2e-12`, and it built one block of size 2 in the V–W sequence and three blocks of size 2 in the P–Q sequence. The Euclidean condition numbers of the coefficient matrices L_n and H_n of the least-squares problems (6.2) and (2.26) at step $n = 58$, when the original QMR algorithm begins to stagnate, are $\mathrm{cond}(L_{58}) =$ `4.5e+06` and $\mathrm{cond}(H_{58}) =$ `2.4e+10`. At step $n = 113$, when the coupled QMR algorithm begins to stagnate, we have $\mathrm{cond}(L_{113}) =$ `4.9e+06` and $\mathrm{cond}(H_{113}) =$ `3.8e+10`. Thus the least-squares problem (6.2) is considerably better conditioned than (2.26).

Example 9.3. Here A is the complex symmetric YOUNG1C matrix from the Harwell–Boeing test collection of sparse matrices [3]. The matrix arises in a scattering problem in aerodynamics research; it is of dimension $N = 841$ with 4089 nonzero elements. We ran Algorithm 8.1 without look-ahead, with various complex symmetric ILUT preconditioners. In all cases, the iteration was started with the same random vector for b and zero initial guess x_0. This system is also a difficult one, and, if not preconditioned, the QMR algorithm requires around 700 iterations to reach the stagnation level of `2.5e-14`; the corresponding convergence curve is plotted in Fig. 9.3 with a dotted line. However, the ILUT preconditioner is quite effective in this example, especially at higher levels of allowed fill-in and/or drop tolerance. In Fig. 9.3, we show, in order of solid lines from right to left, ILUT with no additional fill-in and `0.001` drop tolerance (2375 nonzero elements), ILUT with five additional fill-in and `0.001` drop tolerance (5171 nonzero elements), ILUT with 10 additional fill-in and `0.001` drop tolerance (9320 nonzero elements), and finally ILUT with 16 additional fill-in and `0.0` drop tolerance (13329 nonzero elements). As can be seen, all variants reach roughly the same stagnation level, around `1.0e-14`. However, as shown in Table 9.1, they do so in

TABLE 9.1
Total execution times (secs) for Example 9.3, average of five runs.

Preconditioner	no prec	ILUT 0/0.001	ILUT 5/0.001	ILUT 10/0.001	ILUT 16/0.0
Iterations	750	450	400	200	55
Time (secs)	144.5	111.4	123.3	75.9	44.4

fewer and fewer iterations, and, in fact, for this example, the additional time spent computing the denser preconditioners was almost always made up by faster convergence.

10. Concluding remarks. We presented a new look-ahead algorithm for constructing Lanczos vectors based on coupled two-term recurrences instead of the usual three-term recurrences. We then discussed a new implementation of the QMR algorithm, using the coupled process to build the basis for the Krylov space. While the theoretical results derived for the original algorithm carry over to the new one, the latter was shown in examples to have better numerical properties. We also briefly covered an implementation of the new QMR method without look-ahead, as well as the application of the QMR algorithm to the solution of complex symmetric linear systems, where the underlying Lanczos process naturally simplifies. Finally, an extended version of this paper, with a more detailed implementation section, is available as a RIACS Technical Report [10].

FORTRAN 77 codes for the proposed coupled-two term look-ahead procedure and the resulting new implementation of the QMR algorithm can be obtained electronically from the authors (freund@research.att.com and na.nachtigal@na-net.ornl.gov). We note that FORTRAN 77 codes for the original implementation of QMR and the underlying look-ahead Lanczos algorithm are available from netlib by sending an email message consisting of the single line "send lalqmr from linalg" to netlib@ornl.gov or netlib@research.att.com.

Acknowledgments. The authors wish to acknowledge the fruitful discussions held with Martin Gutknecht and Tedd Szeto. Marlis Hochbruck and Uwe Seidel provided the matrix for Example 9.2.

REFERENCES

[1] J. K. CULLUM AND R. A. WILLOUGHBY, *A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices*, in Large Scale Eigenvalue Problems, J. K. Cullum and R. A. Willoughby, eds., North-Holland, Amsterdam, 1986, pp. 193–240.

[2] A. DRAUX, *Polynômes Orthogonaux Formels — Applications*, Lecture Notes in Mathematics 974, Springer-Verlag, Berlin, 1983.

[3] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[4] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis Dundee 1975, G. A. Watson, ed., Lecture Notes in Mathematics 506, Springer-Verlag, Berlin, 1976, pp. 73–89.

[5] 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.

[6] ———, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, in Numerical Methods of Approximation Theory, D. Braess and L. L. Schumaker, eds., Birkhäuser, Basel, 1992, pp. 77–95.

[7] 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., 14 (1993), pp. 137–158.

[8] R. W. FREUND AND M. HOCHBRUCK, *On the use of two* QMR *algorithms for solving singular systems and applications in Markov chain modeling*, Tech. Report 91.25, RIACS, NASA Ames Research Center, Moffett Field, CA, Dec. 1991.

[9] 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.

[10] ———, *An implementation of the* QMR *method based on coupled two-term recurrences*, Tech. Report 92.15, RIACS, NASA Ames Research Center, Moffett Field, CA, June 1992.

[11] R. W. FREUND AND T. SZETO, *A quasi-minimal residual squared algorithm for non-Hermitian linear systems*, in Proc. 1992 Copper Mountain Conf. on Iterative Methods, April 1992.

[12] M. H. GUTKNECHT, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part* II, IPS Research Report 90–16, IPS, ETH, Zürich, Switzerland, Sept. 1990.

[13] ———, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part* I, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 594–639.

[14] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[15] W. D. JOUBERT, *Generalized Conjugate Gradient and Lanczos Methods for the Solution of Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, Center for Numerical Analysis, The University of Texas at Austin, Jan. 1990.

[16] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 255–282.

[17] ———, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.

[18] H. P. LANGTANGEN AND A. TVEITO, *A numerical comparison of conjugate gradient-like methods*, Comm. Appl. Numer. Methods, 4 (1988), pp. 793–798.

[19] B. N. PARLETT, D.R. TAYLOR, AND Z. A. LIU, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.

[20] J. K. REID, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in Large Sparse Sets of Linear Equations, J. K. Reid, ed., Academic Press, New York, 1971, pp. 231–253.

[21] H. RUTISHAUSER, *Der Quotienten-Differenzen-Algorithmus*, Mitteilungen aus dem Institut für angewandte Mathematik an der ETH Zürich, Nr. 7, E. Stiefel, ed., Birkhaüser, Basel, 1957.

[22] Y. SAAD, ILUT: *a dual threshold incomplete* LU *factorization*, Research Report UMSI 92/38, University of Minnesota Supercomputer Institute, Minneapolis, March 1992.

[23] D. R. TAYLOR, *Analysis of the Look Ahead Lanczos Algorithm*, Ph.D. thesis, Dept. of Mathematics, University of California, Berkeley, CA, Nov. 1982.

[24] H. WOŹNIAKOWSKI, *Roundoff-error analysis of a new class of conjugate-gradient algorithms*, Linear Algebra Appl., 29 (1980), pp. 507–529.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 338–347, March 1994

006

A QUASI-MINIMAL RESIDUAL VARIANT OF THE BI-CGSTAB ALGORITHM FOR NONSYMMETRIC SYSTEMS*

T. F. CHAN†, E. GALLOPOULOS‡, V. SIMONCINI§, T. SZETO†, AND C. H. TONG¶

Abstract. Motivated by a recent method of Freund [*SIAM J. Sci. Comput.*, 14 (1993), pp. 470–482], who introduced a quasi-minimal residual (QMR) version of the conjugate gradients squared (CGS) algorithm, a QMR variant of the biconjugate gradient stabilized (Bi-CGSTAB) algorithm of van der Vorst that is called QMRCGSTAB, is proposed for solving nonsymmetric linear systems. The motivation for both QMR variants is to obtain smoother convergence behavior of the underlying method. The authors illustrate this by numerical experiments that also show that for problems on which Bi-CGSTAB performs better than CGS, the same advantage carries over to QMRCGSTAB.

Key words. conjugate gradients, Lanczos algorithm, iterative methods, BCG, CGS, QRM, Bi-CGSTAB, nonsymmetric linear systems

AMS subject classifications. 65F10, 65Y20

1. Introduction.

In this note we propose a variation of the Bi-CGSTAB algorithm of van der Vorst [18] for solving the linear system

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

where A is a nonsymmetric sparse matrix of order n.

Various attempts have been made in the last forty years to extend the highly successful conjugate gradient (CG) algorithm to the nonsymmetric case [4]. One such natural extension is what is currently called the biconjugate gradient algorithm (BCG) [9], [1]. Although BCG is still quite competitive today, it also has several well-known drawbacks. Among these are (i) the need for matrix-vector multiplications with A^T (which can be inconvenient as well as doubling the number of matrix-vector multiplications compared to CG for each increase in the degree of the underlying Krylov subspace), (ii) the possibility of breakdowns, and (iii) erratic convergence behavior.

Many recently proposed methods can be viewed as improvements over some of these drawbacks of BCG. The most notable of these is the ingenious CGS method proposed by Sonneveld [14], which cures the first drawback mentioned above by computing the square of the BCG polynomial without requiring A^T. Hence when BCG converges, CGS is an attractive, faster converging alternative. However, this relation between the residual polynomials also causes CGS to behave even more erratically than BCG, particularly in near-breakdown situations for BCG [8], [18]. These observations led van der Vorst [18] to introduce Bi-CGSTAB, a more smoothly converging variant of CGS. The main idea is to form a product of the BCG

*Received by the editors May 28, 1992; accepted for publication (in revised form) March 22, 1993.

†Department of Mathematics, University of California at Los Angeles, California 90024 (chan@math.ucla.edu, szeto@math.ucla.edu). The authors' research was partially supported by Department of Energy grant DE-FG03-87ER25037, Office of Naval Research grant N00014-90-J-1695, National Science Foundation grants ASC90-03002 and ASC92-01266, and Army Research Office grant DAAL03-91-G-150.

‡Department of Computer Science and Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801 (stratis@csrd.uiuc.edu). This author's research was supported by Department of Energy grant DOE DE-FG02-85ER25001 and National Science Foundation grants NSF CCR-9120105 and CCR-9024554. Additional support was provided by the State of Illinois Department of Commerce and Community Affairs, State Technology Challenge Fund grant SCCA 91-108.

§Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, Illinois 61801 (simoncin@csrd.uiuc.edu). This author's research was supported by fellowship 2043597 from the Consiglio Nazionale delle Ricerche, Italy.

¶Center for Computer Engineering, Sandia National Laboratories, Livermore, California 94551 (chtong@snll-arpagw.llnl.gov).

polynomial with another, locally defined polynomial. The Bi-CGSTAB method was further refined by Gutknecht [7] to handle complex matrices and also lead to better convergence for the case of complex eigenvalues. Nevertheless, although the Bi-CGSTAB algorithms were found to perform very well compared to CGS in many situations, there are cases where convergence is still quite erratic (see, for example, §4 and [12]).

In a recent paper [3], Freund proposed a new version of CGS, called TFQMR, which "quasi-minimizes" [6] the residual in the space spanned by the vectors generated by the CGS iteration. Numerical experiments show that in most cases TFQMR retains the good convergence features of CGS while correcting its erratic behavior. The transpose-free nature of TFQMR, its low computational cost, and its smooth convergence behavior make it an attractive alternative to CGS. On the other hand, since the square of the residual polynomial for BCG is still in the space being quasi-minimized, in many practical examples CGS and TFQMR converge in about the same number of steps. We note however that in contrast to CGS, the asymptotic behavior of TFQMR has been analyzed [2]. It is also well known that the CGS residual polynomial can be quite polluted by round-off error [16]. One possible remedy would be to combine TFQMR with a look-ahead Lanczos technique as was done for the original QMR method [5]. In this paper, we take an alternative approach by deriving quasi-minimum residual extensions to Bi-CGSTAB. We call the basic method QMRCGSTAB and illustrate its smoothed convergence by means of numerical experiments.

It may appear redundant to combine the local minimization in Bi-CGSTAB with a global quasi-minimization. However, our view is that the local minimization is secondary in nature and is only used as a way of generating residual polynomials in the appropriate Krylov subspace over which the residual is being quasi-minimized. In fact, this view allows us some flexibility in modifying the local minimization step in Bi-CGSTAB, which leads to other quasi-minimal residual variants. Although we use extensively notation introduced in [18] for algorithm Bi-CGSTAB, for the sake of brevity we refer to that paper for a description of the method.

2. The QMRCGSTAB algorithm. The algorithm proposed in this paper is inspired by TFQMR in that it applies the quasi-minimization principle to the Bi-CGSTAB method, in the same way that TFQMR is derived from CGS. During each step of Bi-CGSTAB, the following vector relations hold:

$$s_i = r_{i-1} - \alpha_i A p_i, \qquad r_i = s_i - \omega_i A s_i, \tag{2}$$

where α_i is the same as the analogous coefficient in BCG, and ω_i is chosen by a local steepest descent principle. Note that x_i is completely determined by α_i and ω_i. Instead, our algorithm uses Bi-CGSTAB to generate the vectors p_i and s_i, but chooses x_i by quasi-minimizing the residual over their span. Let $Y_k = \{y_1, y_2, \ldots, y_k\}$, where $y_{2l-1} = p_l$ for $l = 1, \ldots, [(k+1)/2]$ and $y_{2l} = s_l$ for $l = 1, \ldots, [k/2]$ ($[k/2]$ is the integer part of $k/2$). In the same way, let $W_{k+1} = \{w_0, w_1, \ldots, w_k\}$ with $w_{2l} = r_l$ for $l = 0, \ldots, [k/2]$ and $w_{2l-1} = s_l$ for $l = 1, \ldots, [(k+1)/2]$. We also define $\{\delta_1, \delta_2, \ldots, \delta_k\}$, as $\delta_{2l} = \omega_l$ for $l = 1, \ldots, [(k+1)/2]$ and $\delta_{2l-1} = \alpha_l$ for $l = 1, \ldots, [(k+1)/2]$. In this case, for each column of W_{k+1} and Y_k, (2) may be written as

$$A y_j = (w_{j-1} - w_j)\delta_j^{-1}, \qquad j = 1, \ldots, k, \tag{3}$$

or, using matrix notation,

$$A Y_k = W_{k+1} E_{k+1},$$

where E_{k+1} is a $(k+1) \times k$ bidiagonal matrix with diagonal elements δ_j^{-1} and lower diagonal elements $-\delta_j^{-1}$.

It can easily be checked that the degree of the polynomials corresponding to the vectors r_i, s_i, and p_i are $2i$, $2i-1$, and $2i-2$, respectively. Therefore, $\text{span}(Y_k) = \text{span}(W_k) = K_{k-1}$, where K_k is the Krylov subspace of degree k generated by r_0. The main idea in QMRCGSTAB is to look for an approximation to the solution of (1), using the Krylov subspace K_{k-1}, in the form

$$x_k = x_0 + Y_k g_k \quad \text{with } g_k \in \mathbb{R}^n.$$

Hence, we may write the residual $r_k = b - Ax_k$ as

$$r_k = r_0 - AY_k g_k = r_0 - W_{k+1} E_{k+1} g_k.$$

Using the fact that the first vector of W_{k+1} is indeed r_0, it follows that

$$r_k = W_{k+1}(e_1 - E_{k+1} g_k),$$

where e_1 is the first vector of the canonical basis. Since the columns of W_{k+1} are not normalized, it was suggested in [3] to use a $(k+1) \times (k+1)$ scaling matrix Σ_{k+1}=diag$(\sigma_1, \ldots, \sigma_{k+1})$, with $\sigma_i = \|w_i\|$, to make the columns of W_{k+1} to be of unit norm. Then

$$r_k = W_{k+1}\Sigma_{k+1}^{-1}\Sigma_{k+1}(e_1 - E_{k+1}g_k) = W_{k+1}\Sigma_{k+1}^{-1}(\sigma_1 e_1 - H_{k+1}g_k) \tag{4}$$

with $H_{k+1} = \Sigma_{k+1} E_{k+1}$.

The quasi-minimal residual approach consists of the minimization of $\|\sigma_1 e_1 - H_{k+1} g\|$ for some $g \in \mathbb{R}^k$. In §3 we introduce a variant of QMRCGSTAB that generates W_{k+1} with pairwise orthogonal columns.

The least squares minimization of $\|\sigma_1 e_1 - H_{k+1} g\|$ is solved using QR decomposition of H_{k+1}. This is done in an incremental manner by means of Givens rotations. Since H_{k+1} is lower bidiagonal, only the rotation of the previous step is needed. We refer to [3] for a detailed description of the QR decomposition procedure.

The pseudocode for the QMRCGSTAB algorithm is as follows, in which the Givens rotations used in the QR decomposition are written out explicitly:

ALGORITHM QMRCGSTAB(A, b, x_0, ϵ)
(1) Initialization
$r_0 = b - Ax_0$
choose $\tilde{r}_0$ such that $(\tilde{r}_0, r_0) \neq 0$
$p_0 = v_0 = d_0 = 0$
$\rho_0 = \alpha_0 = \omega_0 = 1$; $\tau = \|r_0\|$, $\theta_0 = 0$, $\eta_0 = 0$
(2) for $k = 1, 2, \ldots$ do
$\rho_k = (\tilde{r}_0, r_{k-1})$; $\beta_k = (\rho_k \alpha_{k-1})/(\rho_{k-1}\omega_{k-1})$
$p_k = r_{k-1} + \beta_k(p_{k-1} - \omega_{k-1} v_{k-1})$
$v_k = Ap_k$
$\alpha_k = \rho_k/(\tilde{r}_0, v_k)$
$s_k = r_{k-1} - \alpha_k v_k$
(2.1) First quasi-minimization and update iterate
$\tilde{\theta}_k = \|s_k\|/\tau$; $c = 1/\sqrt{1+\tilde{\theta}_k^2}$; $\tilde{\tau} = \tau\tilde{\theta}_k c$
$\tilde{\eta}_k = c^2 \alpha_k$
$\tilde{d}_k = p_k + \frac{\theta_{k-1}^2 \eta_{k-1}}{\alpha_k} d_{k-1}$
$\tilde{x}_k = x_{k-1} + \tilde{\eta}_k \tilde{d}_k$

(2.2) compute t_k, ω_k and update r_k

$t_k = As_k$

$\omega_k = (s_k, t_k)/(t_k, t_k)$

$r_k = s_k - \omega_k t_k$

(2.3) Second quasi-minimization and update iterate

$\theta_k = \|r_k\|/\tilde{\tau}$; $c = 1/\sqrt{1+\theta_k^2}$; $\tau = \tilde{\tau}\theta_k c$

$\eta_k = c^2\omega_k$

$d_k = s_k + \frac{\tilde{\theta}_k^2\tilde{\eta}_k}{\omega_k}\tilde{d}_k$

$x_k = \tilde{x}_k + \eta_k d_k$

If x_k is accurate enough, then quit

(3) end

To check the convergence, the estimate $\|\hat{r}_k\| \leq \sqrt{k+1}|\tau|$ was used, where $\hat{r}_k$ denotes the QMRCGSTAB residual at step k [3].

Note that the cost per iteration is slightly higher than for Bi-CGSTAB, since two additional inner products are needed to compute the elements of Σ_{k+1}. A more detailed discussion on computational costs is given in §4.

3. Some variants of QMRCGSTAB. The use of quasi-minimization in the "product algorithms" (such as CGS and Bi-CGSTAB) introduces some flexibility. For example, the underlying product algorithm need not be constrained to generate a residual polynomial that has small norm since, presumably, the quasi-minimization step will handle that. Instead, the basic iteration can be viewed as only generating a set of vectors spanning the Krylov subspace over which the quasi-minimization is applied. This leads us to several variants of QMRCGSTAB, which we will briefly describe. Note however that only one of these variants will be used in the numerical experiments.

We make two observations on the QMRCGSTAB method:

1. It is not crucial that the steepest descent step reduces the norm of the residual as long as it increases the degree of the Krylov subspace associated with W_{k+1}.
2. If W_{k+1} were orthogonal, then quasi-minimization becomes true minimization of the residual.

Therefore, it is natural to choose ω_i to make W_{k+1} "more orthogonal." For example, one can choose ω_i to make r_i orthogonal to s_i and W_k pairwise orthogonal. This leads to the formula:

$$\omega_i = \frac{(s_i, s_i)}{(s_i, t_i)},$$

which replaces the corresponding formula in Algorithm QMRCGSTAB. We call this variant QMRCGSTAB2. We note that since the inner-product (s_i, s_i) is already needed to compute $\tilde{\theta}_i$, we save one inner product compared to QMRCGSTAB.

We also note that similarly to Bi-CGSTAB, both QMRCGSTAB and QMRCGSTAB2 break down if $(s_i, t_i) = 0$, which is possible if A is indefinite (in fact it is always true if A is skew symmetric). This is an additional breakdown condition over that of BCG. One possible strategy to overcome this is to set a lower bound for the quantity $|(s_i, t_i)|$. However, for matrices with large imaginary parts, Gutknecht [7] observed that Bi-CGSTAB does not perform well because the steepest descent polynomials have only real roots and thus cannot be expected to approximate the spectrum well. In principle, it is possible to derive a quasi-minimal residual version of Gutknecht's variant of Bi-CGSTAB, but we shall not pursue that here.

4. Numerical experiments. We next compare the performance of the QMRCGSTAB variants with that of Bi-CGSTAB, TFQMR, and CGS.

Table 1 shows the cost per step of the methods under discussion, excluding the cost for computing the residual norm which is the same for all methods.

TABLE 1
Cost per step for each method.

	Inner products	DAXPY operations	Matrix-vector multiplications
Bi-CGSTAB	4	6	2
CGS	2	7**	2
QMRCGSTAB	6	8	2
QMRCGSTAB2	5	8	2
TFQMR	4	10	2

In the sequel we present experiments to show that QMRCGSTAB indeed achieves a smoothing of the residual compared to Bi-CGSTAB. Note however that, because the Bi-CGSTAB method already improves the erratic residual convergence of BCG, the effect of QMRCGSTAB is not as impressive as the one of TFQMR on the residual of CGS.

Unless stated otherwise, in all examples, the right-hand side b was generated as a random vector with values distributed uniformly in (0, 1), and the starting vector x_0 was taken to be zero. All matrices arising from a partial differential operator were obtained using centered, second-order finite differences. The methods were compared on the basis of the number of iterations necessary to achieve relative residual $\|r_k\|/\|r_0\| < 10^{-8}$ with $r_k = b - Ax_k$ being the true residual. Hence, the figures were built with the abscissae representing the number of iterations and the ordinates representing $\|r_k\|/\|r_0\|$ graded with a logarithmic scale. Experiments were conducted using a Beta test version of Matlab 4.0 [10] running on a Sun Sparc workstation.

Example 1. This example was taken from [14] and corresponds to the discretization of the convection-diffusion operator

$$L(u) = -\varepsilon \triangle u + \cos(\alpha) u_x + \sin(\alpha) u_y \tag{5}$$

on the unit square with homogeneous Dirichlet conditions on the boundary and parameters $\varepsilon = 0.1$ and $\alpha = -30°$, using 40 grid points per direction, yielding a matrix of order $n = 1600$. Figure 1 shows the convergence histories, from which we can see the smoothing effect of quasi-minimization on the CGS and Bi-CGSTAB residuals. We see that Bi-CGSTAB and its smoothed counterparts converge slightly faster than CGS and TFQMR, with QMRCGSTAB2 showing the best performance by a small margin.

Example 2. This example was taken from [17] and corresponds to the discretization of

$$-(Du_x)_x - (Du_y)_y = 1 \tag{6}$$

on the unit square with homogeneous boundary conditions. We used a coarser grid than the one considered in [17]; that is, 50 grid points per direction yielding a matrix of order $n = 2500$. Parameter D takes the value $D = 10^5$ in $0 \le x, y \le 0.75$, $D = 0.1$ in $0.75 < x, y \le 1$, and $D = 1$ everywhere else. Left diagonal preconditioning was applied. In [17], this matrix was used to illustrate the better convergence of Bi-CGSTAB over CGS. We see from Fig. 2 that this advantage carries over to the smoothed versions. Furthermore, even though the matrix is symmetric positive definite and hence CG is applicable, as shown in Fig. 2, the method

**Strictly speaking, one of the operations is a simple vector addition. This must be taken into account if floating point operations were to be counted.

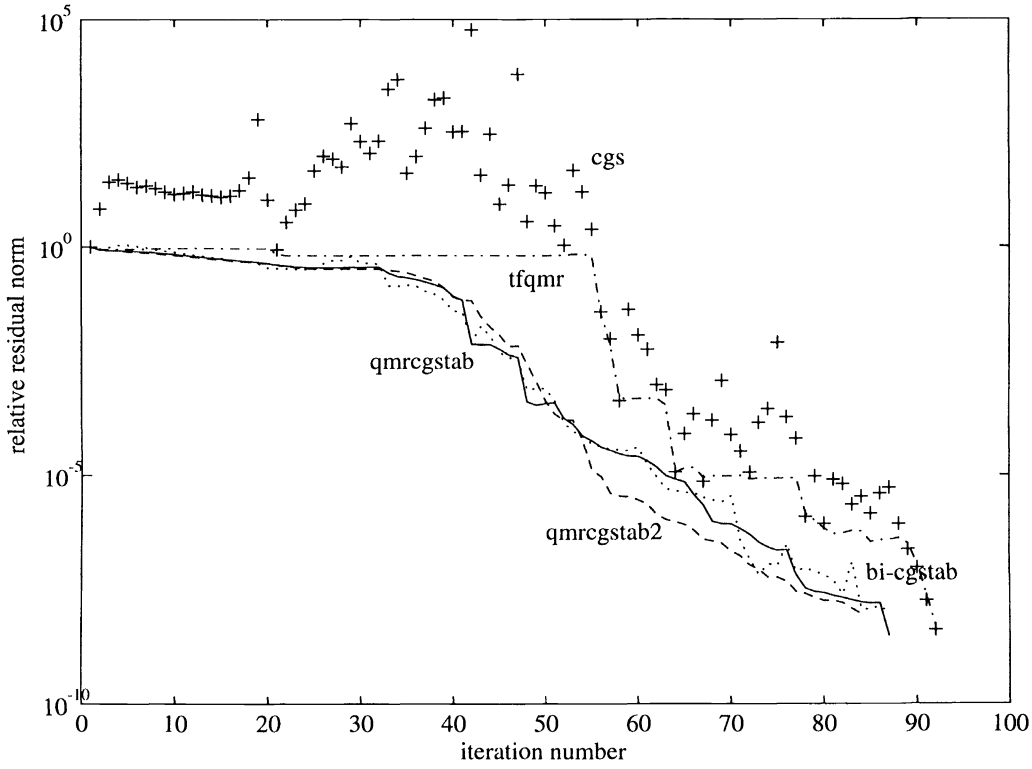


FIG. 1. *Example* 1: *Two-dimensional convection-diffusion operator* (5).

stagnates. This is due to the fact that for this operator, the computed direction vectors of CG methods rapidly lose orthogonality [16]. We note that to make cost comparisons meaningful, the CG curve was plotted so that each "iteration" corresponds to two true CG iterations, i.e., two matrix-vector multiplications.

Example 3. This example comes from the discretization of the convection-diffusion equation

$$L(u) = -\Delta u + \gamma(xu_x + yu_y) + \beta u \tag{7}$$

on the unit square where $\gamma = 100$, $\beta = -100$, for a 63×63 grid, yielding a matrix of order $n = 3969$. No preconditioning was used. In this example, we see the CGS-based methods converge a little faster than Bi-CGSTAB and QMRCGSTAB, but the pairwise orthogonal variant, QMRCGSTAB2, is the fastest. See Fig. 3.

Example 4. Figure 4 shows the results of a three-dimensional version of Example 3 without preconditioning:

$$L(u) = -\Delta u + \gamma(xu_x + yu_y + zu_z) + \beta u \tag{8}$$

on the unit cube where $\beta = -100$, and $\gamma = 50$ for a $15 \times 15 \times 15$ grid, yielding a matrix of order $n = 3375$.

We note that in this example the improvement caused by Bi-CGSTAB over CGS and TFQMR is impressive. Therefore it is not surprising that there is only little additional improvement brought by the variants proposed in this paper. We note that for this operator, the

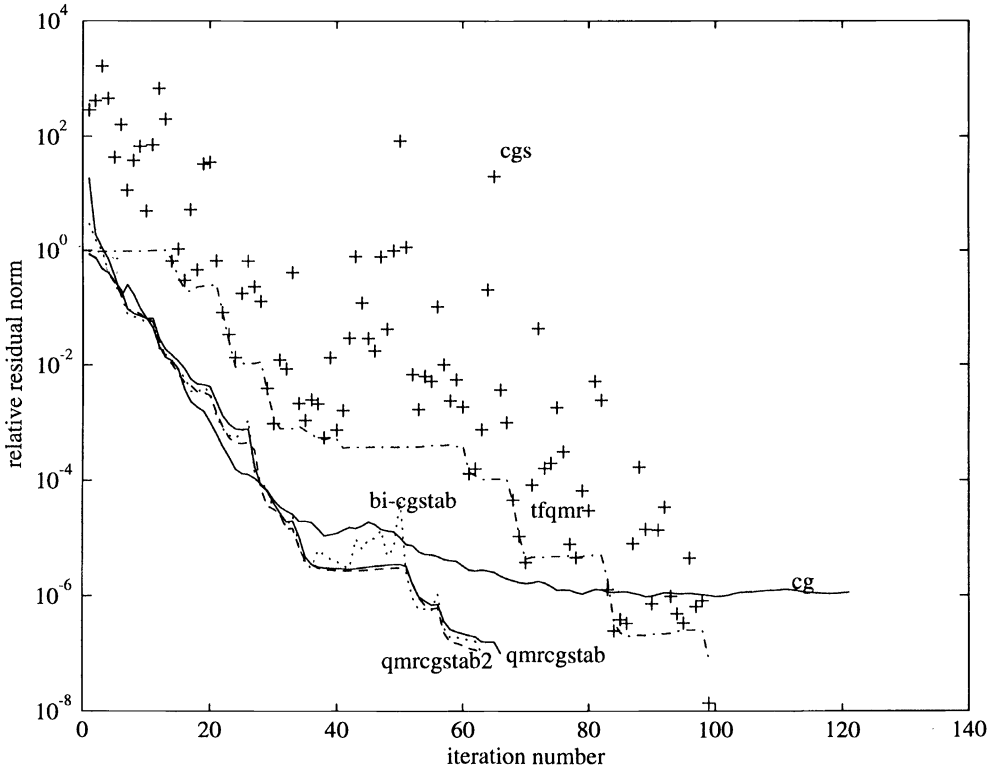


FIG. 2. *Example* 2: *Two-dimensional operator with discontinuous coefficients. Every point on the* CG *curve refers to two CG iterations.*

use of centered differences and large values of γ are unfavorable for Bi-CGSTAB-type methods, since the resulting matrices would have pronounced skew-symmetric components and eigenvalues with large imaginary parts [7]; different discretization methods would be more attractive [13].

Example 5. The next example illustrates how all methods can be affected by the conditioning of the generated polynomial. Matrix A is a modification of an example presented in [11],

$$A = I_{n/2} \otimes \begin{pmatrix} \epsilon & 1 \\ -25 & 100 \end{pmatrix}, \tag{9}$$

i.e., A is an $n \times n$ block diagonal matrix with 2×2 blocks and $n = 40$. We chose $b = (1\ 0\ 1\ 0\ \cdots)^T$ and $\tilde{r}_0 = r_0$. For such a b the norm of the resulting BCG polynomial satisfies $\|\varphi_n\| = \mathcal{O}(\epsilon^{-1})$. Thus, $\|\varphi_n^2\| = \mathcal{O}(\epsilon^{-2})$ in the squared methods and we can foresee numerical problems when ϵ is small.

Each entry of Table 2 shows (i) the number of correct digits, d, in the relative residual obtained after running each algorithm until the relative residual dropped below 10^{-8} but without exceeding 20 matrix vector multiplications, and (ii) in parentheses, the number of matrix vector multiplications, mv, that is a number, not greater than 20, needed to achieve a relative residual of 10^{-d}.

In exact arithmetic, finite termination occurs after the second BCG polynomial φ_2 is computed in both the CGS and Bi-CGSTAB algorithms. We see from Table 2 that all methods

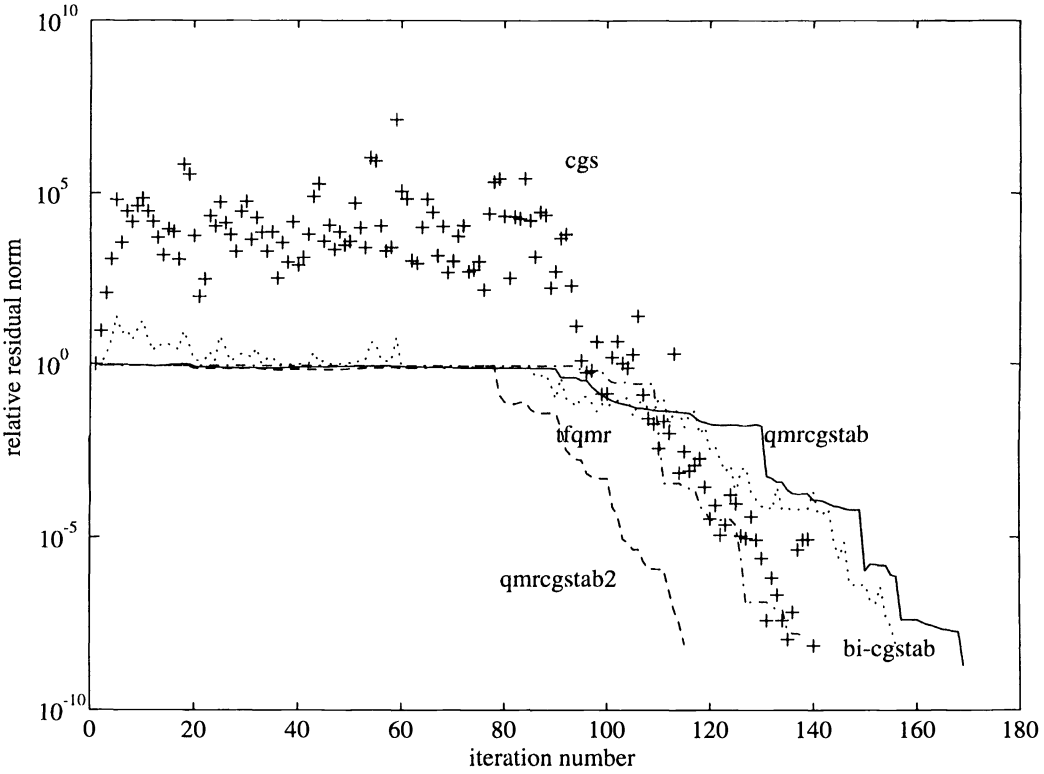


FIG. 3. *Example* 3: *Two-dimensional convection-diffusion operator* (7).

TABLE 2

Example 5: *Correct digits and matrix vector multiplications at termination*: *d(mv). A maximum* 20-*matrix vector multiplications allowed.*

	ϵ			
Method	1.0	10^{-4}	10^{-8}	10^{-12}
CGS	14(4)	5(4)‡	−3(20)*	−1(4)†
TFQMR	13(3)	5(4)‡	1(20)‡	1(20)‡
Bi-CGSTAB	16(3)	12(3)	7(3)‡	3(3)‡
QMRCGSTAB	16(3)	12(3)	7(3)‡	3(3)‡
QMRCGSTAB2	16(3)	12(3)	7(3)‡	3(3)‡

* Oscillatory behavior observed.
‡ Residual stagnates before maximum number of *mv*'s was reached.
† Iterations stopped when division by zero was encountered.

behave equally well for $\epsilon = 1.0$. As ϵ decreases, round-off error causes CGS and TFQMR, which are based on squaring, to fail or not to converge within the expected time. Furthermore, both CGS and TFQMR lose about twice as many digits as Bi-CGSTAB and its quasi-minimal variants. We also mark the instances of the quasi-minimal variants whose residuals stagnate before the maximum number of iterations has been reached. We note that although the example is contrived, it does justify the implementation of a QMRCGSTAB-type method.

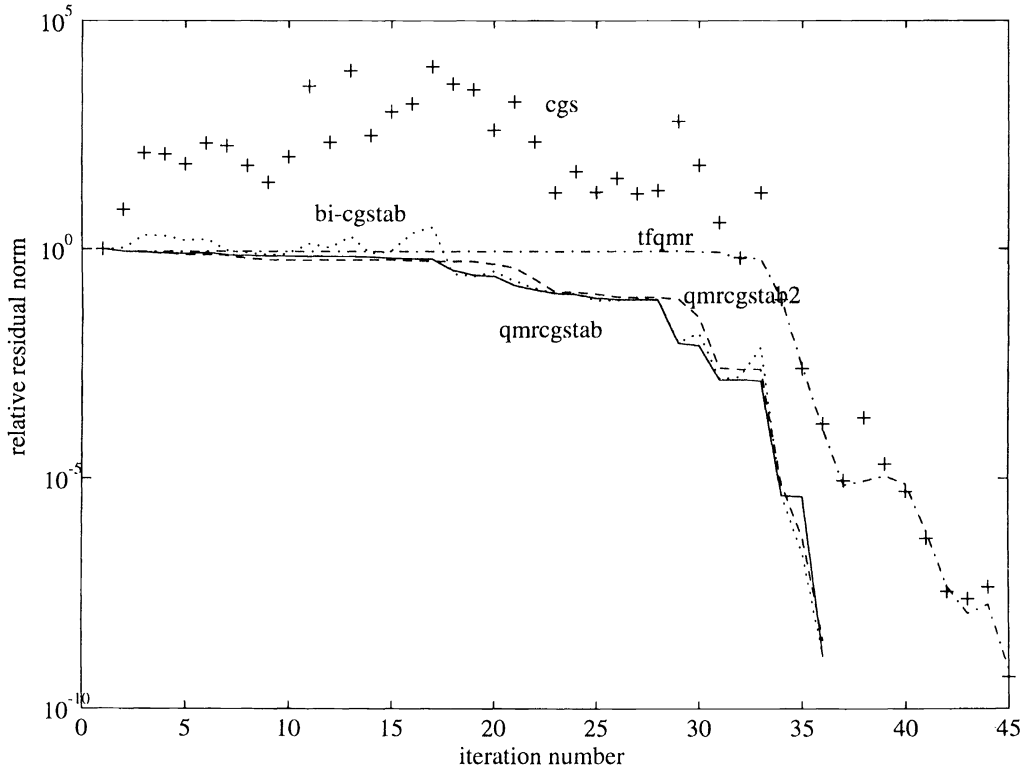


FIG. 4. *Example* 4: *Three-dimensional convection-diffusion operator* (8).

We finally observe that experiments using several of the methods discussed herein, albeit using another naming convention, were presented in [15].

5. Conclusions and future work. We have derived two QMR variants of Bi-CGSTAB. Our motivation for these methods was to inherit any potential improvements on performance that Bi-CGSTAB offers over CGS, while at the same time to provide a smoother convergence behavior. We have shown numerically that this is indeed true for many realistic problems. Although in their present form, the two proposed methods still suffer from some numerical problems, they have many desirable properties: they are transpose-free, they use short recurrences, they make efficient use of matrix-vector multiplications, and they demonstrate smooth convergence behavior.

Acknowledgments. We are grateful to J. M. Hammond, C. Moler, and The Mathworks, Inc. for providing us with the Beta Test version of Matlab 4.0. [10]. We also thank R. Freund and the referees for providing us with very useful suggestions and corrections.

REFERENCES

[1] R. FLETCHER, *Conjugate gradient methods for indefinite linear systems*, in Proc. Dundee Biennial Conf. Numer. Anal., Scotland, G. A. Watson, ed., Lecture Notes in Math., Vol. 506, Springer-Verlag, Berlin, 1976, pp. 73–89.

[2] R. W. FREUND, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, in Numerical Methods in Approximation Theory, Vol. 9, D. Braess and L. L. Schumaker, eds., Birkhäuser Verlag, Basel, 1992, pp. 77–95.

[3] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[4] R. W. FREUND, G. H. GOLUB, AND N. M. NACHTIGAL, *Iterative solution of linear systems*, Acta Numerica, 1 (1992), pp. 57–100.

[5] 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, Research Institute for Advanced Computer Science, Nov. 1990.

[6] 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.

[7] M. H. GUTKNECHT, *Variants of BiCGStab for Matrices with Complex Spectrum*, Tech. Report, Eidgenössische Technische Hochschule, Zürich, Aug. 1991. IPS Research Report 91-14.

[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] C. LANCZOS, *Solution of linear equations by minimized iterations*, J. Res. Natl. Bur. Stand., 49 (1952), pp. 33–53.

[10] THE MATHWORKS, INC., *MATLAB User's Guide*, beta 3 ed., Natick, MA, Feb. 1991.

[11] 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.

[12] C. POMMERELL AND W. FICHTNER, *PILS*: *An iterative linear solver package for ill-conditioned systems*, in Proc. Supercomputing'91, IEEE, Albuquerque, NM, Nov. 1991, pp. 588–599.

[13] A. SEGAL, *Aspects of numerical methods for elliptic singular perturbation problems*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 327–349.

[14] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[15] C. H. TONG, *A comparative study of preconditioned Lanczos methods for nonsymmetric linear systems*, Tech. Report SAND91-8240 UC404, Sandia National Laboratories, Albuquerque, NM, Sept. 1992.

[16] H. A. VAN DER VORST, *The convergence behavior of preconditioned CG and CG-S in the presence of rounding errors*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L. Y. Kolotilina, eds., Lecture Notes in Math., Vol. 1457, Springer-Verlag, Berlin, 1990, pp. 126–136.

[17] ———, *Bi-CGSTAB*: *A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems*, Tech. Report 633, Dept. of Math., University of Utrecht, the Netherlands, Dec. 1990.

[18] ———, *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.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 348–358, March 1994

MAX-MIN PROPERTIES OF MATRIX FACTOR NORMS*

A. GREENBAUM† AND L. GURVITS‡

Abstract. Given a set of real matrices $C_0, C_1, \ldots, C_k$, conditions are considered under which the equality

$$\min_{\alpha_1,\ldots,\alpha_k} \max_{\|w\|=1} \left\| \left(C_0 + \sum_{i=1}^{k} \alpha_i C_i \right) w \right\| = \max_{\|w\|=1} \min_{\alpha_1,\ldots,\alpha_k} \left\| \left(C_0 + \sum_{i=1}^{k} \alpha_i C_i \right) w \right\|$$

holds. It is shown that if the matrices C_i, $i = 0, 1, \ldots, k$ are normal and commute with one another, then the equality holds. In particular, this implies that if $C_i = A^i$ or $C_i = A^{k-i}$, where A is a normal matrix, then the equality holds. An example is given to show that the equality may fail for noncommuting matrices, when $k > 1$. It is shown that the equality holds for arbitrary matrices if $k = 1$.

Key words. GMRES, Arnoldi, matrix approximation problem, normal matrix, minmax

AMS subject classifications. 65F10, 49K35

1. Introduction. The following problem arises in the analysis of iterative methods for solving linear systems and computing eigenvalues. To solve a linear system, $Ax = b$, given an initial guess x^0 for the solution, the generalized minimal residual (GMRES) method [7] generates approximate solutions x^k, $k = 1, 2, \ldots$ of the form

$$x^k = x^0 + \sum_{i=1}^{k} \alpha_{ik} A^{i-1} r^0,$$

where $r^0 \equiv b - Ax^0$ is the initial residual. The residual vectors $r^k \equiv b - Ax^k$ are of the form

$$r^k = r^0 - \sum_{i=1}^{k} \alpha_{ik} A^i r^0,$$

and the coefficients $\alpha_{1k}, \ldots, \alpha_{kk}$ are chosen to make the 2-norm of r^k as small as possible. A bound on the 2-norm of the residual at any step k is given by

$$\|r^k\| \le \min_{\alpha_1,\ldots,\alpha_k} \left\| I - \sum_{i=1}^{k} \alpha_i A^i \right\| \cdot \|r^0\|.$$

The question arises as to whether this bound is ever attained; that is, whether there is an initial residual r^0, such that

$$\min_{\alpha_1,\ldots,\alpha_k} \left\| \left(I - \sum_{i=1}^{k} \alpha_i A^i \right) r^0 \right\| = \min_{\alpha_1,\ldots,\alpha_k} \left\| I - \sum_{i=1}^{k} \alpha_i A^i \right\| \cdot \|r^0\|.$$

In other words, we have the following max-min problem: Is the inequality

$$\max_{\|r^0\|=1} \min_{\alpha_1,\ldots,\alpha_k} \left\| \left(I - \sum_{i=1}^{k} \alpha_i A^i \right) r^0 \right\| \le \min_{\alpha_1,\ldots,\alpha_k} \max_{\|r^0\|=1} \left\| \left(I - \sum_{i=1}^{k} \alpha_i A^i \right) r^0 \right\| \tag{1}$$

actually an equality?

*Received by the editors June 17, 1992; accepted for publication (in revised form) June 9, 1993.

†Courant Institute of Mathematical Sciences, New York University, New York, New York (greenbau@cs.nyu.edu). This author's work was supported in part by the Applied Mathematical Sciences Program of the U.S. Department of Energy under contract DEFG0288ER25053, and was conducted while visiting the Institute for Mathematics and its Applications at the University of Minnesota, Minneapolis, Minnesota.

‡Siemens Research Corporation, 755 College Road East, Princeton, New Jersey 08540. This author's research was performed while visiting the Institute for Mathematics and its Applications at the University of Minnesota, Minneapolis, Minnesota (gurvits@learning.siemens.com).

A similar question arises in analyzing the Arnoldi method [1] for computing eigenvalues. Given an initial vector q with $\|q\| = 1$, the Arnoldi iteration constructs a sequence of monic polynomials P_k, $k = 1, 2, \ldots$ whose coefficients are chosen to minimize $\|p_k(A)q\|$ over all monic polynomials p_k of degree k. The roots of these polynomials are taken as approximate eigenvalues of the matrix A. The question arises as to whether, for each k, there is an initial vector q such that the monic polynomial P_k constructed by the Arnoldi process also minimizes $\|p_k(A)\|$. A similar max-min statement of the problem asks if the inequality

$$\max_{\|q\|=1} \min_{\alpha_1,\ldots,\alpha_k} \left\| \left(A^k - \sum_{i=1}^{k} \alpha_i A^{k-i} \right) q \right\| \leq \min_{\alpha_1,\ldots,\alpha_k} \max_{\|q\|=1} \left\| \left(A^k - \sum_{i=1}^{k} \alpha_i A^{k-i} \right) q \right\| \tag{2}$$

is actually an equality.

In this paper, we consider a somewhat more general question: Given an arbitrary sequence of real matrices $C_0, C_1, \ldots, C_k$, under what circumstances will the equality

$$\min_{\alpha_1,\ldots,\alpha_k} \max_{\|w\|=1} \left\| \left(C_0 + \sum_{i=1}^{k} \alpha_i C_i \right) w \right\| = \max_{\|w\|=1} \min_{\alpha_1,\ldots,\alpha_k} \left\| \left(C_0 + \sum_{i=1}^{k} \alpha_i C_i \right) w \right\| \tag{3}$$

hold? It is shown that if the matrices C_i, $i = 0, 1, \ldots, k$ are normal and commute with one another, then (3) holds. This result has actually been known for some time, in a slightly different form [6, p. 292]. We include a proof of the result here, since it is not widely known in the numerical analysis community and we point out how our theorem is equivalent to an established result in approximation theory. In particular, this implies that if $C_i = A^i$ or $C_i = A^{k-i}$, where A is a normal matrix, then the equality holds. This generalizes some known results showing that equality holds in (1) and (2) when the matrix A is normal [2], [4], [5]. An example is given to show that (3) may fail for noncommuting matrices, when $k > 1$. It is shown that the equality (3) holds for arbitrary matrices if $k = 1$. The question of whether equality holds in (1) and (2) when $k > 1$ remains open, as does the question of more general conditions on the matrices $C_0, C_1, \ldots, C_k$ that would ensure that (3) holds.

Throughout this paper, we assume that the matrices and vectors appearing in our max-min statements and related theorems are real (though, of course, the eigenvalues and eigenvectors of these matrices may be complex). We will use the notation $A > 0$ to mean that the symmetric matrix A is positive definite. For a vector w, $\|w\|$ will always denote the 2-norm, and for a matrix A, $\|A\|$ will denote the corresponding matrix norm, $\max_{\|w\|=1} \|Aw\|$.

The next section gives the main theorems and examples.

2. Main theorems. The first theorem gives conditions under which some linear combination of given symmetric matrices is positive definite.

THEOREM 2.1. *Let $A_1, \ldots, A_k$ be real n-by-n symmetric matrices. There exist scalars $\alpha_1, \ldots, \alpha_k$ such that*

$$\sum_{i=1}^{k} \alpha_i A_i > 0 \tag{4}$$

if and only if for every set $\{w_1, \ldots, w_m\}$, $m \leq n$, of real nonzero orthogonal n-vectors ($w_p \cdot w_j = 0$, for $p \neq j$), there is an i such that

$$\sum_{j=1}^{m} \langle A_i w_j, w_j \rangle \neq 0. \tag{5}$$

If the symmetric matrices $A_1, \ldots, A_k$ commute: $A_i A_j = A_j A_i$, $i, j = 1, \ldots, k$, then there exist scalars $\alpha_1, \ldots, \alpha_k$ for which (4) *holds if and only if for every individual real nonzero n-vector w, there is an i such that*

$$\langle A_i w, w \rangle \neq 0. \tag{6}$$

Proof. To see the necessity of (5), note that if, for some orthogonal set $\{w_1, \ldots, w_m\}$, we have

$$\sum_{j=1}^{m} \langle A_i w_j, w_j \rangle = 0$$

for all i, then also

$$\sum_{j=1}^{m} \left\langle \sum_{i=1}^{k} \alpha_i A_i w_j, w_j \right\rangle = \operatorname{tr}\left(W^T \left(\sum_{i=1}^{k} \alpha_i A_i \right) W \right) = 0$$

for any $\alpha_1, \ldots, \alpha_k$, where W is the matrix whose columns are $w_1, \ldots, w_k$. But since every principal submatrix of a positive definite matrix is positive definite, this implies that $\sum_{i=1}^{k} \alpha_i A_i$ cannot be positive definite.

To see the sufficiency of (5), suppose that no linear combination $\sum_{i=1}^{k} \alpha_i A_i$ is positive definite. Then the linear subspace spanned by $A_1, \ldots, A_k$ and the convex cone of positive definite matrices can be separated. That is, there is a symmetric matrix B such that

$$\operatorname{tr}\left(B \sum_{i=1}^{k} \alpha_i A_i \right) = 0 \tag{7}$$

for all $\alpha_1, \ldots, \alpha_k$ and

$$\operatorname{tr}(BP) > 0 \tag{8}$$

for all positive definite matrices P. Write B in the form $B = QDQ^T$, where D is the diagonal matrix of eigenvalues of B and Q is the orthogonal matrix of eigenvectors. Then (8) implies that

$$\operatorname{tr}(QDQ^T P) = \operatorname{tr}(D\, Q^T P Q) > 0. \tag{9}$$

Since the diagonal elements of $Q^T P Q$ can be any positive numbers, (9) implies that the diagonal elements of D are nonnegative, with at least one of these being positive. But from (7) it follows that

$$\operatorname{tr}(BA_i) = \operatorname{tr}(QDQ^T A_i) = \operatorname{tr}((QD^{1/2})^T A_i (QD^{1/2})) = 0$$

for all i. Taking $w_1, \ldots, w_m$ to be the nonzero columns of $QD^{1/2}$, this says

$$\sum_{j=1}^{m} w_j^T A_i w_j = 0,$$

which contradicts (5).

The second part of the theorem can be proved similarly, using the fact that if the matrices $A_1, \ldots, A_k$ commute, then they can be simultaneously diagonalized. That is, there exists an orthogonal matrix Q such that

$$A_i = Q\Lambda_i Q^T, \quad QQ^T = Q^T Q = I, \quad \Lambda_i = \operatorname{diag}(\lambda_{i1}, \ldots, \lambda_{in}).$$

Again, the necessity of (6) is clear, and to see that it is sufficient, we note that if no linear combination of $A_1, \ldots, A_k$ and hence of $\Lambda_1, \ldots, \Lambda_k$ is positive definite, then there is a diagonal matrix $D = \operatorname{diag}(d_1, \ldots, d_n)$ such that

$$\operatorname{tr}\left(D \sum_{i=1}^{k} \alpha_i \Lambda_i\right) = 0, \tag{10}$$

for all $\alpha_1, \ldots, \alpha_k$ and

$$\operatorname{tr}(DP) > 0 \tag{11}$$

for all positive definite diagonal matrices P. From (11) it follows that the diagonal elements of D are nonnegative, with at least one being positive. Define w to be the vector $(\sqrt{d_1}, \ldots, \sqrt{d_n})^T$. From (10) we have for each i,

$$\operatorname{tr}(D\Lambda_i) = \sum_{j=1}^{n} d_j \lambda_{ij} = \langle \Lambda_i w, w \rangle = 0,$$

which contradicts (6). □

The second part of Theorem 2.1 is really a result about vectors rather than matrices: Given a set of n-vectors, $\phi^{(1)}, \ldots, \phi^{(k)}$ (corresponding to the diagonal matrices $\Lambda_1, \ldots, \Lambda_k$ in the theorem), there is a linear combination of these vectors that has all positive elements if and only if for every set of nonnegative numbers $w_1^2, \ldots, w_n^2$, not all zero, there is an i such that

$$\sum_{j=1}^{n} \phi_j^{(i)} w_j^2 \neq 0.$$

In this form, the second part of Theorem 2.1, as well as Theorem 2.3, which is given later, are known. See, for example, [6]. We include proofs of these theorems here for completeness.

We now use Theorem 2.1 to establish conditions under which the optimal coefficients $\alpha_1, \ldots, \alpha_k$ on both sides of equality (3) are zero.

THEOREM 2.2. *Let $C_1, \ldots, C_k$ be real square matrices such that each pair $(C_i + C_i^T)$ and $(C_j + C_j^T)$ commute. Suppose*

$$\min_{\alpha_1, \ldots, \alpha_k} \max_{\|w\|=1} \left\| \left(I + \sum_{i=1}^{k} \alpha_i C_i\right) w \right\| = 1. \tag{12}$$

Then

$$\max_{\|w\|=1} \min_{\alpha_1, \ldots, \alpha_k} \left\| \left(I + \sum_{i=1}^{k} \alpha_i C_i\right) w \right\| = 1. \tag{13}$$

Proof. For a given vector w, we have

$$\min_{\alpha_1,\ldots,\alpha_k} \left\| \left(I + \sum_{i=1}^{k} \alpha_i C_i \right) w \right\| = 1$$

if and only if $\langle C_i w, w\rangle = 0$ for all i; i.e., if and only if $\langle (C_i + C_i^T)w, w\rangle = 0$ for all i. Suppose (13) does not hold. Then for any vector $w \neq 0$ there is an i such that

$$\langle (C_i + C_i^T)w, w\rangle \neq 0.$$

From Theorem 2.1, there is a linear combination

$$\sum_{i=1}^{k} \alpha_i (C_i + C_i^T)$$

that is positive definite. For ϵ sufficiently small, then,

$$\left\| I - \epsilon \sum_{i=1}^{k} \alpha_i C_i \right\|^2 = \left\| I - \epsilon \sum_{i=1}^{k} \alpha_i (C_i + C_i^T) \right\| + \mathcal{O}(\epsilon^2) < 1,$$

which contradicts the assumption (12). □

Theorem 2.2 is now used to establish certain conditions under which equality (3) holds.

THEOREM 2.3. *Let $C_0, C_1, \ldots, C_k$ be nonsingular normal matrices that commute. Then*

$$\min_{\alpha_1,\ldots,\alpha_k} \max_{\|w\|=1} \left\| \left(C_0 + \sum_{i=1}^{k} \alpha_i C_i \right) w \right\| = \max_{\|w\|=1} \min_{\alpha_1,\ldots,\alpha_k} \left\| \left(C_0 + \sum_{i=1}^{k} \alpha_i C_i \right) w \right\|. \tag{14}$$

Proof. Suppose $\hat{\alpha}_1, \ldots, \hat{\alpha}_k$ minimize $\|C_0 + \sum_{i=1}^{k} \alpha_i C_i\|$. We can assume without loss of generality that this minimal norm is 1. We will consider two cases.

1. First, suppose all singular values of $U \equiv C_0 + \sum_{i=1}^{k} \hat{\alpha}_i C_i$ are equal. Then U is a real orthogonal matrix and it commutes with each matrix C_i. The same holds for the inverse matrix U^T. We can write

$$\min_{\alpha_1,\ldots,\alpha_k} \left\| C_0 + \sum_{i=1}^{k} \alpha_i C_i \right\| = \min_{\beta_1,\ldots,\beta_k} \left\| U^T \left(C_0 + \sum_{i=1}^{k} (_i a \hat{\alpha}_i + \beta_i) C_i \right) \right\|$$

$$= \min_{\beta_1,\ldots,\beta_k} \left\| I + \sum_{i=1}^{k} \beta_i U^T C_i \right\| = 1.$$

Because the matrices C_i are normal and commute with each other and with U and U^T, each pair $(U^T C_i + C_i^T U)$ and $(U^T C_j + C_j^T U)$ commute. Therefore, from Theorem 2.2, we have

$$\max_{\|w\|=1} \min_{\beta_1,\ldots,\beta_k} \left\| \left(I + \sum_{i=1}^{k} \beta_i U^T C_i \right) w \right\| = 1,$$

and from this the desired result follows.

2. Now suppose some singular values are less than 1. We can write the real Schur decomposition of each matrix C_i in the form

$$C_i = Q D_i Q^T, \qquad Q Q^T = Q^T Q = I,$$

where each D_i is a block diagonal matrix with 1-by-1 or 2-by-2 blocks on the main diagonal. It suffices to consider the block diagonal matrices D_i. Define $\hat{D} \equiv D_0 + \sum_{i=1}^{k} \hat{\alpha}_i D_i$, and order the elements so that $\hat{D}$ is of the form

$$\hat{D} = \begin{pmatrix} U & 0 \\ 0 & K \end{pmatrix},$$

where U is, say, a t-by-t matrix whose eigenvalues are all equal to 1 in magnitude and K is an $(n-t)$-by-$(n-t)$ matrix whose eigenvalues are all less than 1 in magnitude. Note that each matrix D_i has this same block structure

$$D_i = \begin{pmatrix} D_{i1} & 0 \\ 0 & D_{i2} \end{pmatrix},$$

since any off-diagonal block X in D_i would have to satisfy the homogeneous Sylvester equation: $UX - XK = 0$ in order that D_i and $\hat{D}$ commute. Since the spectrum of U does not intersect the spectrum of K, this equation has only the trivial solution $X = 0$. We can write

$$\begin{aligned} \min_{\alpha_1,\ldots,\alpha_k} \left\| D_0 + \sum_{i=1}^{k} \alpha_i D_i \right\| &= \min_{\beta_1,\ldots,\beta_k} \left\| D_0 + \sum_{i=1}^{k} (\hat{\alpha}_i + \beta_i) D_i \right\| \\ &= \min_{\beta_1,\ldots,\beta_k} \left\| \begin{pmatrix} U & 0 \\ 0 & K \end{pmatrix} + \sum_{i=1}^{k} \beta_i \begin{pmatrix} D_{i1} & 0 \\ 0 & D_{i2} \end{pmatrix} \right\|. \end{aligned} \tag{15}$$

Since $\|U\| > \|K\|$, the same coefficients $\beta_1, \ldots, \beta_k$ that minimize the norm of the matrix in (15) (namely, $\beta_i = 0$, $i = 1, \ldots, k$) also minimize the norm of the upper left t-by-t block, and we have

$$\min_{\beta_1,\ldots,\beta_k} \left\| U - \sum_{i=1}^{k} \beta_i D_{i1} \right\| = 1.$$

Since the singular values of the minimal norm matrix of this form are all equal to 1 and since the matrices D_{i1} are normal and commute with each other, it now follows from part 1 that there is a t-vector $\hat{w}$ with $\|\hat{w}\| = 1$ such that

$$\min_{\beta_1,\ldots,\beta_k} \left\| U - \sum_{i=1}^{k} \beta_i D_{i1} \right\| = \min_{\beta_1,\ldots,\beta_k} \left\| \left(U - \sum_{i=1}^{k} \beta_i D_{i1} \right) \hat{w} \right\|.$$

Defining $\tilde{w}$ to be the n-vector whose first t elements are equal to those of $\hat{w}$ and whose remaining elements are zero, we find

$$\min_{\alpha_1,\ldots,\alpha_k} \left\| D_0 + \sum_{i=1}^{k} \alpha_i D_i \right\| = \min_{\alpha_1,\ldots,\alpha_k} \left\| \left(D_0 + \sum_{i=1}^{k} \alpha_i D_i \right) \tilde{w} \right\|,$$

from which the desired result follows. □

Stated in terms of vectors, Theorem 2.3 states the following. Let f denote the vector consisting of the eigenvalues of C_0, and let $g^{(1)}, \ldots, g^{(k)}$ denote the vectors consisting of the eigenvalues of $C_1, \ldots, C_k$. The left-hand side of (14) is the distance between f and the closest vector to f in the infinity norm from the space G spanned by $g^{(1)}, \ldots, g^{(k)}$. The right-hand side of (14) is the difference between f and the closest vector to f in some weighted L_2 norm

from the space G. The weights are the squared components of the vector w in the direction of each eigenvector of C_i. With this notation, Theorem 2.3 can be stated as follows.

There exists a vector w in R^n with $\|w\|_2 = 1$ such that

$$\min_{g \in G} \|f - g\|_\infty = \|f - \bar{g}\|_\infty = \|f - \bar{g}\|_w = \min_{g \in G} \|f - g\|_w,$$

where $\|v\|_w^2 \equiv \sum_{i=1}^n v_i^2 w_i^2$.

A more general version of this theorem can be deduced from [6, p. 292]. Corollary 1 in that reference states that for any p, q with $1 < p < q \leq \infty$, the vector $\bar{g}$ minimizes some weighted L_q norm of $f - g$ if and only if it minimizes some weighted L_p norm of $f - g$. To obtain the additional result that $\|f - \bar{g}\|_\infty = \|f - \bar{g}\|_w$, we must use the fact that $|f_i - \bar{g}_i|$ attains its maximum value at each of the points i for which w_i is nonzero [6, p. 302]. These facts form the basis of Lawson's algorithm for computing L_∞ approximations by solving a sequence of weighted L_2 (or L_k) approximation problems [3].

Note that the assumption of commutativity in the second part of Theorem 2.1, and hence in Theorems 2.2 and 2.3, is necessary. Consider, for example, the symmetric matrices

$$A_1 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{16}$$

For any vector w, we have

$$\langle A_1 w, w \rangle = w_1^2 - w_2^2, \qquad \langle A_2 w, w \rangle = w_1^2 + 2w_1 w_2 - w_2^2,$$

and if the first inner product is zero and w_1 and w_2 are not both zero, then the second inner product cannot be zero. Yet there is no linear combination of A_1 and A_2 that is positive definite. We have

$$\min_{\alpha_1, \alpha_2} \|I + \alpha_1 A_1 + \alpha_2 A_2\| = 1,$$

but for any vector w,

$$\min_{\alpha_1, \alpha_2} \|(I + \alpha_1 A_1 + \alpha_2 A_2) w\| = 0.$$

In the next lemma we consider general real m-by-n matrices $C_0, C_1, \ldots, C_k$. Suppose $\hat{\alpha}_1, \ldots, \hat{\alpha}_k$ minimize

$$\left\| C_0 + \sum_{i=1}^k \alpha_i C_i \right\|.$$

Define $\hat{C} \equiv C_0 + \sum_{i=1}^k \hat{\alpha}_i C_i$ and write the singular value decomposition of $\hat{C}$ as

$$\hat{C} = U \Sigma V^T,$$

where U is an m-by-m real orthogonal matrix, V is an n-by-n real orthogonal matrix, and Σ is an m-by-n matrix of the form

$$\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \cdot & & \\ & & \cdot & \\ & & & \sigma_n \\ 0 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & 0 \end{pmatrix} \quad \text{or} \quad \Sigma = \begin{pmatrix} \sigma_1 & & & 0 & \cdot & 0 \\ & \cdot & & \cdot & \cdot & \cdot \\ & & \cdot & \cdot & \cdot & \cdot \\ & & & \sigma_m & 0 & \cdot & 0 \end{pmatrix}$$

accordingly as $m \geq n$ or $m \leq n$. Assume that the singular values σ_i, $i = 1, \ldots, \min\{m, n\}$ satisfy

$$\sigma_1 = \cdots = \sigma_t > \sigma_{t+1} \geq \cdots,$$

and let V_t be the n-by-t matrix consisting of the first t columns of V, while $V_{t+1:n}$ is the n-by-$(n-t)$ matrix consisting of columns $t+1$ through n of V.

The following lemma is used to prove Theorem 2.5, but it is also of significant interest in itself.

LEMMA 2.4. *Using the above notation, the coefficients* $\hat{\alpha}_1, \ldots, \hat{\alpha}_k$ *that minimize*

$$\left\| C_0 + \sum_{i=1}^{k} \alpha_i C_i \right\| \tag{17}$$

also minimize

$$\left\| \left(C_0 + \sum_{i=1}^{k} \alpha_i C_i \right) V_t \right\|. \tag{18}$$

Proof. Suppose $\hat{\alpha}_1, \ldots, \hat{\alpha}_k$ do not minimize (18). Then there are coefficients $\tilde{\alpha}_1, \ldots, \tilde{\alpha}_k$ such that

$$\left\| \left(C_0 + \sum_{i=1}^{k} \tilde{\alpha}_i C_i \right) V_t \right\| < \left\| \left(C_0 + \sum_{i=1}^{k} \hat{\alpha}_i C_i \right) V_t \right\|.$$

We will show that for sufficiently small values of ϵ, the coefficients $(1-\epsilon)\hat{\alpha}_i + \epsilon\tilde{\alpha}_i$ satisfy

$$\left\| C_0 + \sum_{i=1}^{k} ((1-\epsilon)\hat{\alpha}_i + \epsilon\tilde{\alpha}_i) C_i \right\| < \left\| C_0 + \sum_{i=1}^{k} \hat{\alpha}_i C_i \right\|,$$

which contradicts the assumption that $\hat{\alpha}_1, \ldots, \hat{\alpha}_k$ minimize (17).

For any ϵ in $(0, 1)$ we have

$$\begin{aligned} \left\| \left(C_0 + \sum_{i=1}^{k} ((1-\epsilon)\hat{\alpha}_i + \epsilon\tilde{\alpha}_i) C_i \right) V_t \right\| &\leq (1-\epsilon) \left\| \left(C_0 + \sum_{i=1}^{k} \hat{\alpha}_i C_i \right) V_t \right\| \\ &\quad + \epsilon \left\| \left(C_0 + \sum_{i=1}^{k} \tilde{\alpha}_i C_i \right) V_t \right\| < \sigma_1 - \mathcal{O}(\epsilon). \end{aligned} \tag{19}$$

For sufficiently small ϵ, we also have

$$\begin{aligned} \left\| \left(C_0 + \sum_{i=1}^{k} ((1-\epsilon)\hat{\alpha}_i + \epsilon\tilde{\alpha}_i) C_i \right) V_{t+1:n} \right\| &\leq (1-\epsilon) \left\| \left(C_0 + \sum_{i=1}^{k} \hat{\alpha}_i C_i \right) V_{t+1:n} \right\| \\ &\quad + \epsilon \left\| \left(C_0 + \sum_{i=1}^{k} \tilde{\alpha}_i C_i \right) V_{t+1:n} \right\| < \sigma_{t+1} + \frac{1}{2}(\sigma_1 - \sigma_{t+1}). \end{aligned} \tag{20}$$

Define the matrix $K \equiv (K_1, K_2)$ by

$$K_1 = U^T \left(C_0 + \sum_{i=1}^{k} ((1-\epsilon)\hat{\alpha}_i + \epsilon\tilde{\alpha}_i) C_i \right) V_t$$

$$= (1-\epsilon)\sigma_1 \begin{pmatrix} I \\ 0 \end{pmatrix} + \epsilon U^T \left(C_0 + \sum_{i=1}^{k} \tilde{\alpha}_i C_i \right) V_t,$$

$$K_2 = U^T \left(C_0 + \sum_{i=1}^{k} ((1-\epsilon)\hat{\alpha}_i + \epsilon\tilde{\alpha}_i) C_i \right) V_{t+1:n}$$

$$= (1-\epsilon) \begin{pmatrix} 0 \\ \Sigma_{t+1:n} \end{pmatrix} + \epsilon U^T \left(C_0 + \sum_{i=1}^{k} \tilde{\alpha}_i C_i \right) V_{t+1:n},$$

where $\Sigma_{t+1:n} \equiv \operatorname{diag}(\sigma_{t+1}, \ldots, \sigma_n)$. We would like to show that $\|K\| < \sigma_1$ or, equivalently, that the matrix

$$\sigma_1^2 I - K^T K = \begin{pmatrix} \sigma_1^2 I - K_1^T K_1 & -K_1^T K_2 \\ -K_2^T K_1 & \sigma_1^2 I - K_2^T K_2 \end{pmatrix} \tag{21}$$

is positive definite. From (19) and (20) it follows that the diagonal blocks are positive definite, so it suffices to show that

$$(\sigma_1^2 I - K_1^T K_1) - K_1^T K_2 (\sigma_1^2 I - K_2^T K_2)^{-1} K_2^T K_1 > 0.$$

It is easy to check that

$$\|K_1^T K_2 (\sigma_1^2 I - K_2^T K_2)^{-1} K_2^T K_1\| = \mathcal{O}(\epsilon^2),$$

while the eigenvalues of $\sigma_1^2 I - K_1^T K_1$ are of order ϵ. For sufficiently small ϵ, then, the matrix (21) is positive definite, and this gives the desired contradiction. □

Using this lemma and Theorem 2.2 we can now prove equality (3) for general matrices, when $k = 1$.

THEOREM 2.5. *Let C_0 and C_1 be arbitrary real m-by-n matrices. Then*

$$\min_{\alpha} \max_{\|w\|=1} \|(C_0 + \alpha C_1) w\| = \max_{\|w\|=1} \min_{\alpha} \|(C_0 + \alpha C_1) w\|. \tag{22}$$

Proof. We will use induction on the number of columns n. If $n = 1$, the result is clearly true. Assume it is true for matrices with $n-1$ columns, and now consider matrices with n columns. Suppose $\hat{\alpha}$ minimizes $\|C_0 + \alpha C_1\|$. Define $\hat{C} \equiv C_0 + \hat{\alpha} C_1$ and write the singular value decomposition of $\hat{C}$ as

$$\hat{C} = U \Sigma V^T,$$

where U, V, and Σ are as defined earlier. Assume that the singular values $\sigma_i, i = 1, \ldots, \min\{m, n\}$ satisfy

$$\sigma_1 = \cdots = \sigma_t > \sigma_{t+1} \geq \cdots,$$

and let V_t be the n-by-t matrix consisting of the first t columns of V, while $V_{t+1:n}$ is the n-by-$(n-t)$ matrix consisting of columns $t+1$ through n of V.

We will consider two cases. In the first case, assume that $t < n$. According to the lemma, $\hat{\alpha}$ minimizes

$$\|(C_0 + \alpha C_1)V_t\|,$$

and so, by the induction hypothesis,

$$\|(C_0 + \hat{\alpha}C_1)V_t\| = \max_{\|w\|=1} \min_{\alpha} \|(C_0 + \alpha C_1)V_t w\|.$$

If $\hat{w}$ is the t-vector for which this maximum is attained and if we define the n-vector $\tilde{w}$ to be $V_t\hat{w}$, then we have the desired result

$$\|C_0 + \hat{\alpha}C_1\| = \min_{\alpha} \|(C_0 + \alpha C_1)\tilde{w}\|.$$

Now suppose $t = n$. We can assume without loss of generality that $\sigma_1 = 1$, and then we have

$$\min_{\alpha} \|C_0 + \alpha C_1\| = \min_{\beta} \|U^T(C_0 + (\beta + \hat{\alpha})C_1)V\|$$

$$= \min_{\beta} \left\| \begin{pmatrix} I \\ 0 \end{pmatrix} + \beta \begin{pmatrix} U_n^T C_1 V \\ \\ U_{n+1:m}^T C_1 V \end{pmatrix} \right\| = 1,$$

where U_n consists of the first n columns of U and $U_{n+1:m}$ consists of columns $n+1$ through m of U. The same coefficient β that minimizes the norm of the entire matrix also minimizes the norm of the top n-by-n block of this matrix and so we have

$$\min_{\beta} \|I + \beta U_n^T C_1 V\| = 1.$$

From Theorem 2.2 it now follows that

$$\max_{\|w\|=1} \min_{\beta} \|(I + \beta U_n^T C_1 V)w\| = 1$$

and hence that (22) holds. □

Special cases of this theorem, as well as Theorem 2.3, were derived independently and proved in a different way by Joubert [5].

3. Further discussion. Extensive numerical testing of the inequalities in (1) and (2) for a variety of matrices suggests that they are, indeed, equalities. Theorem 2.5 proves this is so for $k = 1$, but we have been unable to prove (or disprove) this result for $k > 1$. The example (16) shows that the proof must rely on special properties of polynomials, since the result is not true for arbitrary noncommuting matrices, even if they are normal.

Acknowledgment. The authors thank Nick Trefethen for pointing out that the results of Theorem 2.3 are equivalent to known results about vector approximation and for indicating where these could be found in the literature.

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] A. GREENBAUM, *Comparison of splittings used with the conjugate gradient algorithm*, Numer. Math., 33 (1979), pp. 181–194.

[3] C. L. LAWSON, *Contributions to The Theory of Linear Least Maximum Approximations*, Ph.D. thesis, University of California at Los Angeles, 1961.

[4] A. GREENBAUM AND L. N. TREFETHEN, *GMRES/CR and Arnoldi/Lanczos as matrix approximation problems*, SIAM J. Sci. Comput., 15 (1994) pp. 359–368.

[5] W. JOUBERT, *A robust GMRES-based adaptive polynomial preconditioning algorithm for nonsymmetric linear systems*, SIAM J. Sci. Comput., 15 (1994) pp. 427–439.

[6] J. R. RICE, *The Approximation of Functions, Vol. 2: Nonlinear and Multivariate Theory*, Addison-Wesley, Reading, MA, 1969.

[7] 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.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 359–368, March 1994

008

GMRES/CR AND ARNOLDI/LANCZOS AS MATRIX APPROXIMATION PROBLEMS*

ANNE GREENBAUM† AND LLOYD N. TREFETHEN‡

Abstract. The GMRES and Arnoldi algorithms, which reduce to the CR and Lanczos algorithms in the symmetric case, both minimize $\| p(A)\, b \|$ over polynomials p of degree n. The difference is that p is normalized at $z = 0$ for GMRES and at $z = \infty$ for Arnoldi. Analogous "ideal GMRES" and "ideal Arnoldi" problems are obtained if one removes b from the discussion and minimizes $\| p(A) \|$ instead. Investigation of these true and ideal approximation problems gives insight into how fast GMRES converges and how the Arnoldi iteration locates eigenvalues.

Key words. GMRES, CR, Arnoldi, Lanczos, matrix approximation problem, normal matrix

AMS subject classifications. 65F10, 49K35

1. Introduction.

Since the 1950s it has been recognized that matrix iterative methods are naturally connected with approximation theory. The most familiar connections are between polynomial approximation and the numerous iterative methods that make use of Krylov subspaces, including the Richardson, Chebyshev, conjugate gradient, biconjugate gradient, CGNR, GMRES, CGS, Bi-CGSTAB, and QMR iterations. Sometimes rational approximation problems also arise, notably in the analysis of ADI iterations, circulant-preconditioned Toeplitz iterations, and Krylov subspace algorithms via Padé approximation. Recent references on these matters include [4], [8], [16], [25].

The approximation problems that are discussed in the linear algebra literature almost invariably involve scalar functions defined on subsets of the complex plane or, if the matrix A is symmetric, the real axis. The set in question is the spectrum $\Lambda(A)$ or an estimate of the spectrum. If A is normal, such reductions are sometimes exact in the sense that the behavior of the matrix iteration is determined exactly by the properties of the approximation problem. If A is not normal, however, they are always approximate. GMRES, for example, does not exactly solve any known approximation problem in the complex plane, when A is not normal. Between the approximation problem and the convergence of the matrix iteration there is a gap of size $\kappa(V)$, the condition number of a matrix of eigenvectors of A. When $\kappa(V)$ is large, predictions based on the approximation problem may have little bearing on the actual convergence of the matrix algorithm [16], [25].

The purpose of this paper is to explore a different kind of approximation problem that can also be associated with iterative linear algebra, involving matrices instead of scalars. Instead of asking how small a polynomial $p(z)$ can be on the set $\Lambda(A)$, we ask how small the norm $\| p(A) \|$ can be. Matrix approximation questions are implicit in much of the literature of matrix iterations; we certainly do not claim to be the first to consider them. However, they have received no discussion in print that we are aware of. We believe it is important to investigate these problems if one's goal is an understanding of matrix iterations that does not depend upon hidden assumptions of near-normality. At the same time, the consideration of matrix approximation problems preserves a familiar feature of scalar approximation problems, the removal from the analysis of the effects of the starting vector. Since most of the phenomena

*Received by the editors May 21, 1992; accepted for publication (in revised form) July 16, 1993. Part of the work of both authors was performed while visiting the Institute for Mathematics and its Applications at the University of Minnesota.

†Courant Institute of Mathematics Science, 251 Mercer St., New York, New York 10012 (greenbau@cims.nyu.edu).

‡Department of Computer Science, Cornell University, Ithaca, New York 14853 (LNT@cs.cornell.edu).
This work was supported by National Science Foundation grant DMS-9116110.

of greatest interest in iterative linear algebra depend mainly on the matrix, not the starting vector, this is a valuable simplification for most applications.

We shall concentrate on two algorithms for nonsymmetric matrix problems: GMRES, which solves systems of equations $Ax = b$, and Arnoldi, which computes eigenvalues of A. Our matrix approximation analogues of these processes are called the "ideal GMRES" and "ideal Arnoldi" problems. Mathematically, the new result presented here is a proof of the existence and uniqueness of ideal GMRES and Arnoldi approximants. (Existence is trivial, but uniqueness is surprisingly tricky.) In the final section we propose five questions whose answers might further advance our understanding of matrix iterations.

2. GMRES and Arnoldi. Throughout this paper N and $n < N$ are integers, A is an $N \times N$ matrix,[1] b is an N-vector, $\| \cdot \|$ is the 2-norm, and

$$P_n = \{\text{polynomials of degree} \le n \text{ with } p(0) = 1\},$$

$$P^n = \{\text{monic polynomials of degree } n\}.$$

The difference between P_n and P^n is that P_n is normalized at $z = 0$ and P^n at $z = \infty$.

GMRES [4], [23] is an algorithm that solves the following approximation problem[2] successively for $n = 1, 2, 3 \ldots$.

GMRES APPROXIMATION PROBLEM. *Find $p_* \in P_n$ such that*

$$\| p_*(A)b \| = \text{minimum}. \tag{1}$$

An equivalent statement is

$$b \approx \langle Ab, A^2b, \ldots, A^nb \rangle, \tag{1'}$$

where "$y \approx V$" denotes the problem of finding the best approximation with respect to $\| \cdot \|$ to the point y in the space V. This characterization of GMRES is well known. To explain it one notes that GMRES finds a vector x_n in the Krylov subspace $\mathcal{K}_n = < b, Ab, \ldots, A^{n-1}b >$ such that the residual $r_n = b - Ax_n$ has minimal norm over all $x \in \mathcal{K}_n$. This vector x_n can be represented in the form $x_n = q(A)b$ for some polynomial $q(z)$ of degree $n - 1$, and (1) comes upon writing $r_n = p_*(A)b$ with $p_*(z) = 1 - zq(z) \in P_n$.

The Arnoldi iteration [1], [4], [20]–[22] is an algorithm that solves the analogous problem involving P^n instead of P_n.

ARNOLDI APPROXIMATION PROBLEM *Find $p^* \in P^n$ such that*

$$\| p^*(A)b \| = \text{minimum}. \tag{2}$$

Equivalently,

$$A^nb \approx \langle b, Ab, \ldots, A^{n-1}b \rangle. \tag{2'}$$

The vector b is no longer the right-hand side of a system of equations, but an arbitrary initial vector. This characterization is also known, but perhaps not as widely known as it should be.[3] It can be readily proved as a consequence of the usual formulation of the Arnoldi process in terms of orthogonality. The Arnoldi iteration "finds p^*" in the sense that it constructs a Hessenberg matrix H_n of which p^* is the characteristic polynomial.

[1] Nothing essential changes if we take $N = \infty$ and let A be a bounded operator.

[2] We have assumed that the initial guess for the iteration is $x_0 = 0$. Arbitrary initial guesses can be handled by an easy modification.

[3] See Theorem 1 of [22]. The symmetric (Lanczos) case of this result was stated by Lanczos himself [11, p. 34] and also appears as Corollary 12-3-7 of [18], unfortunately with a typographical error.

If A is real and symmetric, the GMRES and Arnoldi iterations reduce to the conjugate residual (CR) and Lanczos iterations, respectively. Everything said in this paper about GMRES applies also to CR, and everything said about Arnoldi applies also to Lanczos.

A comparison of (1) and (2) suggests that from an approximation point of view, the difference between the GMRES and Arnoldi algorithms is slight. This analogy is rarely brought out in accounts of these algorithms, partly for historical reasons and partly because the usual applications of the two algorithms are different. Whereas GMRES is applied to solve systems of equations $Ax = b$, so that (1) comes quickly to mind as a description of it, the Arnoldi iteration is traditionally thought of as a method for estimating eigenvalues of A. The "Arnoldi eigenvalue estimates"[4] at step n are the eigenvalues of H_n, that is, the roots of p^*. But what the Arnoldi iteration actually does is solve (2); its connection with eigenvalues is indirect and approximate.

The formulations (1) and (2) provide elegant proofs of certain well-known properties of the GMRES and Arnoldi iterations. For example, one sees immediately from (1) and (2) that both of these iterations are essentially invariant under changes of scale ($A \to \alpha A, \alpha \in \mathbf{C}$) and under unitary similarity transformations ($A \to UAU^*$, $U^* = U^{-1}$). The Arnoldi iteration is also translation-invariant ($A \to A + \alpha I, \alpha \in \mathbf{C}$), since ∞ is translation-invariant, but GMRES is not, since 0 is not [11]. In these statements and throughout this paper, we ignore the effects of rounding errors.

3. Ideal GMRES and ideal Arnoldi. The GMRES and Arnoldi iterations depend on the starting vector b. However, one may remove b from the discussion and pose the following "ideal" approximation problems.

IDEAL GMRES APPROXIMATION PROBLEM. *Find $q_* \in P_n$ such that*

$$\|q_*(A)\| = \text{minimum}. \tag{3}$$

Equivalently,

$$I \approx \langle A, A^2, \ldots, A^n \rangle. \tag{3'}$$

IDEAL ARNOLDI APPROXIMATION PROBLEM. *Find $q^* \in P^n$ such that*

$$\|q^*(A)\| = \text{minimum}. \tag{4}$$

That is,

$$A^n \approx \langle I, A, \ldots, A^{n-1} \rangle. \tag{4'}$$

Whereas (1) and (2) are vector approximation problems, (3) and (4) involve matrices. Procedures for computing these polynomials, either actual or in our imaginations, might be called ideal GMRES and ideal Arnoldi algorithms. Some computations of this kind are discussed in §6.

We believe that studying these idealized problems may be a fruitful way to gain insight into the properties of Krylov subspace iterations in linear algebra. Our reasoning is as follows. The behavior of a GMRES or Arnoldi iteration is determined by two things: A and b. However, though the special properties of b are occasionally important, more often the features that one cares about do not differ very much from one choice of b to another. It is the properties of A

[4]The roots of p^* are Ritz values of A with respect to the Krylov subspace (2'). The roots of p_* have been called "pseudo-Ritz values" [4] and "roots of kernel polynomials" [14]. The Ritz values lie in the field of values of A, and the pseudo-Ritz values lie in the inverse of the field of values of A^{-1}; see [14].

that usually decide between an iteration that converges in 10 steps and one that requires 100 or 1000 (which in practice means it is time to look for a better preconditioner). By passing from (1)–(2) to (3)–(4) we disentangle this matrix essence of the process from the distracting effects of the initial vector and end up with a pair of elegant mathematical problems in the bargain.

The solutions to (1)–(2) and (3)–(4) are related by the following bounds. The proof of this theorem is easy; the four inequalities follow from the minimality properties (1), (3), (2), and (4), respectively.

THEOREM 1. *The true and ideal* GMRES *polynomials are related by*

$$\frac{\|p_*(A)b\|}{\|b\|} \leq \|q_*(A)\| \leq \|p_*(A)\|, \tag{5}$$

and the true and ideal Arnoldi polynomials are related by

$$\frac{\|p^*(A)b\|}{\|b\|} \leq \|q^*(A)\| \leq \|p^*(A)\|. \tag{6}$$

These two pairs of bounds are identical in form, but from the point of view of applications, the nature of the relationship between (4) and (2) is quite different from that between (3) and (1). The purpose of GMRES is to solve (1), not (3). The relevance of (3) is that it gives an upper bound on how slow the convergence may be, thanks to (5), and if the right-hand side b is "random enough," one may expect that this bound may be close to sharp. For an Arnoldi iteration aimed at estimating eigenvalues, the logic is reversed. One can take the view that the essence of the process by which an Arnoldi iteration locates eigenvalues is the solution not of (2) but of (4). The iteration solves (2) because that is what is computationally tractable, but the implicit hope is that if b is "random enough," the solution to (2) will be a good approximation to the solution to (4). It would be interesting to investigate how this point of view can be related to the existing theory of convergence of the Arnoldi iteration as developed by Saad [20], [22].

The ideal Arnoldi polynomial q^* might be called the degree n *Chebyshev polynomial of* A, in analogy to the notion in approximation theory of a Chebyshev polynomial of a subset of the complex plane, which is a monic polynomial that achieves minimal sup-norm on that set.[5] Another way to view q^* is as a *pseudo-annihilating polynomial* for A, i.e., a monic polynomial that maps A to a matrix of norm $\epsilon \approx 0$. According to the usual definition, an annihilating polynomial is a polynomial that annihilates A exactly. This is a fragile concept, however, ill posed with respect to perturbations of A and with little quantitative force. Krylov subspace iterations in numerical linear algebra are founded on the observation that for practical purposes, a pseudo-annihilating polynomial with parameter $\epsilon = 10^{-10}$ or 10^{-20} is as useful as an exact annihilating polynomial and may be of vastly lower degree.

4. The special case when A is normal. When A is normal, problems (1)–(4) reduce to standard problems of approximation theory. For any vector b we have

$$p(A)b = \sum_{j=1}^{N} \omega_j p(\lambda_j) v_j, \qquad \omega_j = v_j^T b,$$

where $\{\lambda_j\}$ and $\{v_j\}$ are a set of eigenvalues and corresponding orthonormal eigenvectors of A, and therefore

$$\|p(A)b\| = \left(\sum_{j=1}^{N} |\omega_j|^2 |p(\lambda_j)|^2 \right)^{1/2}.$$

[5] See Chapter 16 of [9]. This analogy becomes an identity if A is normal; see §4.

Thus the Arnoldi and GMRES problems (1)–(2) are equivalent to weighted least-squares approximation problems in the complex plane: find an appropriately normalized polynomial $p(z)$ that has minimal weighted 2-norm on $\Lambda(A)$ with respect to the discrete weight function $\{|\omega_j|^2\}$. As for the ideal Arnoldi and GMRES problems (3)–(4), the identity

$$\| p(A) \| = \sup_{\lambda \in \Lambda(A)} |p(\lambda)|$$

(for normal matrices only) shows that they are are equivalent to Chebyshev approximation problems in the complex plane: find a polynomial that has minimal supremum norm on $\Lambda(A)$. In particular, the ideal Arnoldi polynomial q_* is exactly the same as the Chebyshev polynomial for the set $\Lambda(A)$, mentioned in the last section.

Both weighted least-squares and Chebyshev approximation problems in the complex plane are well understood and discussed in many books. Existence and uniqueness of best approximations are easily proved (we defer a precise statement to the next section). In the Chebyshev case the computation of best approximations can be carried out by various methods such as linear programming, variations of the Remes algorithm, the Lawson algorithm, or other ideas; see [24] and the references therein. In the least-squares case the computation of best approximants is a matter of routine linear algebra. One can use a QR decomposition, for example, and that is exactly what GMRES does.

There are two reasons to pay special attention to the case in which A is normal. First, some matrices are normal or close enough to normal that the results one obtains are sometimes applicable in practice. In particular, the CR and Lanczos iterations fall in this category since symmetric matrices are normal. Second, most people's intuitions about the behavior of matrices are based on the normal case [26]. By studying how the normal case differs from the general case we obtain a valuable check on our intuitions.

5. Existence and uniqueness. We return now to problems (1)–(4) for matrices A that are arbitrary, i.e., not necessarily normal. The most fundamental questions to be asked about (1)–(4) are those of existence and uniqueness. For (1) and (2) the answers to both are straightforward and well known. For (3) and (4), existence is straightforward but uniqueness is not. So far as we are aware, though this seems surprising, the uniqueness of the solutions to (3) and (4) is a new result.

THEOREM 2. *The optimal polynomials p_*, p^*, q_*, q^* all exist. Provided that the minima in* (1)–(4) *are nonzero, and provided in the cases of* GMRES *and ideal* GMRES *that A is nonsingular, they are unique.*

Proof. Consider the formulations (1′)–(4′). In each case we have a problem of the form $y \approx V$, where V is a finite-dimensional subspace of a vector space W and $y \in W$. (For (1) and (2), V and W are spaces of N-vectors, whereas for (3) and (4) they are spaces of $N \times N$ matrices; generically we can speak of "vectors" in either case.) Existence of a closest point $v \in V$ to y follows by a standard compactness argument that can be found in any book on approximation theory. See, for example, [2, p. 20] or [13, p. 17].

The question of uniqueness of the polynomials p_*, p^*, q_*, q^* can be divided into two parts:

(a) Is the closest point $v \in V$ to y unique?

(b) Does it have a unique representation as a linear combination of the n vectors indicated in (1′)–(4′)?

Part (b) can be dispatched as follows. What we have to show is that the n vectors in question are linearly independent. Consider first the ideal Arnoldi problem (4′), and suppose to the contrary that $I, A, \ldots, A^{n-1}$ are linearly dependent. Then by multiplying by a power of A if necessary we can find a linear combination of them that is zero and in which the coefficient

of A^{n-1} is nonzero. Thus $A^{n-1} \in \langle I, A, \ldots, A^{n-2} \rangle$, which implies $A^n \in \langle A, A^2, \ldots, A^{n-1} \rangle$ and therefore $\|q^*(A)\| = 0$, contradicting the assumption that the minimum in (4) is nonzero. An analogous argument applies to (2′). For the GMRES problem (3′), suppose $A, A^2, \ldots, A^n$ are linearly dependent, which by a similar argument implies $A^n \in \langle A, A^2, \ldots, A^{n-1} \rangle$ and thus again $\|q^*(A)\| = 0$. If the constant term of q^* is nonzero, then dividing by that constant term yields a properly normalized GMRES polynomial q_* with $\|q_*(A)\| = 0$. On the other hand if the constant term is zero, then since A is nonsingular we can multiply by A^{-1} one or more times until it becomes nonzero. An analogous argument applies to (1′).

This brings us to part (a) of the proof of uniqueness. For problems (1′) and (2′) the uniqueness of v follows by another standard result in approximation theory, since the vector norm $\|\cdot\|$ is strictly convex. See, for example, [2, p. 23] or [13, p. 17]. The *matrix* norm $\|\cdot\|$, however, is not strictly convex, and, in general, matrix best approximation problems posed in this norm do not have unique solutions [6], [27]. The following proof of uniqueness for (3′) and (4′) depends on the special property that these problems concern approximation by matrix polynomials, not by arbitrary linear combinations of matrices.

Consider first the ideal Arnoldi problem (4). Suppose that q_1 and q_2 are two distinct solutions to (4), and let the minimal norm they attain be

$$\|q_1(A)\| = \|q_2(A)\| = C.$$

If we define $q(z) = \frac{1}{2}(q_1(z) + q_2(z))$, then $\|q(A)\| \le C$, so we must have $\|q(A)\| = C$ since q_1 and q_2 are minimal. Let $w_1, \ldots, w_J$ be a set of maximal right singular vectors for $q(A)$, i.e., a set of orthonormal vectors with

$$\|q(A)w_j\| = C, \qquad 1 \le j \le J,$$

with J as large as possible. For each w_j we must have

$$\|q_1(A)w_j\| \;=\; \|q_2(A)w_j\| \;=\; C$$

and

$$q_1(A)w_j \;=\; q_2(A)w_j,$$

for otherwise, by the strict convexity of the vector norm $\|\cdot\|$, we would have $\|q(A)w_j\| < C$. Thus

$$(q_2 - q_1)(A)w_j = 0, \qquad 1 \le j \le J.$$

Now since $(q_2 - q_1)(z)$ is not identically zero, we can multiply it by a scalar and a suitable power of z to obtain a monic polynomial $\Delta q \in P^n$ such that

$$\Delta q(A)w_j = 0, \qquad 1 \le j \le J.$$

For $\epsilon \in (0, 1)$, consider now the polynomial $q_\epsilon \in P^n$ defined by the convex linear combination

$$q_\epsilon(z) \;=\; (1-\epsilon)q(z) + \epsilon\, \Delta q(z).$$

If $w_{J+1}, \ldots, w_N$ denote the remainder of a set of N singular vectors of $q(A)$, with corresponding singular values $C > \sigma_{J+1} \ge \cdots \ge \sigma_N \ge 0$, then we have

$$\|q_\epsilon(A)w_j\| \;\le\; \begin{cases} (1-\epsilon)C & (1 \le j \le J), \\ (1-\epsilon)\sigma_{J+1} + \epsilon\,\|\Delta q(A)\| & (J+1 \le j \le N). \end{cases}$$

The first row is $< C$ for arbitrary ϵ, and the second row is $< C$ for sufficiently small ϵ, since $\sigma_{J+1} < C$. Since the singular vectors $w_1, \ldots, w_N$ form an orthonormal basis for $\mathbf{R}^N$, this implies that $\|q_\epsilon(A)\| < C$ for sufficiently small ϵ, contradicting the assumption that q_1 and q_2 are minimal.

For the ideal GMRES problem (3), the argument is the same except that from $q_2 - q_1$ we need to construct $\Delta q \in P_n$ rather than $\Delta q \in P^n$. If the constant term of $q_2 - q_1$ is nonzero, we do this by dividing by that term. If it is zero, we make use of the assumption that A is nonsingular and multiply by a suitable power of z^{-1}. □

6. Computations. If A is normal, the ideal Arnoldi and GMRES polynomials q^* and q_* are simply Chebyshev polynomials for the set $\Lambda(A)$, as noted in §4, and can be computed by various algorithms. If A is not normal, however, we know of no simple algorithm that is guaranteed to compute q^* and q_*. For simplicity, from now on we shall consider the ideal GMRES polynomial q_*; our remarks carry over straightforwardly to the ideal Arnoldi problem.

We have found that in many cases, q_* can be computed by using an optimization code to determine an initial vector b, with $\|b\| = 1$, for which $\|p_*(A)b\|$ is maximal at the prescribed step n. From Theorem 1 we have

$$\|p_*(A)b\| \leq \|q_*(A)\| \leq \|p_*(A)\| \tag{7}$$

for any b with $\|b\| = 1$. If a choice of b can be found for which $\|p_*(A)b\| = \|p_*(A)\|$, it follows that $p_* = q_*$. It is not known whether such a b always exists, but we conjecture that it does (see the first question of the next section). Maximizing the left-hand side of (7) seems to be easier in practice than minimizing the right-hand side, presumably because the latter problem is nonsmooth. We have carried out our computations using the Matlab optimization routine `fminu` [15] coupled with a GMRES subprogram to compute $\|p_*(A)b\|$ for a given vector b. Although several attempts with different initial guesses b are often required for the optimization code to succeed, it usually does so eventually.

Figure 1 illustrates the behavior of the ideal vs. true GMRES polynomials by a simple example—a matrix of Lenferink and Spijker [12], [26]. This is a nonnormal tridiagonal matrix of the form $\texttt{tridiag}((i+1)^{-1}, -3-2i, i+1)$, $i = 1, \ldots, N$. For the Lenferink–Spijker matrix of order $N = 16$, we computed the ideal GMRES polynomials q_* of degree 1 through 15 as well as the true GMRES polynomials p_* for five different random initial vectors. The thick curve in the figure represents the norms $\|q_*(A)\|$ as a function of n, and the thinner curves represent $\|p_*(A)b\|$ for the various vectors b. The GMRES curves lie below the ideal GMRES curve, as they must, but exhibit qualitatively the same shape. Our experiments have not been sufficiently extensive to draw conclusions about how GMRES and ideal GMRES convergence curves compare in general.

It is interesting to note that while the norm of the ideal GMRES polynomial for this problem decreases strictly monotonically, this does not always happen. For many problems, $\|q_*(A)\|$ remains exactly equal to 1 for a number of steps before it begins to decrease. The case $n = 1$ of this phenomenon is fully understood: one can show that $\|q_*(A)\| < 1$ at step 1 if and only if the field of values of A lies in an open half-plane with respect to the origin in $\mathbf{C}$ (see [3] and [16, §6]). For some problems, $\|q_*(A)\| = 1$ for steps 1 through $N-1$. This happens frequently with random matrices, for example. For such problems there is a right-hand side vector b for which the GMRES algorithm makes no progress whatsoever until step N.

A difference we have observed between the ideal GMRES polynomials for normal and nonnormal matrices is the following. If A is normal, then $q_*(A)$ must have at least $n+1$ equal maximal singular values. This follows from the fact that the degree n Chebyshev polynomial for a set in the complex plane always takes on its maximum absolute value in at least $n+1$

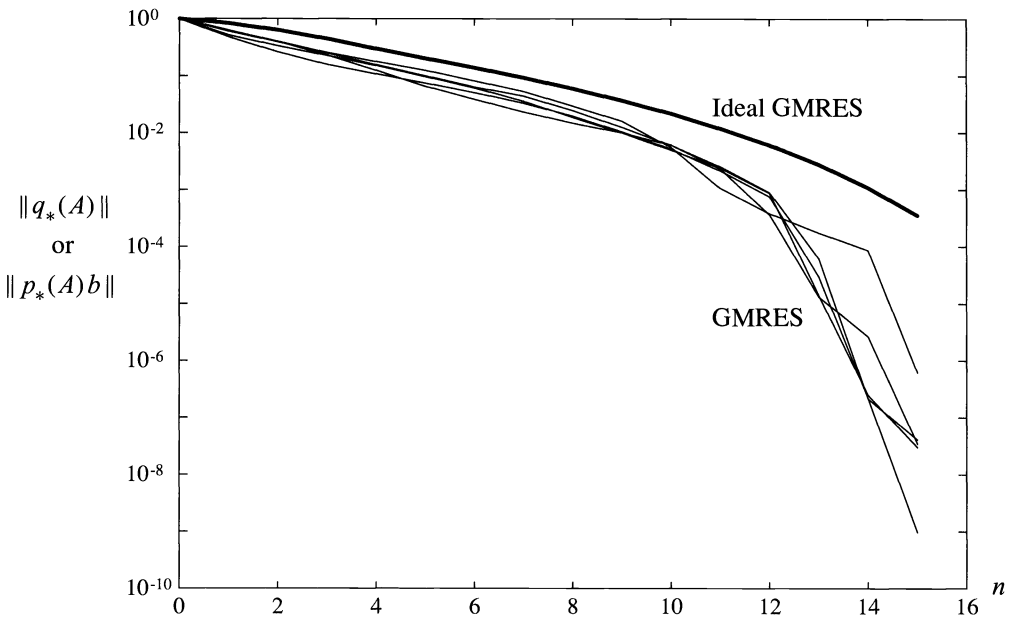


FIG. 1. *Convergence curves for the* 16 × 16 *Lenferink–Spijker matrix. The upper curve corresponds to ideal* GMRES *and the lower curves to standard* GMRES *with five different random initial vectors b.*

distinct points. In contrast, when A is not normal, our experiments indicate that after the initial phase with $\|q_*(A)\| = 1$ just mentioned, $q_*(A)$ usually has only one maximal singular value.

7. Open questions. Our work on ideal GMRES and Arnoldi approximations has raised more questions than it has answered. We shall close with a list of five questions that we consider particularly interesting. In each of the following, A is a matrix, b is a vector normalized by $\|b\| = 1$, and the *convergence curve* is the curve of $\|p_*(A)b\|$ or $\|q_*(A)\|$ as a function of the step number n. The questions are posed for GMRES, but they all have Arnoldi analogues.

1. *Is the envelope attained*? Theorem 1 asserts that the GMRES convergence curve lies below the ideal GMRES convergence curve, as illustrated in Fig. 1. Given n, does there exist an initial vector b such that these two curves intersect at step n, i.e., such that $\|p_*(A)b\| = \|q_*(A)\|$? The answer is known to be yes for symmetric matrices [5] and more generally for normal matrices [6], [10], and for arbitrary matrices at step $n = 1$ [6], [10]. It is also yes in the "generic" nonnormal case in which the maximal singular value of $q_*(A)$ is simple. Whether it is yes in all cases is not known. If it is, then the ideal GMRES convergence curve can be described as the upper envelope of the GMRES convergence curves corresponding to all initial vectors b.

2. *How close is the average case to the worst case*? Assuming that the envelope is attained in the sense above, there is still the question of how closely the GMRES and ideal GMRES convergence curves agree for typical starting vectors. If A is normal, it is easily argued that two agree typically to within a factor on the order of $\sqrt{N}$. We do not know what can be said for general A.

3. *What convergence curves are possible*? At step n, $\|q_*(A)\|$ must be at least as small as $\min_{1\le k\le n-1} \|q_k(A)\| \, \|q_{n-k}(A)\|$, where q_k denotes the ideal GMRES polynomial for A at step k. Geometrically this means that the ideal GMRES convergence curve is convex when plotted on a logarithmic scale. Are all convergence curves that satisfy this convexity constraint

possible? If not, how can one characterize those convergence curves that are possible?

4. *Can any matrix be simulated by a normal matrix*? Short of a full characterization of convergence curves, one may naturally ask if the possibilities for normal matrices are more restricted than for nonnormal matrices. In other words, are there convergence curves that can only be generated by a nonnormal matrix? For standard GMRES the answer has recently been proved to be no; any sequence $\| p_*(A)b \|$ as a function of n can be duplicated by another sequence $\| p_*(\tilde{A})b \|$ where $\tilde{A}$ is normal—in fact, unitary [7]. For ideal GMRES, the answer is unknown.

5. *Is there a variant of Lawson's algorithm for ideal* GMRES *approximation*? In our experience the "brute force" use of a general optimization program such as `fminu` to compute p_*, as described in the last section, is neither efficient nor reliable. As an alternative we have found that one can sometimes maximize the left-hand side of (7) by means of an iteration modeled on Lawson's algorithm, which is a method of iteratively reweighted least squares that has been proved convergent for problems of scalar L^∞ approximation [19]. We have obtained good results this way in many cases, but have not succeeded in developing a method of this kind that converges consistently (and consequently we will not provide details here). Can a matrix variant of Lawson's algorithm with guaranteed convergence be devised for the ideal GMRES problem?

Acknowledgments. We are happy to acknowledge discussions with Michael Overton, who showed us how the ideal Arnoldi and GMRES problems relate to more general problems of minimization of singular values of functions of matrices [17].

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] E. W. Cheney, *Introduction to Approximation Theory*, McGraw-Hill, New York, 1966.

[3] 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.

[4] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, in Acta Numerica 1992, Cambridge University Press, London, 1992.

[5] A. Greenbaum, *Comparison of splittings used with the conjugate gradient algorithm*, Numer. Math., 33 (1979), pp. 181–194.

[6] A. Greenbaum and L. Gurvits, *Max-min properties of matrix factor norms*, SIAM J. Sci. Comput., 15 (1994), pp. 348–358.

[7] A. Greenbaum and Z. Strakos, *Matrices that generate the same Krylov residual spaces*, Courant Institute Tech. Rep. 608, May, 1992; Proc. IMA Conf. on Iterative Methods for Sparse and Structured Problems, to appear.

[8] M. H. Gutknecht, *The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions, and the qd algorithm*, Preliminary Proc. Copper Mountain Conf. Iterative Methods, 1990, preliminary version.

[9] E. Hille, *Analytic Function Theory*, 2, Chelsea, New York, 1973.

[10] W. A. Joubert, *A robust* GMRES*-based adaptive polynomial preconditioning algorithm for nonsymmetric linear systems*, SIAM J. Sci. Comput., 15 (1994), pp. 427–439.

[11] C. Lanczos, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Stand., 49 (1952), pp. 33–53.

[12] H. W. J. Lenferink and M. N. Spijker, *On the use of stability regions in the numerical analysis of initial value problems*, Math. Comp., 57 (1991), pp. 221–237.

[13] G. G. Lorentz, *Approximation of Functions*, Chelsea, New York, 1986.

[14] T. A. Manteuffel and J. Otto, *On the roots of the orthogonal polynomials and residual polynomials associated with a conjugate gradient method*, J. Numer. Linear Algebra, to appear.

[15] The MathWorks, Inc., *MATLAB User's Guide*, The MathWorks, Inc., Natick, MA, 1987.

[16] 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.

[17] M. L. OVERTON, *Large-scale optimization of eigenvalues*, SIAM J. Optim., 2 (1992), pp. 88–120.
[18] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[19] J. R. RICE AND K. H. USOW, *The Lawson algorithm and extensions*, Math. Comp., 22 (1968), pp. 118–127.
[20] Y. SAAD, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.
[21] ———, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp.105–126.
[22] ———, *Projection methods for solving large sparse eigenvalue problems*, in Matrix Pencils, Lect. Notes in Math. 973, B. Kågström and A. Ruhe, eds., Springer-Verlag, New York, Berlin, 1983, pp. 121–144.
[23] Y. SAAD AND M. H. SCHULTZ, GMRES: *A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[24] P. T. P. TANG, *A fast algorithm for linear complex Chebyshev polynomial approximation*, Math. Comp., 51 (1988), pp. 721–739.
[25] L. N. TREFETHEN, *Approximation theory and numerical linear algebra*, in Algorithms for Approximation II, J. C. Mason and M. G. Cox, eds., Chapman and Hall, London, 1990.
[26] ———, *Pseudospectra of matrices*, in Numerical Analysis 1991, D. F. Griffiths and G. A. Watson, eds., Longman Scientific and Technical, Essex, UK, 1992.
[27] K. ZIĘTAK, *Properties of linear approximations of matrices in the spectral norm*, Linear Algebra Appl., 183 (1993), pp. 41–60.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 369–384, March 1994

009

ALTERNATING DIRECTION PRECONDITIONING FOR NONSYMMETRIC SYSTEMS OF LINEAR EQUATIONS*

GERHARD STARKE†

Abstract. Preconditioning strategies based on the application of the alternating direction implicit (ADI) method to large systems of linear equations of the form

$$(H + V)\mathbf{u} = \mathbf{f},$$

where both H and V can be "easily inverted," are presented and analyzed. Besides other applications, such systems arise naturally from finite difference discretizations of two-dimensional elliptic boundary value problems. The emphasis here is on the case where H and V are nonsymmetric.

The use of alternating direction preconditioning is especially attractive for massively parallel computers since, during each iteration, a large number of tridiagonal systems must be solved simultaneously. Numerical experiments are presented comparing ADI with other preconditioners for some examples of discretized nonselfadjoint elliptic boundary value problems including nonseparable cases.

Key words. ADI, nonsymmetric linear systems, elliptic boundary value problems, preconditioning, parallel computing

AMS subject classifications. 65F10, 65N30

1. Introduction.

We consider systems of linear equations of the form

$$(H + V)\mathbf{u} = \mathbf{f}, \tag{1.1}$$

where H and V are large, sparse, and, in general, nonsymmetric matrices. Moreover, both H and V are assumed to be "easily invertible," in the sense that the solution of linear systems with H and V is cheap compared to the solution of the original system (1.1). Such linear systems arise in connection with Lyapunov and Sylvester matrix equations in control theory and from finite difference and finite element discretizations of elliptic boundary value problems.

Our purpose in this paper is to study the use of the alternating direction implicit (ADI) method, which was originally proposed by Peaceman and Rachford [23] in 1955, as a preconditioning technique for Krylov subspace methods for *nonsymmetric* linear systems. It turns out that this leads to a very effective preconditioner, especially for two-dimensional elliptic difference equations. For separable nonselfadjoint elliptic equations, it can be shown that the number of iterations required to achieve a certain accuracy is proportional to $O(h^{-1/2})$ as $h \to 0$ for the stationary ADI method. Moreover, using l different parameters in a cyclic fashion, the iteration count can be reduced to $O(h^{-1/4})$ for $l = 2$ and to $O(\log h^{-1})$ for $l = \log h^{-1}$ similarly to the selfadjoint case. ADI preconditioning can, in contrast to many other techniques, e.g., symmetric successive overrelaxation (SSOR) or fast direct solvers, be implemented on a variety of parallel architectures in a straightforward way.

The idea to use ADI as a preconditioner goes back to the 1960s. In 1961, D'Yakonov [7] studied the combination of ADI and the Chebyshev method in an inner-outer iteration fashion. To solve symmetric positive definite linear systems, ADI was used as a preconditioner for the conjugate gradient method and was studied by Wachspress ([32, pp. 214], see also [33]). There, a number of steps of the ADI method, applied to a model problem (e.g., the Poisson equation), were used to precondition more general diffusion equations. Chin, Manteuffel, and de Pillis

*Received by the editors May 18, 1992; accepted for publication (in revised form) April 6, 1993. This work, done while the author was visiting the Institute for Mathematics and its Applications at the University of Minnesota during the first three months of 1992, was supported by the Alexander-von-Humboldt-Stiftung.

†Institut für Praktische Mathematik, Universität Karlsruhe, Englerstrasse 2, 76128 Karlsruhe, Germany (starke@ipmsun1.mathematik.uni-karlsruhe.de).

used ADI as a preconditioner for nonsymmetric systems that arise from convection-diffusion problems. Recently, a number of papers focused on the performance of the ADI method on different parallel architectures (for a survey and a list of references, see Johnsson, Saad, and Schultz [19]). A new approach to alternating direction preconditioning on multiprocessors for symmetric positive definite systems was presented by Jiang and Wong [18].

Our approach to the construction of an ADI preconditioner for nonsymmetric systems of the form (1.1) is motivated by the one in Chin, Manteuffel, and de Pillis [3]. There, the stationary ADI method is used to precondition the Chebyshev iteration, i.e., instead of solving the original system (1.1), the left preconditioned system

$$(V + \psi I)^{-1}(H + \varphi I)^{-1}(H + V)\mathbf{u} = (V + \psi I)^{-1}(H + \varphi I)^{-1}\mathbf{f}$$

or the right preconditioned system

$$(H + V)(V + \psi I)^{-1}(H + \varphi I)^{-1}\mathbf{v} = \mathbf{f}\,, \qquad \mathbf{u} = (V + \psi I)^{-1}(H + \varphi I)^{-1}\mathbf{v}$$

are considered. In §2 this type of *alternating direction preconditioning* will be introduced and generalized to higher degree, i.e., using polynomials

$$p_l(H) = (H + \varphi_1 I)\cdots(H + \varphi_l I)\,, \qquad q_l(V) = (V + \psi_1 I)\cdots(V + \psi_l I)$$

instead of $H + \varphi I$, $V + \psi I$, respectively. Furthermore, we point out relations between the preconditioned matrices and corresponding ADI iteration operators. After that, in §3, we discuss the question of how to choose the parameters $\varphi_1, \ldots, \varphi_l$ and $\psi_1, \ldots, \psi_l$ to get a well-preconditioned system. Convergence bounds for Krylov subspace methods (which exist, e.g., for the generalized minimal residual (GMRES) and the quasi-minimal residual (QMR) algorithms) will be used to show that it is a good idea to choose the parameters in such a way that the spectral radius of the associated ADI iteration operator is minimized. If H and V commute, this leads to a classical minimization problem for rational functions, the so-called "rational Zolotarev problem" (see, e.g., [27]). In the noncommutative case, only weaker bounds based on norm estimates are possible.

From §4 on, we will restrict ourselves to the *model problem* of *finite difference* discretizations of second-order elliptic equations on *rectangular two-dimensional domains* to illustrate our analysis and to test our techniques. We will, however, allow *nonseparable* and *nonselfadjoint* problems. Following Elman and Schultz ([12], see also Elman [10]), we consider elliptic equations of the type

$$-(au_x)_x - (bu_y)_y + cu_x + (cu)_x + du_y + (du)_y + eu = f \tag{1.2}$$

on a rectangular region $\Omega \subseteq \mathbb{R}^2$ (let, without loss of generality, Ω be the unit square $(0, 1) \times (0, 1)$) with Dirichlet boundary conditions on $\partial\Omega$. The functions $a(x, y)$, $b(x, y)$, $c(x, y)$, and $d(x, y)$ are assumed to be in $C^1(\Omega) \cap C(\overline{\Omega})$ and $e(x, y)$ to be continuous on $\overline{\Omega}$. The discretization of this elliptic boundary value problem by finite differences on an $n \times n$ grid leads to a linear system of the form (1.1), where H and V are the parts of the difference operator that arise from the derivatives in the x- and in y-direction, respectively. Both matrices, H and V, are usually large and sparse with a very regular structure. For example, if we use centered differences for the second-order terms as well as for the first-order terms, then H is a block diagonal matrix with tridiagonal blocks $H_{ii} \in \mathbb{R}^{n,n}$, $i = 1, \ldots, n$, and V is a block tridiagonal matrix where the blocks are diagonal matrices of order n.

It was already observed by Peaceman and Rachford in [23] that, for the solution of Poisson's equation on a rectangle, parameters for the ADI method can be found such that the

number of operations that is necessary to approximate the solution to a given accuracy is of the order $O(n^2 \log n)$ ($\log n$ is the required number of ADI iterations, each of which consists of the solution of n tridiagonal systems of order n). In §4, we will show that the same complexity can be achieved for general separable elliptic problems, i.e., if we have

(1.3)
$$a(x, y) \equiv a(x),\ b(x, y) \equiv b(y),\ c(x, y) \equiv c(x),\ d(x, y) \equiv d(y),\ e(x, y) = e_1(x) + e_2(y),$$

in (1.2). In practice, however, we might not want to increase the degree of the ADI preconditioner with n. Instead, it is often sufficient to choose $l = 2$, which already leads to an iteration count of $O(n^{1/4})$.

A lot of research has been done in the last 30 years to apply the alternating direction idea to more general problems than (1.2). Alternating direction methods in three dimensions were studied by Douglas [5], and the application to mixed finite element methods was discussed by Douglas, Durán, and Pietra [6]. Mixed derivatives in (1.2) can be treated with the technique proposed by Beam and Warming [1]. The use of the ADI method for the numerical solution of Navier–Stokes equations is described in Beam and Warming [2]; a variant for nonlinear variational problems is described in Glowinski [16]. The ADI method as an iterative method for the solution of Lyapunov matrix equations has been investigated by Wachspress in [34] (see also [17] for the application of the techniques presented in this paper to such matrix equations).

Fast direct algorithms for the solution of separable elliptic equations that also require $O(n^2 \log n)$ operations have been designed in the last 20 years (see, e.g., Swarztrauber [29] for block cyclic reduction, which is also applicable to nonselfadjoint problems). For nonseparable (and nonselfadjoint) problems, preconditioning by a "nearby" separable problem that is solved by a fast direct method was studied by Concus and Golub [4], Widlund [36], and Elman and Schultz [12].

Our motivation for this work is the fact that the ADI method is especially attractive for parallel computers. In contrast to other preconditioners that are currently used in this context, such as SSOR or the above mentioned fast methods, this technique has the advantage of being very easy and straightforward to (massively) parallelize on a variety of architectures. In each iteration, we first perform operations (multiplication by tridiagonal matrices and solving tridiagonal systems) independently on each column of the grid, and then, in the second half of the iteration, we do the same independently on each row. An architecture that seems to be perfectly suited for this method is the arrangement of the processors in a two-dimensional grid. There is, however, a bottleneck in the transfer between the half-steps since we must switch from rowwise to columnwise communication or vice versa. Techniques to overcome this problem on different multiprocessor architectures are discussed in Johnsson, Saad, and Schultz [19]. In any case, routines for the efficient solution of a large number of independent tridiagonal systems are now available in software libraries on parallel machines (see [30, Chap. 11] for the CM-5).

Finally, in §5, we present numerical experiments for some examples of discretized elliptic boundary value problems, including nonselfadjoint and nonseparable cases. Besides comparing ADI to other preconditioners, we will also use different Krylov subspace methods as acceleration schemes.

2. Alternating direction preconditioning. In this section we will show how to construct effective preconditioners, based on the ADI method, for linear systems of the form (1.1). Let us first consider the stationary case $l = 1$ that was studied earlier by Chin, Manteuffel, and de Pillis in [3]. Their preconditioning strategy has the form

$$(V + \psi I)^{-1}(H + \varphi I)^{-1}(H + V)\mathbf{u} = (V + \psi I)^{-1}(H + \varphi I)^{-1}\mathbf{f} \tag{2.1}$$

(left preconditioning) or

$$(H+V)(V+\psi I)^{-1}(H+\varphi I)^{-1}\mathbf{v}=\mathbf{f}\,,\qquad \mathbf{u}=(V+\psi I)^{-1}(H+\varphi I)^{-1}\mathbf{v} \tag{2.2}$$

(right preconditioning).

When using one of the Krylov subspace methods for nonsymmetric systems (like biconjugate gradient (Bi-CG), conjugate gradient squared (CGS), QMR, GMRES), the goal is to get the preconditioned problem "as close as possible" to the identity operator, in the sense that the eigenvalues of the operator approximate a positive constant. This can be motivated heuristically by the fact that the preconditioned operator being close to the identity means, roughly speaking, that we have already "inverted most" of the linear system. This will be discussed, in more detail, in terms of error bounds for Krylov subspace methods in §3.

Our aim is to choose the parameters φ and ψ in such a way that the spectrum of the preconditioned matrix

$$\Omega_1 := \sigma((V+\psi I)^{-1}(H+\varphi I)^{-1}(H+V)) = \sigma((H+V)(V+\psi I)^{-1}(H+\varphi I)^{-1}) \tag{2.3}$$

is as close as possible to a positive constant. The following theorem provides a connection between the distance of the preconditioned matrix to the identity and the spectral radius of the ADI iteration operator, which is given by

$$(V+\psi I)^{-1}(H-\psi I)(H+\varphi I)^{-1}(V-\varphi I)$$

(see [27]).

THEOREM 2.1. *With the constant $c=\varphi+\psi$, we have*

$$\max\{|1-c\tau| : \tau\in\Omega_1\} = \rho((H+\varphi I)^{-1}(H-\psi I)(V+\psi I)^{-1}(V-\varphi I))\,. \tag{2.4}$$

Proof. The relation (2.4) follows directly from

$$\begin{aligned} &I-(\varphi+\psi)(V+\psi I)^{-1}(H+\varphi I)^{-1}(H+V)\\ &=(V+\psi I)^{-1}(H+\varphi I)^{-1}(H-\psi I)(V-\varphi I)\,, \end{aligned} \tag{2.5}$$

and

$$\begin{aligned} &I-(\varphi+\psi)(H+V)(V+\psi I)^{-1}(H+\varphi I)^{-1}\\ &=(H-\psi I)(V-\varphi I)(V+\psi I)^{-1}(H+\varphi I)^{-1}\,. \end{aligned} \tag{2.6}$$

□

It will turn out to be important for the analysis of this ADI preconditioning technique that, from (2.5) and (2.6), instead of multiplying a vector $\mathbf{x}$ with the preconditioned matrix, one can compute

$$\mathbf{x}-(V+\psi I)^{-1}(H+\varphi I)^{-1}(H-\psi I)(V-\varphi I)\mathbf{x} \tag{2.7}$$

in the case of left preconditioning and

$$\mathbf{x}-(H-\psi I)(V-\varphi I)(V+\psi I)^{-1}(H+\varphi I)^{-1}\mathbf{x} \tag{2.8}$$

for right preconditioning, respectively. It is obvious that this requires exactly the same amount of work as the application of the preconditioned operator to $\mathbf{x}$. This implies that we could as well implement the preconditioner in this way as acceleration of the ADI iterative method.

Because of this representation of the ADI preconditioned matrix as a rational function in H and V, we will also use the term *rational preconditioning*.

In what follows, we restrict our attention to preconditioning from the right as in (2.2). Rational preconditioners of higher order can be constructed in the following way. Let the polynomials $p_l(z) = (z+\varphi_1)\cdots(z+\varphi_l)$ and $q_l(z) = (z+\psi_1)\cdots(z+\psi_l)$ be given, then the operator

$$I - q_l(-H)p_l(-V)[q_l(V)]^{-1}[p_l(H)]^{-1} \tag{2.9}$$

is the natural generalization of (2.2). Note that, for the construction of these alternating direction preconditioners, we did not make any assumption on commutativity of the matrices H and V.

3. Convergence bounds. Convergence bounds for conjugate gradient (CG)-like methods for nonsymmetric systems usually involve the size of the residual polynomial on the spectrum of the (preconditioned) operator. For example, for the GMRES algorithm (see Saad and Schultz [25]), if the corresponding coefficient matrix A can be diagonalized such that $A = T\Lambda T^{-1}$, one obtains

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{r}_0\|_2} \le \|T\|_2 \, \|T^{-1}\|_2 \min_{\Phi_m \in \Pi_m, \Phi_m(0)=1} \max_{z\in\sigma(A)} |\Phi_m(z)| \, . \tag{3.1}$$

Here, Π_m denotes the polynomials of degree $\le m$. A weaker bound for the error reduction that also involves

$$\min_{\Phi_m \in \Pi_m, \Phi_m(0)=1} \max_{z\in\sigma(A)} |\Phi_m(z)| \tag{3.2}$$

holds, under certain assumptions, for the QMR algorithm by Freund and Nachtigal [15]. If A is not diagonalizable, then similar bounds are valid that involve derivatives at certain eigenvalues (where the corresponding Jordan blocks have size > 1). A convergence result that depends on the size of $\Phi_m(z)$ and $\Phi_m'(z)$ was shown for the transpose-free QMR method proposed by Freund in [14] (see [13, Thm. 6]). In any case, the convergence of these iterative methods is governed by (3.2).

Since, for nonnormal matrices, the condition number $\|T\|_2 \, \|T^{-1}\|_2$ (or the corresponding expression for the QMR bound) can be quite large, it is important that convergence bounds are known that are not based on eigenvalues. These bounds are based on polynomial approximations on sets containing the spectrum of A like the ε-pseudo-spectra (see [22, Thm. 4]) or the field of values (see [9, Thm. 1]). We will not pursue this further and instead will follow the approach based on (3.1).

Our goal is to choose the parameters φ and ψ in such a way that

$$\min_{\Phi_m \in \Pi_m, \Phi_m(0)=1} \max_{z\in\Omega} |\Phi_m(z)| \tag{3.3}$$

with

$$\begin{aligned} \Omega_1 &= \sigma((H+\varphi I)^{-1}(H+V)(V+\psi I)^{-1}) \\ &= 1 - \sigma((H+\varphi I)^{-1}(H-\psi I)(V-\varphi I)(V+\psi I)^{-1}) \end{aligned}$$

(see (2.3)) is as small as possible. In the case that Ω_1 is a real interval (which is fulfilled, for example, if the eigenvalues of H and V are real and H and V commute), this leads to the problem

$$\min_{\varphi,\psi} \rho((H+\varphi I)^{-1}(H-\psi I)(V-\varphi I)(V+\psi I)^{-1}) \, , \tag{3.4}$$

i.e., minimizing the spectral radius of the corresponding ADI iteration matrix. If Ω_1 is not contained on the real line, and if we do not have any further information about the shape of Ω_1, our aim is still to choose φ and ψ in such a way that Ω_1 is as far away from the origin as possible. This again leads to (3.4). We remark that, for complex Ω_1, it is not rigorously justified that choosing the parameters φ and ψ in this way leads to the best possible Ω_1 for (3.3). Instead, it might be advantageous to minimize the largest imaginary part of Ω_1, although this would lead to a smaller distance to the origin. It was actually observed by Chin, Manteuffel, and de Pillis [3] that choosing the parameters according to (3.4) does not necessarily lead to the best (overall) convergence for ADI accelerated by the Chebyshev iteration in the case of complex spectra. For simplicity, we will, however, determine φ and ψ from (3.4). Analogously, for $l > 1$, (2.9) leads to

$$\min_{p_l, q_l \in \Pi_l} \rho([p_l(H)]^{-1} q_l(-H)\, p_l(-V)[q_l(V)]^{-1}) \,. \tag{3.5}$$

If H and V commute, we can transform H and V simultaneously into Schur form (see, e.g., Marcus and Minc [21, p. 77]), i.e., there exists a unitary matrix U such that $U^H H U$ and $U^H V U$ are both upper triangular. This assumption about commutativity is fulfilled, e.g., for discretized elliptic boundary value problems if the differential operator is separable (see (1.3)), and for Lyapunov and Sylvester matrix equations. Under these circumstances, we have

$$\begin{aligned}
&\rho([p_l(H)]^{-1} q_l(-H)\, p_l(-V)[q_l(V)]^{-1}) \\
&= \rho([U^H p_l(H) U]^{-1} U^H q_l(-H) U U^H p_l(-V) U [U^H q_l(V) U]^{-1}) \\
&\le \max_{\lambda \in \sigma(V)} \left| \frac{p_l(-\lambda)}{q_l(\lambda)} \right| \max_{\mu \in \sigma(H)} \left| \frac{q_l(-\mu)}{p_l(\mu)} \right|
\end{aligned}$$

leading to the rational minimization problem

$$\min_{p_l, q_l \in \Pi_l} \left[\max_{\lambda \in \sigma(V)} \left| \frac{p_l(-\lambda)}{q_l(\lambda)} \right| \max_{\mu \in \sigma(H)} \left| \frac{q_l(-\mu)}{p_l(\mu)} \right| \right] . \tag{3.6}$$

If $\sigma(V)$ and $\sigma(H)$ are replaced by arbitrary compact subsets of the complex plane, this is the so-called rational Zolotarev problem. For different approaches to this approximation problem, see [27] and references therein. The results in [27], and the numerical experiments in §5 of this paper, indicate that the main improvement of the convergence rate takes place when we use two parameters instead of one, especially for nonsymmetric problems. Explicit formulas for the optimal parameters for $l = 2$ are known for special regions, e.g., rectangles, enclosing $\sigma(H)$ and $\sigma(V)$ (see, again, [27]). On the other hand, these parameters can also be computed cheaply in many situations using minimization procedures. It is also possible to use the generalized Leja points discussed in [27] that solve the rational Zolotarev problem in an asymptotic sense. ADI with these Leja points as parameters could then be combined with the adaptive Chebyshev iteration [20] or iterative methods based on Faber polynomials (see [8] and [28]).

In the noncommutative case we can still get bounds for the location of the spectrum of the preconditioned operator using norm estimates. Obviously,

$$\rho([p_l(H)]^{-1} q_l(-H)\, p_l(-V)[q_l(V)]^{-1}) \le \|[p_l(H)]^{-1} q_l(-H)\|_2 \, \|p_l(-V)[q_l(V)]^{-1}\|_2 \,.$$

If H and V are symmetric, we have

$$\|[p_l(H)]^{-1} q_l(-H)\|_2 = \rho([p_l(H)]^{-1} q_l(-H)) = \max_{\mu \in \sigma(H)} \left| \frac{q_l(-\mu)}{p_l(\mu)} \right|$$

and

$$\|p_l(-V)[q_l(V)]^{-1}\|_2 = \rho(p_l(-V)[q_l(V)]^{-1}) = \max_{\lambda \in \sigma(V)} \left| \frac{p_l(-\lambda)}{q_l(\lambda)} \right| ,$$

which again leads to the rational expression (3.6) as an upper bound for the spectral radius of the ADI operator. It should be remarked that these norm estimates for the noncommutative case depend on the fact that we apply the ADI preconditioner in the form (2.9), i.e., in such a way that the terms with H and V are separated.

If H and V are nonsymmetric (and do not commute), then only much weaker results can be obtained. For the case $l = 1$, with real parameters φ and ψ, we get

$$\|(H+\varphi I)^{-1}(H-\psi I)\|_2^2 = \max_{\|\mathbf{x}\|_2=1} \frac{\|(H-\psi I)\mathbf{x}\|_2^2}{\|(H+\varphi I)\mathbf{x}\|_2^2}$$

$$= \max_{\|\mathbf{x}\|_2=1} \frac{\mathbf{x}^T(H^T-\psi I)(H-\psi I)\mathbf{x}}{\mathbf{x}^T(H^T+\varphi I)(H+\varphi I)\mathbf{x}} = \max_{\|\mathbf{x}\|_2=1} \frac{\|H\mathbf{x}\|_2^2 - \psi\mathbf{x}^T(H+H^T)\mathbf{x} + \psi^2}{\|H\mathbf{x}\|_2^2 + \varphi\mathbf{x}^T(H+H^T)\mathbf{x} + \varphi^2}$$

$$\le \max_{\|\mathbf{x}\|_2=1} \frac{\|H\mathbf{x}\|_2 - 2\psi \min \mathrm{Re}W(H) + \psi^2}{\|H\mathbf{x}\|_2 + 2\varphi \min \mathrm{Re}W(H) + \varphi^2} \le \frac{\|H\|_2^2 - 2\psi \min \mathrm{Re}W(H) + \psi^2}{\|H\|_2^2 + 2\varphi \min \mathrm{Re}W(H) + \varphi^2},$$

where $W(H)$ and $W(V)$ denote the field of values of the matrices H and V. Analogously, we obtain

$$\|(V-\varphi I)(V+\psi I)^{-1}\|_2^2 \le \frac{\|V\|_2^2 - 2\varphi \min \mathrm{Re}W(V) + \varphi^2}{\|V\|_2^2 + 2\psi \min \mathrm{Re}W(V) + \psi^2},$$

which leads to

$$\begin{aligned} &\rho((H+\varphi I)^{-1}(H-\psi I)(V-\varphi I)(V+\psi I)^{-1}) \\ &\le \frac{\|H\|_2^2 - 2\psi \min \mathrm{Re}W(H) + \psi^2}{\|H\|_2^2 + 2\varphi \min \mathrm{Re}W(H) + \varphi^2} \cdot \frac{\|V\|_2^2 - 2\varphi \min \mathrm{Re}W(V) + \varphi^2}{\|V\|_2^2 + 2\psi \min \mathrm{Re}W(V) + \psi^2}. \end{aligned} \tag{3.7}$$

4. Application to elliptic boundary value problems. We now apply the bounds derived in the last section to the elliptic boundary value problem (1.2). We use centered difference approximations for all the derivatives, which means that the operators H and V are defined by

$$\begin{aligned} [Hu]_{i,j} &= (a_{i,j+1/2} + a_{i,j-1/2})u_{i,j} - a_{i,j+1/2}u_{i,j+1} - a_{i,j-1/2}u_{i,j-1} \\ &\quad + [(c_{i,j} + c_{i,j+1})u_{i,j+1} - (c_{i,j} + c_{i,j-1})u_{i,j-1}]h/2 + e_{i,j}u_{i,j}h^2/2, \\ [Vu]_{i,j} &= (b_{i+1/2,j} + b_{i-1/2,j})u_{i,j} - b_{i+1/2,j}u_{i+1,j} - b_{i-1/2,j}u_{i-1,j} \\ &\quad + [(d_{i,j} + d_{i+1,j})u_{i+1,j} - (d_{i,j} + d_{i-1,j})u_{i-1,j}]h/2 + e_{i,j}u_{i,j}h^2/2, \end{aligned} \tag{4.1}$$

where $h = 1/(n+1)$. Let the numbers $\underline{a}, \overline{a}, \underline{b}, \overline{b}, \overline{c}$, and $\overline{d}$ be given such that

$$0 < \underline{a} \le a(x,y) \le \overline{a}, \quad 0 < \underline{b} \le b(x,y) \le \overline{b},$$
$$|c(x,y)| \le \overline{c}, \quad |d(x,y)| \le \overline{d}$$

on $(0,1) \times (0,1)$. With these bounds we obtain, for any matrix representation of the difference operators defined by (4.1),

(4.2)

$$\begin{aligned} \mathrm{Re}\sigma(H) \subseteq \mathrm{Re}W(H) \subseteq \sigma((H+H^T)/2) \subseteq [2\underline{a}(1-\cos\pi h), 2\overline{a}(1+\cos\pi h)] =: [\alpha_H, \beta_H], \\ \mathrm{Re}\sigma(V) \subseteq \mathrm{Re}W(V) \subseteq \sigma((V+V^T)/2) \subseteq [2\underline{b}(1-\cos\pi h), 2\overline{b}(1+\cos\pi h)] =: [\alpha_V, \beta_V], \\ \mathrm{Im}\sigma(H) \subseteq \mathrm{Im}W(H) \subseteq \sigma(i(H-H^T)/2) \subseteq [-2h\overline{c}, 2h\overline{c}] =: [-\gamma_H, \gamma_H], \\ \mathrm{Im}\sigma(V) \subseteq \mathrm{Im}W(V) \subseteq \sigma(i(V-V^T)/2) \subseteq [-2h\overline{d}, 2h\overline{d}] =: [-\gamma_V, \gamma_V]. \end{aligned}$$

In what is to follow, we will also use

$$\alpha := \min\{\alpha_H, \alpha_V\} = 2\min\{\underline{a}, \underline{b}\}(1 - \cos \pi h) ,$$
$$\beta := \max\{\beta_H, \beta_V\} = 2\max\{\overline{a}, \overline{b}\}(1 + \cos \pi h) ,$$
$$\gamma := \max\{\gamma_H, \gamma_V\} = 2h \max\{\overline{c}, \overline{d}\} .$$

Note that $\alpha = O(h^2)$ and $\gamma = O(h)$ for $h \to 0$.

We are interested in the location of the eigenvalues of the preconditioned operator as $h \to 0$. It can be shown that the matrices H and V can be transformed into real symmetric matrices if $h\overline{c} < 1$ and $h\overline{d} < 1$, respectively. This implies that, for sufficiently small h, the spectra of H and V will be contained on the real line, more precisely,

$$\sigma(H) \subseteq [\alpha, \beta] , \qquad \sigma(V) \subseteq [\alpha, \beta] .$$

Let us first investigate the separable case, i.e., we assume that (1.3) is fulfilled. In this case, it is well known that H and V commute (see [31, §7.3] and [27] for a different point of view in terms of matrix equations). Let φ^* and ψ^* denote the optimal ADI parameters with respect to $\sigma(H)$ and $\sigma(V)$ (see the following section for questions on how to find these parameters); then, for $\varphi = \psi = \sqrt{\alpha\beta}$, we have

$$\max_{\lambda\in\sigma(V)} \left|\frac{\lambda - \varphi^*}{\lambda + \psi^*}\right| \max_{\mu\in\sigma(H)} \left|\frac{\mu - \psi^*}{\mu + \varphi^*}\right| \le \max_{\lambda\in\sigma(V)} \left|\frac{\lambda - \sqrt{\alpha\beta}}{\lambda + \sqrt{\alpha\beta}}\right| \max_{\mu\in\sigma(H)} \left|\frac{\mu - \sqrt{\alpha\beta}}{\mu + \sqrt{\alpha\beta}}\right|$$
$$\le \max_{z\in[\alpha,\beta]} \left|\frac{z - \sqrt{\alpha\beta}}{z + \sqrt{\alpha\beta}}\right|^2 = \left(\frac{\sqrt{\beta} - \sqrt{\alpha}}{\sqrt{\beta} + \sqrt{\alpha}}\right)^2 .$$

Hence, by (2.3), we have, for sufficiently small h, for the spectrum Ω_1 of the ADI preconditioned operator,

$$\Omega_1 \subseteq \left[1 - \left(\frac{\sqrt{\beta} - \sqrt{\alpha}}{\sqrt{\alpha} + \sqrt{\beta}}\right)^2 , 1 + \left(\frac{\sqrt{\beta} - \sqrt{\alpha}}{\sqrt{\alpha} + \sqrt{\beta}}\right)^2\right] = \left[\frac{4\sqrt{\alpha\beta}}{(\sqrt{\alpha} + \sqrt{\beta})^2}, \frac{\alpha + \beta}{(\sqrt{\alpha} + \sqrt{\beta})^2}\right] .$$

By straightforward computation of the size of the associated Chebyshev polynomial, this leads to

$$\min_{\Phi_m\in\Pi_m, \Phi_m(0)=1} \max_{z\in\Omega_1} |\Phi_m(z)|$$
$$\le \frac{2(\sqrt{\beta} - \sqrt{\alpha})^{2m}}{[(\sqrt{\beta} + \sqrt{\alpha})^2 + \sqrt{8}(\alpha\beta)^{1/4}(\alpha + \beta)^{1/2}]^m + [(\sqrt{\beta} + \sqrt{\alpha})^2 - \sqrt{8}(\alpha\beta)^{1/4}(\alpha + \beta)^{1/2}]^m}$$
$$\approx 1 - [(4 - \sqrt{2})m + \sqrt{2}m^2] \left(\frac{\alpha}{\beta}\right)^{1/2}$$

Using (3.1), we observe that, to ensure that the residual norm is reduced by a factor of ε,

$$(4 - \sqrt{2})m + \sqrt{2}m^2 \le \left(\frac{\beta}{\alpha}\right)^{1/2} \left(1 - \frac{\varepsilon}{\|T\|_2 \, \|T^{-1}\|_2}\right)$$

must be fulfilled. Since $\alpha = O(h^2)$, this implies that the number of GMRES iterations m is proportional to $n^{1/2}$ (compared to n if we use ADI as an iteration scheme). Similarly, for $l = 2$, if $p_2^*, q_2^* \in \Pi_2$ are the optimal polynomials for (3.6), then

$$\max_{\lambda\in\sigma(V)} \left|\frac{p_2^*(-\lambda)}{q_2^*(\lambda)}\right| \max_{\mu\in\sigma(H)} \left|\frac{q_2^*(-\mu)}{p_2^*(\mu)}\right|$$

$$\leq \max_{z\in[\alpha,\beta]} \left| \frac{z^2 - \sqrt{2}(\alpha\beta)^{1/4}(\alpha+\beta)^{1/2}z + \alpha\beta}{z^2 + \sqrt{2}(\alpha\beta)^{1/4}(\alpha+\beta)^{1/2}z + \alpha\beta} \right|^2 = \left(\frac{(\alpha+\beta)^{1/2} - \sqrt{2}(\alpha\beta)^{1/4}}{(\alpha+\beta)^{1/2} + \sqrt{2}(\alpha\beta)^{1/4}} \right)^2 .$$

Hence, for the spectrum Ω_2 of the corresponding preconditioned system, we have

$$\Omega_2 \subseteq \left[1 - \left(\frac{(\alpha+\beta)^{1/2} - \sqrt{2}(\alpha\beta)^{1/4}}{(\alpha+\beta)^{1/2} + \sqrt{2}(\alpha\beta)^{1/4}} \right)^2 , 1 + \left(\frac{(\alpha+\beta)^{1/2} - \sqrt{2}(\alpha\beta)^{1/4}}{(\alpha+\beta)^{1/2} + \sqrt{2}(\alpha\beta)^{1/4}} \right)^2 \right]$$

$$= \left[\frac{4\sqrt{2}(\alpha\beta)^{1/4}(\alpha+\beta)^{1/2}}{[(\alpha+\beta)^{1/2} + \sqrt{2}(\alpha\beta)^{1/4}]^2}, \frac{2[\sqrt{\alpha}+\sqrt{\beta}]^2}{[(\alpha+\beta)^{1/2} + \sqrt{2}(\alpha\beta)^{1/4}]^2} \right].$$

In the same way as for $l = 1$, we derive that the number of GMRES iterations m needed to reduce the error by a given factor is proportional to $n^{1/4}$ (compared to $n^{1/2}$ if we use two-parameter ADI as an iteration scheme). Generally, if we use $l = 2^k$ optimal parameters, the number of required iterations is of the order $n^{1/2^k}$ for the ADI method and of the order $n^{1/2^{k+1}}$ for GMRES (or CG in the symmetric case) with ADI preconditioner. If we choose $l = \log n$ parameters for a problem of size n, it is even possible to bound the number of ADI cycles (and the number of GMRES steps using this ADI preconditioner of degree $l = \log n$) by a constant. This means that we are able to get the same asymptotic operation count of $O(n^2 \log n)$ as we can with fast direct methods.

Let us investigate the nonseparable case when the elliptic equation is selfadjoint. At the end of §3, we showed that

$$\rho([p_l(H)]^{-1} q_l(-H) p_l(-V)[q_l(V)]^{-1}) \leq \max_{\lambda\in\sigma(V)} \left| \frac{p_l(-\lambda)}{q_l(\lambda)} \right| \max_{\mu\in\sigma(H)} \left| \frac{q_l(-\mu)}{p_l(\mu)} \right|$$

also holds under these circumstances. Using the above analysis, this implies that the spectrum of the ADI preconditioned matrix for $l = 1$, Ω_1, is contained in a circle with center 1 and radius

$$\left(\frac{\sqrt{\beta} - \sqrt{\alpha}}{\sqrt{\beta} + \sqrt{\alpha}} \right)^2 = 1 - O(h) \text{ for } (h \to 0) .$$

Therefore, the number of required ADI iterations is of the order n, and, using this bound, the same order is obtained for GMRES with ADI preconditioning. Analogously, for $l = 2$, the number of iterations is proportional to $n^{1/2}$, and, using $l = \log n$ parameters, an iteration count of $\log n$ is achievable. Again, the above norm estimates for the noncommutative case are only valid because we apply the ADI preconditioner in the form (2.9) such that the terms with H and V are separated. If we use the cyclic form of ADI (as an iterative scheme or as preconditioner), the situation is different. In fact, Price and Varga [24] constructed an example of a diffusion equation, where the cyclic ADI method with two parameters fails to converge. Using the technique by Widlund [35] of appropriately scaling the diffusion equation or, equivalently, varying the ADI parameters with the space variables, it is possible to bound the number of iterations by $O(\log n)$ also for the cyclic ADI version.

In the nonselfadjoint (and nonseparable) case, using the bounds (4.2) in (3.7), we obtain

$$\rho((H + \varphi I)^{-1}(H - \psi I)(V - \varphi I)(V + \psi I)^{-1})$$

$$\leq \frac{(\beta+\gamma)^2 - 2\alpha\psi + \psi^2}{(\beta+\gamma)^2 + 2\alpha\varphi + \varphi^2} \cdot \frac{(\beta+\gamma)^2 - 2\alpha\varphi + \varphi^2}{(\beta+\gamma)^2 + 2\alpha\psi + \psi^2}.$$

This quantity is minimized for $\varphi = \psi = \beta + \gamma$, leading to

$$\rho((H+\varphi I)^{-1}(H-\psi I)(V-\varphi I)(V+\psi I)^{-1}) \leq \left(\frac{\beta+\gamma-\alpha}{\beta+\gamma+\alpha}\right)^2.$$

Using this bound, no improvement of the $O(n^2)$ dependence of the number of iterations for GMRES, applied directly to $(H+V)\mathbf{u} = \mathbf{f}$, can be guaranteed. Our numerical experiments, however, indicate that ADI preconditioning works as well in the separable case as it works for many nonseparable examples.

5. Numerical experiments. This final section contains numerical experiments carried out for certain examples of elliptic boundary value problems (1.2) including nonselfadjoint and nonseparable cases. In all the examples and for all preconditioners, we will precondition from the right. All the tables show the number of iterations that is needed to reduce the Euclidean norm of the starting residual by a factor of 10^{-6}. The entries of the right-hand side of the matrix problem are chosen randomly with coefficients uniformly distributed in the interval $[-1, 1]$; the initial guess is always the zero matrix.

Example 1. We first consider the Poisson equation, i.e., $a(x, y) = b(x, y) \equiv 1$ and $c(x, y) = d(x, y) = e(x, y) \equiv 0$.

The following conclusions can be drawn from a glance at Table 5.1. First, ADI *can* obviously be accelerated significantly by the CG method. Note that the additional work per iteration step of CG with ADI(l) preconditioning compared to ADI(l) consists in two scalar products of vectors of size n^2, which we may assume to be a fraction of the work required for the matrix-vector multiplication and tridiagonal linear system solving. Second, the convergence becomes significantly faster as we increase the number of parameters. The work for one iteration step of CG with ADI(2) preconditioning is essentially twice as much as for CG with ADI(1) preconditioning (actually a little less since the CG scalar products are only computed once). But the number of required CG/ADI(2) iterations is less than half the CG/ADI(1) iteration counts and remains almost constant as the number of unknowns increases (actually, the dependency of the number of iterations on $n^{1/4}$ can be observed quite nicely). While the step from two to four parameters still pays for the ADI iteration, this is not the case when using ADI as a preconditioner.

TABLE 5.1
Iteration counts for Example 1.

h	CG	ADI(1)	ADI(2)	ADI(4)	CG ADI(1)	CG ADI(2)	CG ADI(4)
1/16	39	32	7	3	15	6	3
1/32	77	70	11	4	21	7	3
1/64	157	139	16	4	28	9	4
1/128	309	255	20	6	46	12	5

Example 2. Let us now turn to the nonsymmetric (but still separable) problem

$$-\Delta u + \tau(xu_x + yu_y) + \eta u = f \tag{5.1}$$

in $(0, 1)\times(0, 1)$ with Dirichlet boundary conditions. This is Example 6.1 in [14] and is obtained from (1.2) if we set $a(x, y) = b(x, y) \equiv 1$, $c(x, y) = \frac{\tau}{2}x$, $d(x, y) = \frac{\tau}{2}y$, and $e(x, y) \equiv \eta - \tau$.

The linear system, resulting from discretization by central differences, is symmetrizable when $\frac{\tau}{2}h \leq 1$; otherwise we get (some) complex eigenvalues. For our experiments, we did not only use different preconditioners (ADI(1), ADI(2), block SSOR and a fast Fourier transform (FFT)-based fast Poisson solver (FPS)) but also different acceleration schemes; in particular, GMRES (without restarts) [25], CGS [26], and TFQMR (transpose-free QMR) [14]. We first choose $\tau = \eta = 200$. Figures 5.1 and 5.2 show the convergence behavior for different iterative methods and different preconditioners, respectively. The relative residual norm $\|\mathbf{r}\|/\|\mathbf{r}_0\|$ is plotted versus computational work. The latter is measured in units equivalent to the number of operations required for one half-step of the ADI iteration, i.e., the solution of n tridiagonal systems of order n plus n matrix-vector products with matrices of size n.

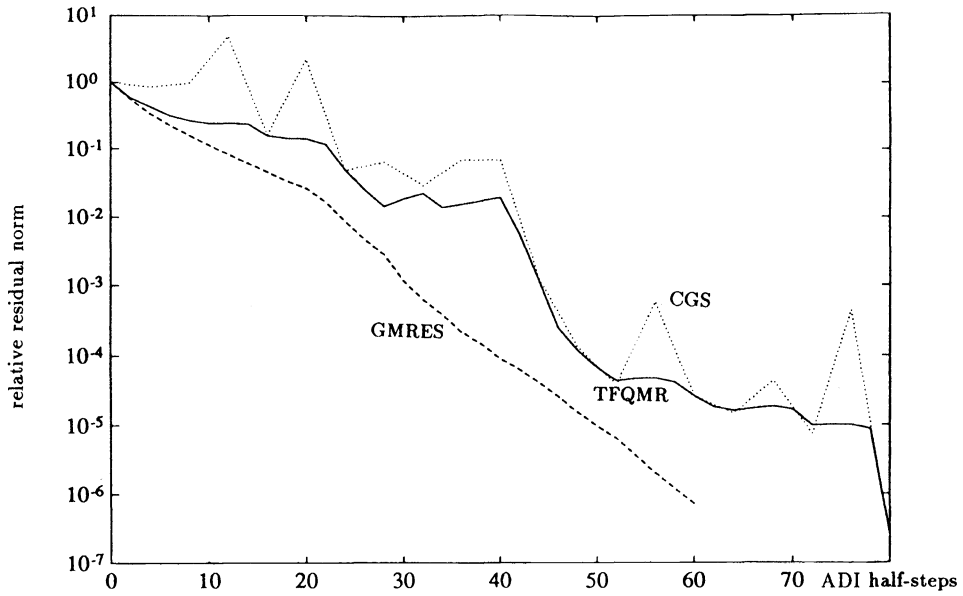


FIG. 5.1. *Example* 2 ($h = 1/64$) : ADI(1) *preconditioning, different iterative methods.*

From Table 5.2 and Fig. 5.1, we see that, although all the methods seem to work well with ADI preconditioning, using GMRES gives the fastest convergence (note that one step of GMRES involves only one multiplication, while one CGS iteration requires two multiplications with the preconditioned matrix). However, since, for the full GMRES algorithm, the number of SAXPY operations and inner products per GMRES step is growing linearly with the iteration count, this part of the computation amounts to a significant part of the work. Although the behavior of the residual norm is much more erratic, CGS reaches the desired accuracy after about the same time as TFQMR. (The amount of work is about the same for one CGS iteration and two TFQMR iterations.)

Let us now turn to the performance of the different preconditioners (see Tables 5.2, 5.3, and 5.4 and Fig. 5.2). Since (block) SSOR preconditioning involves the same amount of work per iteration as ADI(1), the results show that SSOR is preferable for any size of h in this example. Obviously, the results obtained by using an FPS as preconditioner are not satisfactory in this example (note that not only is the iteration count higher, but also that each iteration step requires more operations).

The iteration counts in Tables 5.2, 5.3, and 5.4 should be sufficient to illustrate that the dependence of the rate of convergence on the preconditioner is essentially the same for

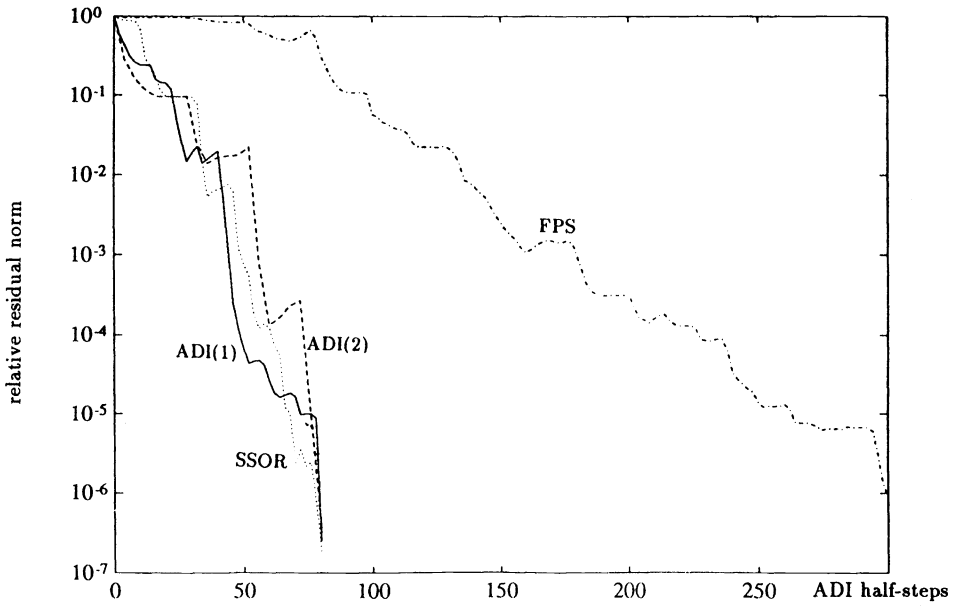


FIG. 5.2. *Example* 2 ($h = 1/64$) : TFQMR, *different preconditioners.*

TABLE 5.2
Iteration counts for Example 2, $\tau = \eta = 200$.

h	ADI(1)	ADI(2)	GMRES ADI(1)	GMRES ADI(2)	GMRES SSOR	GMRES FPS
1/16	37	18	27	13	99	67
1/32	31	15	23	13	51	79
1/64	35	20	30	16	30	86
1/128	62	25	34	19	41	93

TABLE 5.3
Iteration counts for Example 2, $\tau = \eta = 200$.

h	CGS	CGS ADI(1)	CGS ADI(2)	CGS SSOR	CGS FPS
1/16	49	19	9	93	69
1/32	71	14	8	34	75
1/64	133	20	10	20	79
1/128	253	24	13	33	81

TABLE 5.4
Iteration counts for Example 2, $\tau = \eta = 200$.

h	TFQMR	TFQMR ADI(1)	TFQMR ADI(2)	TFQMR SSOR	TFQMR FPS
1/16	98	38	18	181	134
1/32	153	28	15	62	144
1/64	258	40	20	40	150
1/128	502	47	24	64	160

GMRES, CGS, and TFQMR. From now on, we will restrict ourselves to TFQMR as our basic iterative method.

The results in Table 5.5 were obtained for the same example with the choice of parameters as in [14, Ex. 6.1] ($\tau = 100$ and $\eta = -200$) to show that ADI preconditioning also works well if the linear system has (some) eigenvalues in the left half-plane. Note that, as h gets smaller, the iteration count for the unpreconditioned method increases much faster than the one for ADI(1). On the other hand, preconditioning by an FPS and, for some h, also SSOR did not show any signs of convergence at all after a reasonable number of iterations.

TABLE 5.5
TFQMR *iteration counts for Example* 2, $\tau = 100$, $\eta = -200$.

h	No prec.	ADI(1)	SSOR	FPS
1/16	130	102	—	—
1/32	360	69	76	—
1/64	292	78	84	—
1/128	842	104	—	—

Example 3. Let us now turn to the nonseparable problem

$$-\Delta u + \tau y(1 - x^2)u_x - \tau x(1 - y^2)u_y = f,$$

which has the physical interpretation of modelling circular flow of a fluid around a point (see Elman and Golub [11]). To derive this equation from the general formulation (1.2), we set $a(x, y) = b(x, y) = 1, c(x, y) = \frac{\tau}{2}y(1 - x^2), d(x, y) = -\frac{\tau}{2}x(1 - y^2)$, and $e(x, y) = 0$. Again, we consider the elliptic differential equation on $(0, 1) \times (0, 1)$ with Dirichlet boundary conditions. For $\tau = 200$, the convergence behavior is illustrated in Table 5.6 and Fig. 5.3.

TABLE 5.6
TFQMR *iteration counts for Example* 3, $\tau = 200$.

h	No prec.	ADI(1)	ADI(2)	SSOR	FPS
1/16	76	19	15	102	109
1/32	133	40	26	78	132
1/64	235	82	24	62	124
1/128	466	123	30	80	131

Example 4. In our final example, we look at the nonseparable equation

$$-(e^{-2xy}u_x)_x - (e^{-2xy}u_y)_y + \tau(xu_x + yu_y) = f,$$

where we also have variable diffusion coefficients. The corresponding computational results are shown in Table 5.7 and Fig. 5.4. We see that while SSOR does not lead to convergence at all, ADI preconditioning still works quite well.

6. Conclusions. In this paper, we have studied, both analytically and numerically, the performance of ADI preconditioning for finite difference discretizations of elliptic boundary value problems in two dimensions. One of the advantages of this approach is that it can be easily parallelized on a variety of architectures. For separable, not necessarily selfadjoint, problems, we proved that the number of iterations required to reduce the norm of the residual by a certain factor is proportional to $h^{-1/2}$ if we use ADI(1) preconditioning and is proportional to $h^{-1/4}$ for ADI(2). In our numerical experiments we compared these ADI preconditioners

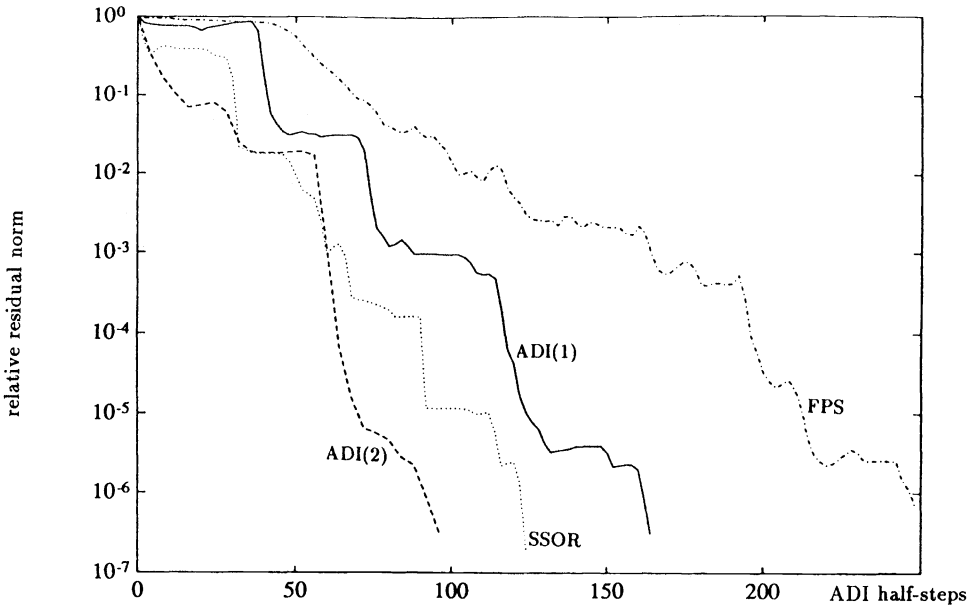


FIG. 5.3. *Example* 3 ($h = 1/64$): TFQMR, *different preconditioners.*

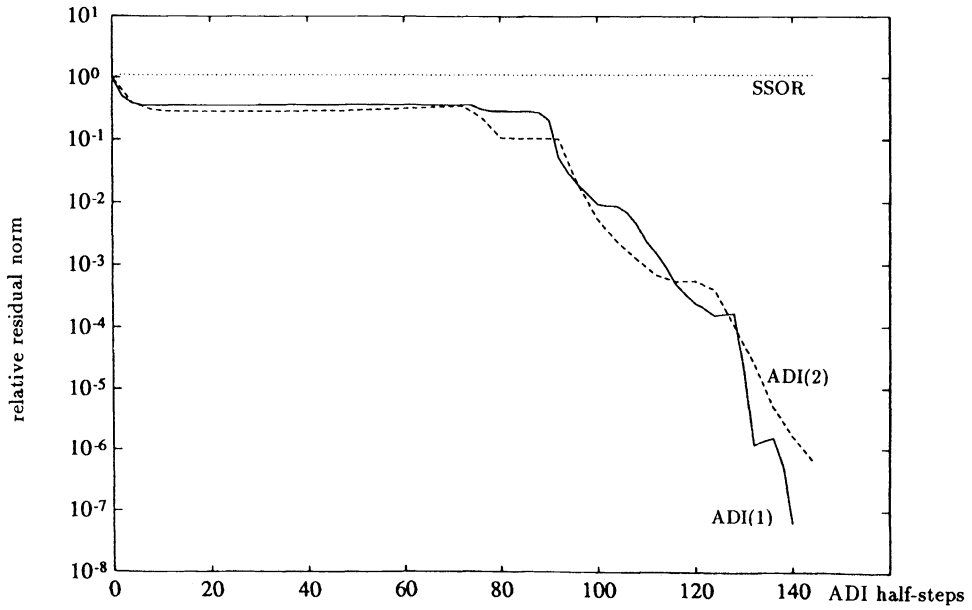


FIG. 5.4. *Example* 4 ($h = 1/64$): TFQMR, *different preconditioners.*

with other approaches for a number of different iterative methods. These numerical results illustrate the analysis which implies that, at least for small h, ADI(2) should be preferred to ADI(1). Moreover, ADI frequently performs better than preconditioning by SSOR or an FPS. This is particularly true for SSOR for small h or for problems where the underlying matrix has large imaginary eigenvalue components.

TABLE 5.7
TFQMR *iteration counts for Example* 4, $\tau = 200$.

h	No prec.	ADI(1)	ADI(2)	SSOR
1/16	419	122	98	—
1/32	—	80	40	—
1/64	—	69	36	—
1/128	—	74	38	—

Acknowledgments. I would like to thank Marlis Hochbruck and Torgeir Rusten for many stimulating discussions and for the careful reading of this manuscript. Moreover, I am thankful to Paul Saylor and one of the referees for helpful suggestions. I am also grateful to Gene Golub for comments and for providing the pleasant atmosphere at Stanford University where the idea for this paper was born.

REFERENCES

[1] R. M. BEAM AND R. F. WARMING, *Alternating direction implicit methods for parabolic equations with a mixed derivative*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 131–159.

[2] ———, *Implicit Numerical Methods for the Compressible Navier-Stokes and Euler Equations*, Lecture Notes Computational Fluid Dynamics, von Karman Institute for Fluid Dynamics, Rhode-Saint-Génese, Belgium, 1982.

[3] R. C. Y. CHIN, T. A. MANTEUFFEL, AND J. DE PILLIS, *ADI as a preconditioning for solving the convection-diffusion equation*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 281–299.

[4] P. CONCUS AND G. H. GOLUB, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, SIAM J. Numer. Anal., 10 (1973), pp. 1103–1120.

[5] J. DOUGLAS, *Alternating direction methods for three space variables*, Numer. Math., 4 (1962), pp. 41–63.

[6] J. DOUGLAS, R. DURÁN, AND P. PIETRA, *Alternating-direction iteration for mixed finite element methods*, in Computing Methods in Applied Sciences and Engineering, VII, R. Glowinski and J. L. Lions, eds., Elsevier, North-Holland, 1986, pp. 181–196.

[7] E. G. D'YAKONOV, *An iteration method for solving systems of finite difference equations*, Dokl. Akad. Nauk SSSR, 24 (1961), pp. 271–274.

[8] M. EIERMANN, *On semiiterative methods generated by Faber polynomials*, Numer. Math., 56 (1989), pp. 139–156.

[9] ———, *Fields of values and iterative methods*, Linear Algebra Appl., 180 (1993), pp. 167–198.

[10] H. ELMAN, *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, Yale University, New Haven, CT, 1982.

[11] H. C. ELMAN AND G. H. GOLUB, *Line iterative methods for cyclically reduced discrete convection-diffusion problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 339–363.

[12] H. C. ELMAN AND M. H. SCHULTZ, *Preconditioning by fast direct methods for nonself-adjoint nonseparable elliptic equations*, SIAM J. Numer. Anal., 23 (1986), pp. 44–57.

[13] R. W. FREUND, *Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations*, in Numerical Methods in Approximation Theory, Vol. 9, D. Braess and L. L. Schumaker, eds., Birkhäuser, Basel, Boston, Berlin, 1992, pp. 77–95.

[14] ———, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.

[15] 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.

[16] R. GLOWINSKI, *Numerical Methods for Nonlinear Variational Problems*, Springer-Verlag, New York, Berlin, Heidelberg, Tokyo, 1984.

[17] M. HOCHBRUCK ANDA G. STARKE, *Preconditioning Lyapunov matrix equations*, IPS Research Report 92-17, ETH-Zentrum, Zürich, October 1992; SIAM J. Matrix Anal. Appl., submitted.

[18] H. JIANG AND Y. S. WONG, *A parallel alternating direction implicit preconditioned method*, J. Comput. Appl. Math., 36 (1991), pp. 209–226.

[19] S. L. JOHNSSON, Y. SAAD, AND M. H. SCHULTZ, *Alternating direction methods on multiprocessors*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 686–700.
[20] T. A. MANTEUFFEL, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183–208.
[21] M. MARCUS AND H. MINC, *A Survey of Matrix Theory and Matrix Inequalities*, Allyn and Bacon, Boston, 1964.
[22] 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.
[23] D. W. PEACEMAN AND H. H. RACHFORD, *The numerical solution of parabolic and elliptic differential equations*, J. Soc. Indust. Appl. Math., 3 (1955), pp. 28–41.
[24] H. S. PRICE AND R. S. VARGA, *Recent numerical experiments comparing successive overrelaxation iterative methods with implicit alternating direction methods*, Gulf Research and Development Co. Report, Harmarville, PA, 1962.
[25] 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.
[26] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
[27] G. STARKE, *Optimal* ADI *parameters for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 28 (1991), pp. 1431–1445.
[28] G. STARKE AND R. S. VARGA, *A hybrid Arnoldi–Faber method for nonsymmetric systems of linear equations*, Numer. Math., 64 (1993), pp. 213–240.
[29] P. N. SWARZTRAUBER, *A direct method for the discrete solution of separable elliptic equations*, SIAM J. Numer. Anal., 11 (1974), pp. 1136–1150.
[30] Thinking Machines Corporation, *The Connection Machine CM-5 Technical Summary*, Cambridge, MA, Oct. 1991.
[31] R. S. VARGA, *Matrix Iterative Analysis*, Prentice Hall, New York, 1962.
[32] E. L. WACHSPRESS, *Iterative Solution of Elliptic Systems*, Prentice Hall, New York, 1966.
[33] ———, *Generalized ADI preconditioning*, Comp. Math. Appl., 10 (1984), pp. 457–461.
[34] ———, *Iterative solution of the Lyapunov matrix equation*, Appl. Math. Lett., 1 (1988), pp. 87–90.
[35] O. B. WIDLUND, *On the rate of convergence of an alternating direction implicit method in a noncommutative case*, Math. Comp., 20 (1966), pp. 500–515.
[36] ———, *On the use of fast methods for separable finite difference equations for the solution of general elliptic problems*, in Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972, pp. 121–131.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 385–407, March 1994

010

SEMICIRCULANT PRECONDITIONERS FOR FIRST-ORDER PARTIAL DIFFERENTIAL EQUATIONS*

SVERKER HOLMGREN† AND KURT OTTO†

Abstract. This paper considers solving time-independent systems of first-order partial differential equations (PDEs) in two space dimensions using a conjugate gradient (CG)-like iterative method. The systems of equations are preconditioned using semicirculant preconditioners. Analytical formulas for the eigenvalues and the eigenvectors are derived for a scalar model problem with constant coefficients. The main problems in constructing and analyzing the numerical methods are caused by the numerical boundary conditions required at the outflow boundaries. It is proved that, when the grid ratio is less than one, the spectrum asymptotically becomes two finite curve segments that are independent of the number of gridpoints. The same type of result for a time-dependent problem has previously been established. For the restarted generalized minimal residual (GMRES) iteration, a slight reduction of the grid ratio from one substantially improves the convergence rate. This is also predicted by an asymptotic analysis of the eigenvalues.

Key words. first-order PDE, finite difference discretization, preconditioners, spectra, convergence analysis

AMS subject classifications. 65F10, 65N22

1. Introduction.

In this paper we consider solving first-order PDEs in two space dimensions. The systems of equations arising are solved using a CG-like iterative method combined with semicirculant preconditioners. Eventually we intend to use the methods presented here in the process of solving the Euler and Navier–Stokes equations.

We first give a brief description of semicirculant preconditioners. Then we present scalar model problems with constant coefficients. We mainly consider a steady-state hyperbolic equation, but results for a related time-dependent problem are included for comparison. For the model problems we give analytical formulas for the eigenvalues of the preconditioned coefficient matrices. The eigenvectors of the steady-state problem are also studied.

Using a grid in which the number of gridpoints in one space direction is greater than in the other, it is proved [Otto92] that the spectrum for the time-dependent model problem asymptotically resides on two finite curve segments in the complex plane. The curves are well separated from zero and independent of the quotient κ between the time- and space-steps. These results should be compared to those presented in [HoOtto91]. There Holmgren and Otto analyze a similar model problem, but with periodic boundary conditions in one space direction and the same number of gridpoints in both directions. The results for that problem imply that the spectrum is independent of the number of gridpoints, but that $\max_i |\lambda_i| / \min_i |\lambda_i| = \mathcal{O}(\sqrt{\kappa})$.

Generalizing the results in [Otto92], we are able to prove that asymptotically the spectrum for the steady-state problem resides on the same curve segments as for the time-dependent problem, provided a weak artificial viscosity is added in one space direction. Also, the grid ratio must be less than one to guarantee a bounded spectrum. This is the main topic of this paper. The theoretical arguments are verified by numerical experiments.

2. Notation.

We solve

$$M^{-1}Bu = M^{-1}b\,, \tag{1}$$

where M is the preconditioner matrix. The matrix B arises from a discretization of a system of n_c PDEs in two space dimensions using a standard five-point operator. The solution is

*Received by the editors May 18, 1992; accepted for publication (in revised form) April 23, 1993. 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, kurt@tdb.uu.se).

unknown in $m_1 \times m_2$ gridpoints, hence u has $n = n_c m_1 m_2$ components. The only restriction on the difference approximation at the boundaries is that it must not be wider than the five-point operator used in the interior. The matrix B is assumed to have the following structure:

$$B = \mathrm{trid}_{k,m_2}(B_{k,-1}, B_{k,0}, B_{k,1}), \tag{2}$$

where

$$\begin{aligned} B_{k,-1} &= \mathrm{diag}_{j,m_1}(v_{j,k,-2}), \\ B_{k,0} &= \mathrm{trid}_{j,m_1}(v_{j,k,-1}, v_{j,k,0}, v_{j,k,1}), \\ B_{k,1} &= \mathrm{diag}_{j,m_1}(v_{j,k,2}). \end{aligned}$$

Here $v_{j,k,r}$, $r = -2, \ldots, 2$, are $n_c \times n_c$ matrices. Hence, $\sim 5n_c n$ memory cells are required to store B, and $\sim 10n_c n$ arithmetic operations are required to perform the matrix-vector multiplication $y = Bx$. Above, we have used the notation

$$\mathrm{diag}_{j,m}(\beta_j) = \begin{pmatrix} \beta_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \beta_m \end{pmatrix}, \quad \mathrm{trid}_{j,m}(\alpha_j, \beta_j, \gamma_j) = \begin{pmatrix} \beta_1 & \gamma_1 & & \\ \alpha_2 & \ddots & \ddots & \\ & \ddots & \ddots & \gamma_{m-1} \\ & & \alpha_m & \beta_m \end{pmatrix}.$$

We will also denote circulant tridiagonal and periodic tridiagonal matrices by

$$\mathrm{ctrid}_m(\alpha, \beta, \gamma) = \begin{pmatrix} \beta & \gamma & & \alpha \\ \alpha & \ddots & \ddots & \\ & \ddots & \ddots & \gamma \\ \gamma & & \alpha & \beta \end{pmatrix}, \quad \mathrm{ptrid}_{j,m}(\alpha_j, \beta_j, \gamma_j) = \begin{pmatrix} \beta_1 & \gamma_1 & & \alpha_1 \\ \alpha_2 & \ddots & \ddots & \\ & \ddots & \ddots & \gamma_{m-1} \\ \gamma_m & & \alpha_m & \beta_m \end{pmatrix}.$$

The Kronecker product, denoted by $\otimes$, will be used extensively. The mth-order Fourier matrix F_m is given by

$$[F_m]_{j,k} = m^{-1/2} \cdot \omega_m^{(j-1)(k-1)}, \qquad j, k = 1, \ldots, m,$$

where $\omega_m \equiv \exp(i2\pi/m)$.

3. Semicirculant preconditioners. For a more complete discussion on semicirculant preconditioners and some numerical results, see [HoOtto91], [Holm93], and [Otto93]. Circulant preconditioners are discussed in [TChan88], [RChSt89], [RChan89], [RChJin91], and [TChOl92]. A semicirculant preconditioner M for B is defined by

$$M = \mathrm{ptrid}_{k,m_2}(M_{k,-1}, M_{k,0}, M_{k,1}), \tag{3}$$

where

$$M_{k,-1} = I_{m_1} \otimes \rho_{k,-2}, \quad M_{k,0} = \mathrm{ctrid}_{m_1}(\rho_{k,-1}, \rho_{k,0}, \rho_{k,1}), \quad M_{k,1} = I_{m_1} \otimes \rho_{k,2}.$$

Here $\rho_{k,r}$ are $n_c \times n_c$ matrices. When $\rho_{1,-2}$ and $\rho_{m_2,2}$ are zero, M is nonperiodic block-tridiagonal just like B. However, the diagonal blocks in M are circulant matrices, whereas the corresponding blocks in B are tridiagonal. Notice that the preconditioning matrix M is almost as sparse as B, and it is completely described by the $5m_2$ $n_c \times n_c$ matrices $\rho_{k,r}$. Different choices of these parameters result in different preconditioners. In this paper we use

$$\rho_{k,r} = \frac{1}{m_1} \sum_{j=1}^{m_1} \tilde{v}_{j,k,r}, \quad k = 1, \ldots, m_2, \quad r = -2, \ldots, 2. \tag{4}$$

The matrix entries $\tilde{v}_{j,k,r}$, $j = 1, \ldots, m_1$, $k = 1, \ldots, m_2$, $r = -2, \ldots, 2$, form the matrix $\tilde{B}$ corresponding to the difference approximation of the original problem, but with periodic boundary conditions in the x_1-direction. Most of the entries $\tilde{v}_{j,k,r}$ will be equal to the entries $v_{j,k,r}$ in the matrix B. Especially, $\rho_{1,-2}$ and $\rho_{m_2,2}$ become zero, and consequently M becomes block-tridiagonal. The choice of parameters for circulant and semicirculant preconditioners is discussed in [TChan88], [Tyr92], and [HoOtto91].

The reason for introducing circulant matrices in the preconditioner is that such matrices are diagonalized by using discrete Fourier transforms, which are performed by the fast Fourier transform (FFT) algorithm. The derivation of the preconditioner solve is based on the observation that M^{-1} may be written as

$$M^{-1} = (I_{m_2} \otimes F_{m_1} \otimes I_{n_c}) \big(\text{ptrid}_{k,m_2}(M_{k,-1}, \Lambda_k, M_{k,1})\big)^{-1} (I_{m_2} \otimes F_{m_1}^* \otimes I_{n_c}), \tag{5}$$

where $\Lambda_k = \text{diag}_{j,m_1}(\Lambda_{j,k})$ and $\Lambda_{j,k} = \rho_{k,0} + \omega_{m_1}^{(j-1)} \rho_{k,1} + \omega_{m_1}^{-(j-1)} \rho_{k,-1}$. Details are given in [HoOtto91]. In [Holm93] some savings in arithmetics, generalizations, and parallel implementations are discussed. An algorithmic formulation of the preconditioner solve follows:

PRECONDITIONER SOLVE

for $k = 1$ to $n_c m_2$ do (in parallel):
    Perform an FFT of length $m_1/2$.
endfor
for $j = 1$ to $m_1/2 + 1$ do (in parallel):
    Solve a (possibly periodic) block-tridiagonal system of equations
    with blocksize $n_c \times n_c$ and $n_c m_2$ unknowns.
endfor
for $k = 1$ to $n_c m_2$ do (in parallel):
    Perform an FFT of length $m_1/2$.
endfor

For the implementation used in this paper, $\sim 6 n_c n$ memory cells are required to store the factorized reduced systems and for workspace. To form the preconditioner and to factorize the reduced systems $\sim (32 n_c^2 - 16 n_c + 3) n$ arithmetic operations are required. Each preconditioner solve requires $\sim (5 \log_2 m_1 + 24 n_c + 2) n$ arithmetic operations. Both the memory requirement and the arithmetic complexity is increased if m_1 is not a power of two, since the discrete Fourier transforms are then computed according to the *fractional* Fourier transform algorithm [BaSw91]. The semicirculant framework can be extended to three space dimensions; see [Holm93] and [Otto93]. The three-dimensional preconditioner solve is based on FFT methods in two dimensions and the solution of block-tridiagonal systems in the third dimension.

In [Otto93] and [Holm93] standard incomplete LU (ILU) and block ILU preconditioners are considered for the type of problems presented in the next section. Analysis of the spectra for the preconditioned systems shows [Otto93] that no significant improvement is obtained compared to the unpreconditioned system. Furthermore, some preconditioners exhibit stability problems [Holm93]. For large, nondiagonally dominant problems the convergence properties are poor. The conclusion is that ILU and block ILU preconditioners are not beneficial for the problems of interest here.

4. The model problems. We present a hyperbolic model problem in both a time-dependent and a steady-state setting. The time-dependent problem is analyzed in [Otto92]. We present it here since it is instructive to compare the coefficient matrices and the results for the two problem settings.

The time-dependent problem. We consider the scalar ($n_c = 1$) two-dimensional problem

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} = g, \tag{6}$$

on the unit square for $t > 0$. Here $u(0, x_2, t)$, $u(x_1, 0, t)$, $u(x_1, x_2, 0)$, and g are chosen so that the analytical solution is known. The time discretization is performed using the second-order accurate trapezoidal rule. The space discretization is performed on a uniform grid with $(m_1+1) \times (m_2+1)$ gridpoints, where the space-steps are given by $h_d = 1/m_d$, $d = 1, 2$. Let $u_{j,k}$ denote the approximate solution at the point (jh_1, kh_2), $j = 0, \ldots, m_1$, $k = 0, \ldots, m_2$. Observe that $u_{0,k}$ for $k = 0, \ldots, m_2$ and $u_{j,0}$ for $j = 0, \ldots, m_1$ are given directly by the boundary conditions for the PDE problem. This means that we have to solve for $m_1 \times m_2$ unknowns in each time-step. The spatial derivatives in (6) 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. Now define $\kappa_1 \equiv \Delta t/h_1$ and $\kappa_2 \equiv \Delta t/h_2$, where Δt is the time-step. The solution vector is defined as $u^N = (u^N_{1,1}\ u^N_{2,1}\ \ldots\ u^N_{m_1,1}\ u^N_{1,2}\ \ldots\ u^N_{m_1,m_2})^T$. Introducing the discretizations in (6) yields the following system of equations for the unknowns at time level $N+1$:

$$(I_{m_2} \otimes B_1 + \kappa_2 B_2 \otimes I_{m_1}) u^{N+1} = b. \tag{7}$$

Here b contains known quantities and

$$B_1 = \begin{pmatrix} 4 & \kappa_1 & & & \\ -\kappa_1 & 4 & \kappa_1 & & \\ & \ddots & \ddots & \ddots & \\ & & -\kappa_1 & 4 & \kappa_1 \\ & & & -2\kappa_1 & 4+2\kappa_1 \end{pmatrix},$$

$$B_2 = \begin{pmatrix} 0 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -2 & 2 \end{pmatrix}.$$

The steady-state problem. Here we consider the steady-state version of (6):

$$\frac{\partial u}{\partial x_1} + \frac{\partial u}{\partial x_2} = g. \tag{8}$$

The boundary conditions are of the same form as for the time-dependent problem. The boundary data and g are again chosen so that the analytical solution is known. The space discretization is performed on the same type of grid as for the time-dependent problem, 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

$$D_{0,x_1} + D_{0,x_2} - \frac{\gamma}{2} h_1^{2-\alpha} D_{+,x_1} D_{-,x_1}, \tag{9}$$

where $D_{+,x_1} u_{j,k} = h_1^{-1}(u_{j+1,k} - u_{j,k})$, $D_{-,x_1} u_{j,k} = D_{+,x_1} u_{j-1,k}$, and finally $D_{0,x_1} = \frac{1}{2}(D_{+,x_1} + D_{-,x_1})$. The constants α and γ are chosen so that $\alpha \in [0, 1)$ and $\gamma \in (0, 1)$. The order of accuracy for this operator is $2 - \alpha$ in the x_1-direction and 2 in the x_2-direction.

Adding artificial viscosity is customary when using centered differences for flow problems. Normally, a stronger artificial viscosity like the difference operator $f(\nabla u)\gamma_1 h_1 D_{+,x_1} D_{-,x_1} + \gamma_2 h_1^3 (D_{+,x_1} D_{-,x_1})^2$ is added. The function f is large when the gradient of u is large, i.e., in a shock, and small otherwise. It would be interesting to perform the analysis using an artificial viscosity of the form $\gamma_2 h_1^3 (D_{+,x_1} D_{-,x_1})^2$, which preserves second-order accuracy. However, we have not yet pursued this approach. Artificial viscosity could also be added in the x_2-direction. The solution method presented here would still be defined, and reasonably the performance should be enhanced.

For the numerical boundary conditions we again use one-sided differences. The difference operators at the outflow boundaries are given by

$$\begin{aligned} D_{-,x_1} + D_{0,x_2}, &\quad j = m_1, \quad k = 1, \ldots, m_2 - 1, \\ D_{-,x_1} + D_{-,x_2}, &\quad j = m_1, \quad k = m_2, \\ D_{0,x_1} + D_{-,x_2} - \frac{\gamma}{2} h_1^{2-\alpha} D_{+,x_1} D_{-,x_1}, &\quad k = m_2, \quad j = 1, \ldots, m_1 - 1. \end{aligned}$$

Define the grid ratio β and δ by

$$\beta \equiv h_1/h_2\,, \qquad \delta \equiv \gamma h_1^{1-\alpha}.$$

We then have the following system of equations:

$$Bu \equiv (I_{m_2} \otimes \hat{B}_1 + \beta B_2 \otimes I_{m_1})u = b. \tag{10}$$

Here b again contains known quantities and

$$\hat{B}_1 = \begin{pmatrix} 2\delta & 1-\delta & & & \\ -1-\delta & 2\delta & 1-\delta & & \\ & \ddots & \ddots & \ddots & \\ & & -1-\delta & 2\delta & 1-\delta \\ & & & -2 & 2 \end{pmatrix}.$$

The difference between the two problem settings lies in the difference between B_1 and $\hat{B}_1$. The study of the steady-state problem was initiated by an examination of the quotient between the diagonal and off-diagonal entries in these matrices. When the number of gridpoints is large, the steady-state problem is closely related to a time-dependent problem with $\kappa_1 = \frac{2}{\delta} = 2\gamma^{-1} h_1^{\alpha-1}$. Asymptotically the matrix $\hat{B}_1$ approaches the "true" wave propagation matrix. This is not the case for the standard form of artificial viscosity with $\alpha = 1$, since the main diagonal does not vanish.

5. Two lemmas. In this section we state two fundamental lemmas. The first lemma concerns the location of the eigenvalues of the matrix B_2. 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 1. *The eigenvalues $\lambda_{k,2}$ of B_2 satisfy*:

(i) $\lambda_{k,2} \neq \lambda_{j,2}\,, \quad k \neq j.$

(ii) $\lambda_{k,2} \in \mathcal{E}^+(4m_2^{-3/4}, 2 + 4m_2^{-3/2})\,, \quad k = 1, \ldots, m_2.$

(iii) $\lambda_{k,2} > \ln(m_2)/m_2$ *when* $\Im m(\lambda_{k,2}) = 0.$

The proof is based on difference equation theory and is available in [Otto92].

Lemma 2 concerns the evaluation of certain sums in the proof of Theorem 2.

LEMMA 2. *Assume that $d \geq 1$. Let ζ denote a complex number satisfying*

$$\zeta \notin \mathcal{E}\left(\frac{d-1}{2}, \frac{d+1}{2}\right) \quad \text{and} \quad \Re e(\zeta) > 0\,.$$

Then

$$\sigma_p \equiv m_1^{-1} \sum_{j=1}^{m_1} \frac{\omega_{m_1}^{p(j-1)}}{2\zeta + \omega_{m_1}^{(j-1)} - d\omega_{m_1}^{-(j-1)}} = s_p + r_p\,, \qquad p = -m_1 + 1, \ldots, m_1 - 1,$$

where

$$s_p = \frac{(-dz^{-1})^p}{z + dz^{-1}}, \quad p < 0,$$

$$s_p = \frac{z^p}{z + dz^{-1}}, \quad p \geq 0,$$

$$r_p = \frac{z^p}{z + dz^{-1}} \cdot \frac{z^{m_1}}{1 - z^{m_1}} + \frac{(-dz^{-1})^p}{z + dz^{-1}} \cdot \frac{(-d^{-1}z)^{m_1}}{1 - (-d^{-1}z)^{m_1}},$$

and

$$z = -\zeta + \sqrt{d + \zeta^2}.$$

The proof is available in [Otto93]. It is based on the Poisson summation formula, residue theory, and geometric series.

6. The steady-state problem. A semicirculant preconditioner, with parameters chosen according to (4), for the coefficient matrix in (10) is given by

$$M = I_{m_2} \otimes \hat{C}_1 + \beta B_2 \otimes I_{m_1}, \tag{11}$$

where

$$\hat{C}_1 = \mathrm{ctrid}_{m_1}(-1 - \delta, 2\delta, 1 - \delta).$$

Theorem 1 establishes the existence of M^{-1} for all finite grids.

THEOREM 1. *The matrix M is diagonalizable and nonsingular.*

Proof. Since $\hat{C}_1$ is a circulant matrix we have

$$F_{m_1}^* \hat{C}_1 F_{m_1} = \Lambda_1 = \mathrm{diag}_{j,m_1}(\lambda_{j,1}), \qquad \lambda_{j,1} = 2\delta + (1 - \delta)\omega_{m_1}^{(j-1)} - (1 + \delta)\omega_{m_1}^{-(j-1)}.$$

From Lemma 1 we conclude that there exists a nonsingular matrix V_2 diagonalizing B_2, i.e.,

$$V_2^{-1} B_2 V_2 = \Lambda_2 = \mathrm{diag}_{k,m_2}(\lambda_{k,2}).$$

This implies

$$\begin{aligned}(V_2^{-1} \otimes F_{m_1}^*) M (V_2 \otimes F_{m_1}) &= (V_2^{-1} \otimes F_{m_1}^*)(I_{m_2} \otimes \hat{C}_1 + \beta B_2 \otimes I_{m_1})(V_2 \otimes F_{m_1}) \\ &= I_{m_2} \otimes F_{m_1}^* \hat{C}_1 F_{m_1} + \beta V_2^{-1} B_2 V_2 \otimes I_{m_1} \\ &= I_{m_2} \otimes \Lambda_1 + \beta \Lambda_2 \otimes I_{m_1} \equiv \Lambda.\end{aligned}$$

Thus, M is diagonalizable. The eigenvalues of M are the diagonal entries of Λ, i.e., $\lambda_{j,1} + \beta\lambda_{k,2}$, $k = 1, \ldots, m_2$, $j = 1, \ldots, m_1$. We prove by contradiction that these eigenvalues are nonzero. Assume that $\lambda_{j,1} + \beta\lambda_{k,2} = 0$ for some j and k. Then we have

$$\begin{aligned}0 &= 2\delta + (1 - \delta)\omega_{m_1}^{(j-1)} - (1 + \delta)\omega_{m_1}^{-(j-1)} + \beta\lambda_{k,2} \\ &= 2\delta\big(1 - \cos((j-1)2\pi/m_1)\big) + \beta\Re e(\lambda_{k,2}) \\ &\quad + i\big(2\sin((j-1)2\pi/m_1) + \beta\Im m(\lambda_{k,2})\big).\end{aligned}$$

Now $\delta, \beta > 0$ and $\Re e(\lambda_{k,2}) \geq 0$ according to Lemma 1. Then the only possibility is $j = 1$, $\Re e(\lambda_{k,2}) = 0$, and $\Im m(\lambda_{k,2}) = 0$. But $\lambda_{k,2} = 0$ contradicts Lemma 1, and consequently the assumption is false. □

The convergence analysis in §8, for GMRES applied to (1), is based on the spectral decomposition of $M^{-1}B$. For the rest of this section, we therefore examine the eigenvalues and the eigenvectors of $M^{-1}B$. Now define the error matrix $E \equiv B - M$. Observe that $E = I_{m_2} \otimes E_1$, where

$$E_1 = \hat{B}_1 - \hat{C}_1 = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 & 1+\delta \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 \\ -1+\delta & 0 & \cdots & 0 & -1+\delta & 2-2\delta \end{pmatrix}. \tag{12}$$

We obtain

$$M^{-1}B = M^{-1}(M+E) = I + M^{-1}E. \tag{13}$$

Theorem 2 provides analytical formulas relating the eigenvalues of $M^{-1}E$ to the eigenvalues of B_2.

THEOREM 2. *Let $\lambda_{k,2}$, $k = 1, \ldots, m_2$, denote the eigenvalues of B_2. Define ζ_k and d by*

$$\zeta_k \equiv \frac{\delta}{1-\delta} + \frac{\beta\lambda_{k,2}}{2(1-\delta)} \quad \text{and} \quad d \equiv \frac{1+\delta}{1-\delta}.$$

Then the matrix $M^{-1}E$ has the eigenvalue zero with multiplicity at least $m_2(m_1 - 2)$, and $2m_2$ eigenvalues given by

$$\lambda^{(SC)}_{(1,m_1),k} = \frac{2 - (2-d^{-1})z_k + R_1(z_k) \pm \sqrt{-(4d^{-1} - d^{-2})z_k^2 - 4(2-d^{-1})z_k - 4(d-1) + R_2(z_k)}}{2(z_k + dz_k^{-1})},$$

$$k = 1, \ldots, m_2,$$

where

$$\begin{aligned}
R_1(z_k) &= \big((d-1)z_k^{-1} + 2 - z_k\big)\frac{z_k^{m_1}}{1-z_k^{m_1}} + \big((d^{-1}-1)z_k + 2 + dz_k^{-1}\big)\frac{(-d^{-1}z_k)^{m_1}}{1-(-d^{-1}z_k)^{m_1}}, \\
R_2(z_k) &= \big(-6d^{-1}z_k^2 - (12-4d^{-1})z_k - 8d + 6 - 2d^{-1} + (4d-4)z_k^{-1} - 4dz_k^{-2}\big)\frac{z_k^{m_1}}{1-z_k^{m_1}} \\
&\quad + \big(-(6d^{-1} - 2d^{-2})z_k^2 - (12 - 8d^{-1})z_k - 8d + 10 + 4dz_k^{-1}\big)\frac{(-d^{-1}z_k)^{m_1}}{1-(-d^{-1}z_k)^{m_1}} \\
&\quad + \big(z_k^2 - 4z_k - 2d + 6 + (4d-4)z_k^{-1} + (d-1)^2 z_k^{-2}\big)\Big(\frac{z_k^{m_1}}{1-z_k^{m_1}}\Big)^2 \\
&\quad + \big((d^{-1}-1)^2 z_k^2 + (4d^{-1} - 4)z_k - 2d + 6 + 4dz_k^{-1} + d^2 z_k^{-2}\big)\Big(\frac{(-d^{-1}z_k)^{m_1}}{1-(-d^{-1}z_k)^{m_1}}\Big)^2 \\
&\quad + \big(-(2 + 6d^{-1})z_k^2 - (8 - 4d^{-1})z_k - 12d + 4 - 2d^{-1} \\
&\qquad + (8d-4)z_k^{-1} - (2d^2 + 6d)z_k^{-2}\big)\frac{z_k^{m_1}}{1-z_k^{m_1}} \cdot \frac{(-d^{-1}z_k)^{m_1}}{1-(-d^{-1}z_k)^{m_1}},
\end{aligned}$$

and finally

$$z_k = -\zeta_k + \sqrt{d + \zeta_k^2}.$$

Proof. First we construct a sparse block-diagonal matrix having the same eigenvalues as $M^{-1}E$. Using the notation from Theorem 1 we introduce

$$D = (I_{m_2} \otimes \Lambda_1 + \beta \Lambda_2 \otimes I_{m_1})^{-1} = \mathrm{diag}_{k,m_2}(D_k),$$

where

$$D_k = \mathrm{diag}_{j,m_1}(d_{j,k}) = \mathrm{diag}_{j,m_1}\left((2\delta + (1-\delta)\omega_{m_1}^{(j-1)} - (1+\delta)\omega_{m_1}^{-(j-1)} + \beta\lambda_{k,2})^{-1}\right).$$

Employing the transformation of M from Theorem 1, the spectral decomposition of M^{-1} becomes

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

We now consider the matrix T given by

$$\begin{aligned} T &= (V_2 \otimes I_{m_1})^{-1} M^{-1} E (V_2 \otimes I_{m_1}) \\ &= (V_2^{-1} \otimes I_{m_1})(V_2 \otimes F_{m_1}) D (V_2^{-1} \otimes F_{m_1}^*)(I_{m_2} \otimes E_1)(V_2 \otimes I_{m_1}) \\ &= (I_{m_2} \otimes F_{m_1}) \mathrm{diag}_{k,m_2}(D_k)(I_{m_2} \otimes F_{m_1}^* E_1) = \mathrm{diag}_{k,m_2}(T_k), \end{aligned}$$

where $T_k = F_{m_1} D_k F_{m_1}^* E_1$. Since $M^{-1}E$ and T are similar, and T is block-diagonal, the eigenvalues of $M^{-1}E$ equal the eigenvalues of T_k, $k = 1, \ldots, m_2$.

Due to the structure of E_1 displayed in (12), we see that T_k has only three nonzero columns $t_{p,1}$, t_{p,m_1-1}, t_{p,m_1}, $p = 1, \ldots, m_1$. Performing the matrix multiplication and using the periodicity $\omega_{m_1}^{(p \pm m_1)} = \omega_{m_1}^{p}$, we obtain

$$\begin{aligned} t_{p,1} &= m_1^{-1} \sum_{j=1}^{m_1} \omega_{m_1}^{p(j-1)} (-1+\delta) d_{j,k} \\ &= -m_1^{-1} \sum_{j=1}^{m_1} \frac{\omega_{m_1}^{p(j-1)}}{\frac{2\delta+\beta\lambda_{k,2}}{1-\delta} + \omega_{m_1}^{(j-1)} - \frac{1+\delta}{1-\delta}\omega_{m_1}^{-(j-1)}} = -\sigma_p, \\ t_{p,m_1-1} &= t_{p,1} = -\sigma_p, \\ t_{p,m_1} &= m_1^{-1} \sum_{j=1}^{m_1} \left(\omega_{m_1}^{(p-1)(j-1)}(1+\delta) + \omega_{m_1}^{p(j-1)}(2-2\delta)\right) d_{j,k} \\ &= m_1^{-1} \sum_{j=1}^{m_1} \frac{d\omega_{m_1}^{(p-1)(j-1)} + 2\omega_{m_1}^{p(j-1)}}{\frac{2\delta+\beta\lambda_{k,2}}{1-\delta} + \omega_{m_1}^{(j-1)} - \frac{1+\delta}{1-\delta}\omega_{m_1}^{-(j-1)}} = d\sigma_{p-1} + 2\sigma_p, \end{aligned}$$

where

$$\sigma_p = m_1^{-1} \sum_{j=1}^{m_1} \frac{\omega_{m_1}^{p(j-1)}}{2\zeta_k + \omega_{m_1}^{(j-1)} - d\omega_{m_1}^{-(j-1)}}.$$

Using the periodicity $\sigma_{p \pm m_1} = \sigma_p$, the characteristic equation for T_k becomes

$$0 = \det(\lambda I - T_k) = \lambda^{m_1-3}\begin{vmatrix} \lambda - t_{1,1} & -t_{1,m_1-1} & -t_{1,m_1} \\ -t_{m_1-1,1} & \lambda - t_{m_1-1,m_1-1} & -t_{m_1-1,m_1} \\ -t_{m_1,1} & -t_{m_1,m_1-1} & \lambda - t_{m_1,m_1} \end{vmatrix}$$

$$= \lambda^{m_1-3}\begin{vmatrix} \lambda + \sigma_1 & \sigma_1 & -d\sigma_0 - 2\sigma_1 \\ \sigma_{-1} & \lambda + \sigma_{-1} & -d\sigma_{-2} - 2\sigma_{-1} \\ \sigma_0 & \sigma_0 & \lambda - d\sigma_{-1} - 2\sigma_0 \end{vmatrix}$$

$$= \lambda^{m_1-2}\big(\lambda^2 - ((d-1)\sigma_{-1} + 2\sigma_0 - \sigma_1)\lambda + d(\sigma_0^2 - \sigma_{-1}^2 - \sigma_1\sigma_{-1} + \sigma_0\sigma_{-2})\big).$$

Thus, T_k has at most two nonzero eigenvalues

$$\lambda^{(SC)}_{(1,m_1),k} = \frac{(d-1)}{2}\sigma_{-1} + \sigma_0 - \frac{\sigma_1}{2} \pm$$
$$\sqrt{\frac{\sigma_1^2}{4} - (d-1)\sigma_0^2 + \frac{(d+1)^2}{4}\sigma_{-1}^2 + \frac{(d+1)}{2}\sigma_1\sigma_{-1} - \sigma_1\sigma_0 - d\sigma_0\sigma_{-2} + (d-1)\sigma_0\sigma_{-1}}.$$

Now

$$\zeta_k = \frac{\delta}{1-\delta} + \frac{\beta\lambda_{k,2}}{2(1-\delta)} = \frac{d-1}{2} + \frac{\beta\lambda_{k,2}}{2(1-\delta)},$$

and due to Lemma 1 we get $\Re e(\zeta_k) \geq (d-1)/2 > 0$, and $\Re e(\zeta_k) > (d-1)/2$ when $\Im m(\zeta_k) = 0$. From Lemma 2 we then obtain formulas for σ_p. Inserting those formulas into the equation above and rearranging terms, the final formulas for $\lambda^{(SC)}_{(1,m_1),k}$ follow. □

We now determine the eigenvectors of $M^{-1}B$.

THEOREM 3. *Assume that there are constants $\eta_1 \geq \eta_2 > 0$ and $\eta_3 > 0$ such that $\eta_1 \geq |\lambda^{(SC)}_{(1,m_1),k}| \geq \eta_2$ and $|\lambda^{(SC)}_{1,k} - \lambda^{(SC)}_{m_1,k}| \geq \eta_3$. Then the matrix $M^{-1}B$ has a nonsingular eigenvector matrix*

$$W_{M^{-1}B} = (V_2 \otimes I_{m_1})\mathrm{diag}_{k,m_2}(\hat{U}_k).$$

V_2 is the eigenvector matrix of B_2. The columns $\hat{u}^{(j),k}$ of $\hat{U}_k$ are given by

$$\begin{aligned} \hat{u}^{(1),k} &= u^{(1),k}/\|u^{(1),k}\|_2, \\ \hat{u}^{(j),k} &= e_j, \quad j = 2, \ldots, m_1 - 2, \\ \hat{u}^{(m_1-1),k} &= (e_1 - e_{m_1-1})/\sqrt{2}, \\ \hat{u}^{(m_1),k} &= u^{(m_1),k}/\|u^{(m_1),k}\|_2. \end{aligned}$$

The vector e_j denotes the jth canonical vector. The entries of $u^{(1,m_1),k}$ are given by

$$u_p^{(1,m_1),k} = \frac{\lambda^{(SC)}_{(1,m_1),k}(1 - (-d^{-1}z_k)^{m_1}) + 1}{(z_k + dz_k^{-1})(1 - z_k^{m_1})(1 - (-d^{-1}z_k)^{m_1})} z_k^p$$
$$+ \frac{\lambda^{(SC)}_{(1,m_1),k}(1 - z_k^{m_1}) - z_k^{m_1}}{(z_k + dz_k^{-1})(1 - z_k^{m_1})(1 - (-d^{-1}z_k)^{m_1})}(-d^{-1}z_k)^{m_1-p},$$
$$p = 1, \ldots, m_1,$$

where z_k is defined in Theorem 2.

Proof. Lemma 1 yields that V_2 is nonsingular. In Theorem 2 we derived

$$(V_2 \otimes I_{m_1})^{-1} M^{-1} E (V_2 \otimes I_{m_1}) = \mathrm{diag}_{k,m_2}(T_k),$$

where the eigenvalues of T_k are $\lambda_{1,k}^{(SC)}$, $\lambda_{m_1,k}^{(SC)}$, and zero with multiplicity $m_1 - 2$. Only columns $t^{(1)}$, $t^{(m_1-1)}$, $t^{(m_1)}$ of T_k are nonzero and $t^{(1)} = t^{(m_1-1)}$.

We now investigate the eigenvector matrix $\hat{U}_k$ of T_k. Due to the structure of T_k, we conclude that $\left\{\{e_j\}_{j=2}^{m_1-2}, (e_1 - e_{m_1-1})/\sqrt{2}\right\} \in \mathrm{null}(T_k)$. Consequently, those $m_1 - 2$ orthonormal vectors are eigenvectors of T_k corresponding to the eigenvalue zero.

The assumption $|\lambda_{1,k}^{(SC)} - \lambda_{m_1,k}^{(SC)}| \geq \eta_3$ implies that the corresponding eigenvectors $u^{(1),k}$ and $u^{(m_1),k}$ are linearly independent. Furthermore, $|\lambda_{(1,m_1),k}^{(SC)}| \geq \eta_2$ leads to $u^{(1,m_1),k} \notin \mathrm{null}(T_k)$. Thus, T_k has m_1 linearly independent eigenvectors. Using the assumption $|\lambda_{(1,m_1),k}^{(SC)}| \geq \eta_2$, $\sigma_0 \neq 0$, and the formulas for the entries of T_k given in Theorem 2, the two nontrivial eigenvectors become

$$u_p^{(1,m_1),k} = \lambda_{(1,m_1),k}^{(SC)} \sigma_p + d(\sigma_0 \sigma_{p-1} - \sigma_{-1}\sigma_p), \qquad p = 1, \ldots, m_1.$$

By substituting σ_p with formulas from Lemma 2, the formulas for $u^{(1,m_1),k}$ follow. We then obtain

$$\hat{U}_k^{-1} T_k \hat{U}_k = \Lambda_k = \mathrm{diag}_{m_1}(\lambda_{1,k}^{(SC)}, 0, \ldots, 0, \lambda_{m_1,k}^{(SC)}), \qquad k = 1, \ldots, m_2.$$

Now the matrix $W_{M^{-1}B} = (V_2 \otimes I_{m_1})\mathrm{diag}_{k,m_2}(\hat{U}_k)$ is nonsingular and

$$\begin{aligned} W_{M^{-1}B}^{-1} M^{-1} B W_{M^{-1}B} &= I + \mathrm{diag}_{k,m_2}(\hat{U}_k)^{-1}(V_2 \otimes I_{m_1})^{-1} M^{-1} E (V_2 \otimes I_{m_1}) \mathrm{diag}_{k,m_2}(\hat{U}_k) \\ &= I + \mathrm{diag}_{k,m_2}(\hat{U}_k^{-1}) \mathrm{diag}_{k,m_2}(T_k) \mathrm{diag}_{k,m_2}(\hat{U}_k) = I + \mathrm{diag}_{k,m_2}(\Lambda_k), \end{aligned}$$

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

Later (in Corollary 1) it is established that the assumptions in Theorem 3 are valid, when m_1 and m_2 are sufficiently large. Later we will study the condition number reduction $\psi \equiv \mathrm{cond}_2(W_{M^{-1}B}) / \mathrm{cond}_2(W_B)$, where W_B is the eigenvector matrix of B. For the moment we just state the following theorem.

THEOREM 4. *Under the same assumptions as in Theorem* 3,

$$\psi \leq \psi_0 = \max_{1 \leq k \leq m_2} \|\hat{U}_k\|_2 \cdot \max_{1 \leq k \leq m_2} \|\hat{U}_k^{-1}\|_2 \Big/ \mathrm{cond}_2(\hat{V}_1),$$

where $\hat{V}_1$ *is the eigenvector matrix of* $\hat{B}_1$.

Proof. We obtain

$$\begin{aligned} \mathrm{cond}_2(W_{M^{-1}B}) &= \|W_{M^{-1}B}\|_2 \|W_{M^{-1}B}^{-1}\|_2 \\ &= \|(V_2 \otimes I_{m_1})\mathrm{diag}_{k,m_2}(\hat{U}_k)\|_2 \|\mathrm{diag}_{k,m_2}(\hat{U}_k^{-1})(V_2^{-1} \otimes I_{m_1})\|_2 \\ &\leq \|V_2 \otimes I_{m_1}\|_2 \|V_2^{-1} \otimes I_{m_1}\|_2 \|\mathrm{diag}_{k,m_2}(\hat{U}_k)\|_2 \|\mathrm{diag}_{k,m_2}(\hat{U}_k^{-1})\|_2 \\ &= \mathrm{cond}_2(V_2) \cdot \max_{1 \leq k \leq m_2} \|\hat{U}_k\|_2 \cdot \max_{1 \leq k \leq m_2} \|\hat{U}_k^{-1}\|_2. \end{aligned}$$

We also have $W_B = V_2 \otimes \hat{V}_1$ implying

$$\begin{aligned} \mathrm{cond}_2(W_B) &= \|V_2 \otimes \hat{V}_1\|_2 \|V_2^{-1} \otimes \hat{V}_1^{-1}\|_2 = \|V_2\|_2 \|\hat{V}_1\|_2 \|V_2^{-1}\|_2 \|\hat{V}_1^{-1}\|_2 \\ &= \mathrm{cond}_2(V_2) \cdot \mathrm{cond}_2(\hat{V}_1). \end{aligned}$$

Employing these results yields

$$\psi = \frac{\operatorname{cond}_2(W_{M^{-1}B})}{\operatorname{cond}_2(W_B)} \le \max_{1\le k\le m_2} \|\hat{U}_k\|_2 \cdot \max_{1\le k\le m_2} \|\hat{U}_k^{-1}\|_2 \Big/ \operatorname{cond}_2(\hat{V}_1) \equiv \psi_0. \qquad \square$$

Using the information on the spectrum of B_2 given in Lemma 1 and a condition on the grid, it is possible to give asymptotic expressions for the eigenvalues of $M^{-1}E$.

THEOREM 5. *Assume that*

$$0 < \alpha < 1$$

and

$$\frac{m_2(1+2m_2^{-3/2})}{m_1(1-\gamma m_1^{\alpha-1})} \le \varphi\,, \qquad \varphi < 1.$$

Then, in the limit $m_1, m_2 \to \infty$, the nonzero eigenvalues of $M^{-1}E$ approach two curve segments given by

$$\lambda_1^{(SC)}(\theta) = \frac{2-\sqrt{1-\theta^2}+i\theta-\sqrt{-3+6\theta^2-4\sqrt{1-\theta^2}+i\theta(4+6\sqrt{1-\theta^2})}}{4\sqrt{1-\theta^2}},$$
$$-\varphi \le \theta < 0,$$
$$\lambda_1^{(SC)}(\theta) = \frac{2-\sqrt{1-\theta^2}+i\theta+\sqrt{-3+6\theta^2-4\sqrt{1-\theta^2}+i\theta(4+6\sqrt{1-\theta^2})}}{4\sqrt{1-\theta^2}},$$
$$0 \le \theta \le \varphi,$$

$$\lambda_\infty^{(SC)}(\theta) = \bar{\lambda}_1^{(SC)}(\theta)\,, \quad -\varphi \le \theta \le \varphi.$$

Proof. Using $\delta = \gamma m_1^{\alpha-1}$ and $\beta = m_2/m_1$, ζ_k can be rewritten

$$\zeta_k = \frac{\gamma m_1^{\alpha-1}}{1-\gamma m_1^{\alpha-1}} + \frac{m_2\lambda_{k,2}}{2m_1(1-\gamma m_1^{\alpha-1})} \equiv \frac{\gamma m_1^{\alpha-1}}{1-\gamma m_1^{\alpha-1}} + w_k.$$

Exploiting that $\lambda_{k,2} \in \mathcal{E}^+(4m_2^{-3/4}, 2+4m_2^{-3/2})$ yields

$$w_k \in \mathcal{E}^+\left(\frac{2m_2^{1/4}}{m_1(1-\gamma m_1^{\alpha-1})}, \frac{m_2(1+2m_2^{-3/2})}{m_1(1-\gamma m_1^{\alpha-1})}\right), \qquad k = 1, \ldots, m_2.$$

By assumption we have $m_2/m_1 \le \varphi(1-\gamma m_1^{\alpha-1})/(1+2m_2^{-3/2}) < 1$ yielding

$$\frac{2m_2^{1/4}m_1^{-1}}{1-\gamma m_1^{\alpha-1}} < \frac{2m_1^{-3/4}}{1-\gamma m_1^{\alpha-1}},$$

which implies $w_k \in \mathcal{E}^+(2m_1^{-3/4}/(1-\gamma m_1^{\alpha-1}), \varphi)$. Thus,

$$\zeta_k = \epsilon + i\theta_k\,, \qquad -\varphi \le \theta_k \le \varphi,$$
$$\epsilon = \frac{\gamma m_1^{\alpha-1} + 2\delta_k m_1^{-3/4}}{1-\gamma m_1^{\alpha-1}}\,, \qquad 0 \le \delta_k \le 1.$$

For $m_1 \gg 1$ we have $0 < \epsilon \ll 1$, and a Taylor expansion yields

$$z_k = -\zeta_k + \sqrt{d+\zeta_k^2} = -\epsilon - i\theta_k + \sqrt{d-\theta_k^2+\epsilon^2+i2\epsilon\theta_k}$$
$$= \sqrt{d-\theta_k^2} - i\theta_k - \epsilon + \frac{i\epsilon\theta_k}{\sqrt{d-\theta_k^2}} + \mathcal{O}(\epsilon^2).$$

Using $d = (1+\gamma m_1^{\alpha-1})/(1-\gamma m_1^{\alpha-1})$ we obtain $z_{k,\infty} \equiv \lim_{m_1\to\infty} z_k = \sqrt{1-\theta_k^2} - i\theta_k$ and

$$|z_k|^2 = d\left(1 - \frac{2\epsilon}{\sqrt{d-\theta_k^2}}\right) + \mathcal{O}(\epsilon^2)$$
$$= 1 - \frac{2\gamma m_1^{\alpha-1}}{1-\gamma m_1^{\alpha-1}}\left(\frac{1}{\sqrt{d-\theta_k^2}} - 1\right) - \frac{4\delta_k m_1^{-3/4}}{(1-\gamma m_1^{\alpha-1})\sqrt{d-\theta_k^2}} + \mathcal{O}(m_1^{\alpha-1}\epsilon + \epsilon^2).$$

We now determine $\lim_{m_1\to\infty} R_\ell(z_k)$, $\ell = 1, 2$. First consider the case $\theta_k^2 > 0$. Choose $\theta_0^2 > 0$ arbitrarily small. For m_1 sufficiently large we have $2\gamma m_1^{\alpha-1}/(1-\gamma m_1^{\alpha-1}) \le \theta_0^2/2$. For $\theta_0^2 \le \theta_k^2 \le \varphi^2$, we obtain

$$\frac{\theta_0^2}{2} + d - 1 = \frac{\theta_0^2}{2} + \frac{2\gamma m_1^{\alpha-1}}{1-\gamma m_1^{\alpha-1}} \le \theta_0^2 \le \theta_k^2,$$

implying

$$\frac{1}{\sqrt{d-\theta_k^2}} - 1 \ge \frac{1}{\sqrt{1-\theta_0^2/2}} - 1 \ge 1 - \sqrt{1-\frac{\theta_0^2}{2}} \ge \frac{\theta_0^2}{4}.$$

This yields

$$|z_k|^2 \le 1 - 2\gamma m_1^{\alpha-1}\frac{\theta_0^2}{4} + \mathcal{O}(m_1^{\alpha-1}\epsilon + \epsilon^2) = 1 - \frac{\gamma\theta_0^2}{2}m_1^{\alpha-1} + \mathcal{O}(m_1^q),$$
$$q = \max\left(2\alpha - 2, \alpha - \frac{7}{4}, -\frac{3}{2}\right).$$

$0 < \alpha < 1$ leads to $\alpha - 1 > q$ implying

$$\lim_{m_1\to\infty}\left(1 - \frac{\gamma\theta_0^2}{2}m_1^{\alpha-1} + \mathcal{O}(m_1^q)\right)^{m_1/2} = 0,$$

and consequently $\lim_{m_1\to\infty} z_k^{m_1} = 0$. This and $d^{-1} \le 1$ implies $\lim_{m_1\to\infty}(-d^{-1}z_k)^{m_1} = 0$, which yields

$$\lim_{m_1\to\infty} R_1(z_k) = \lim_{m_1\to\infty} R_2(z_k) = 0.$$

Now consider the case $\theta_k = 0$. We obtain $\Im m(w_k) = \Im m(\lambda_{k,2}) = 0$ and

$$\zeta_k = \frac{\gamma m_1^{\alpha-1}}{1-\gamma m_1^{\alpha-1}} + w_k = \epsilon,$$

and consequently $d = 1 + 2\gamma m_1^{\alpha-1}/(1-\gamma m_1^{\alpha-1}) = 1 + 2\epsilon - 2w_k$. This yields

$$|z_k| = -\epsilon + \sqrt{1 + 2\epsilon - 2w_k + \epsilon^2} = -\epsilon + (1+\epsilon)\sqrt{1 - 2w_k(1+\epsilon)^{-2}}$$

$$\leq -\epsilon + (1+\epsilon)\big(1 - w_k(1+\epsilon)^{-2}\big) = 1 - \frac{w_k}{1+\epsilon}$$

$$= 1 - \frac{m_2\lambda_{k,2}}{2m_1(1-\gamma m_1^{\alpha-1})} \Big/ \left(1 + \frac{\gamma m_1^{\alpha-1}}{1-\gamma m_1^{\alpha-1}} + \frac{m_2\lambda_{k,2}}{2m_1(1-\gamma m_1^{\alpha-1})}\right)$$

$$= 1 - \frac{m_2\lambda_{k,2}}{2m_1} \Big/ \left(1 + \frac{m_2\lambda_{k,2}}{2m_1}\right) = \frac{1}{1 + m_2\lambda_{k,2}/(2m_1)}.$$

Using $\lambda_{k,2} > \ln(m_2)/m_2$ from Lemma 1, we obtain

$$|z_k|^{m_1} \leq \frac{1}{\big(1 + \ln(m_2)/(2m_1)\big)^{m_1}} \leq \frac{1}{1 + \ln(m_2)/2}$$

implying

$$\lim_{m_1\to\infty} |z_k|^{m_1} \leq \lim_{m_2\to\infty} \frac{1}{1 + \ln(m_2)/2} = 0.$$

$|z_k| \leq 1$ leads to $|-d^{-1}z_k| \leq d^{-1} \leq 1 - \gamma m_1^{\alpha-1}$ implying $\lim_{m_1\to\infty} |-d^{-1}z_k|^{m_1} = 0$. This yields

$$\lim_{m_1\to\infty} R_1(z_k) = \lim_{m_1\to\infty} R_2(z_k) = 0.$$

With $\lim_{m_1\to\infty} d = 1$, we now obtain

$$\lim_{m_1\to\infty} \lambda^{(SC)}_{(1,m_1),k} = \frac{2 - z_{k,\infty} \pm \sqrt{-3z_{k,\infty}^2 - 4z_{k,\infty}}}{2(z_{k,\infty} + z_{k,\infty}^{-1})}$$

$$= \frac{2 - \sqrt{1-\theta_k^2} + i\theta_k \pm \sqrt{-3 + 6\theta_k^2 - 4\sqrt{1-\theta_k^2} + i\theta_k(4 + 6\sqrt{1-\theta_k^2})}}{4\sqrt{1-\theta_k^2}}.$$

Exploiting $-\varphi \leq \theta_k \leq \varphi$ and distinguishing the branch cut of the square root function, we can conclude that the nonzero eigenvalues of $M^{-1}E$ approach the two curve segments, $\lambda_1^{(SC)}(\theta)$ and $\lambda_\infty^{(SC)}(\theta)$. □

We restate the following corollary originally established in [Otto92].

COROLLARY 1. *It holds that*

$$1 + \lambda_1^{(SC)}(-\varphi) = \chi_1 + \mathcal{O}(\sqrt{1-\varphi}),$$

$$1 + \lambda_1^{(SC)}(0) = \frac{5 + \sqrt{7}i}{4},$$

$$1 + \lambda_1^{(SC)}(\varphi) = \chi_2 + \mathcal{O}(\sqrt{1-\varphi}),$$

$$\frac{15}{28} < |1 + \lambda_1^{(SC)}(\theta)| < 1 + \frac{\sqrt{5}+\sqrt{7}}{4\sqrt{1-\varphi^2}}, \qquad -\varphi \le \theta \le \varphi,$$

$$\frac{5}{14} < |\lambda_1^{(SC)}(\theta)| < \frac{\sqrt{5}+\sqrt{7}}{4\sqrt{1-\varphi^2}}, \qquad -\varphi \le \theta \le \varphi,$$

$$\lim_{m_1 \to \infty} |\lambda_{1,k}^{(SC)} - \lambda_{m_1,k}^{(SC)}| \ge \frac{\sqrt{7}}{2}, \qquad k = 1, \ldots, m_2,$$

where

$$\chi_1 \equiv \frac{4+2i}{5},$$

$$\chi_2 \equiv \frac{2+i}{2\sqrt{1-\varphi^2}} + \frac{7+4i}{10}.$$

Proof. The formulas for $1 + \lambda_1^{(SC)}(0)$ and $1 + \lambda_1^{(SC)}(\pm\varphi)$ are directly derived from the formula for $\lambda_1^{(SC)}(\theta)$ and Taylor expansions. Using the triangle inequality, we obtain

$$|\lambda_1^{(SC)}(\theta)| \le \frac{|2 - \sqrt{1-\theta^2} + i\theta| + \sqrt{|-3(\sqrt{1-\theta^2} - i\theta)^2 - 4(\sqrt{1-\theta^2} - i\theta)|}}{4\sqrt{1-\theta^2}}$$

$$\le \frac{\sqrt{5 - 4\sqrt{1-\theta^2}} + \sqrt{3+4}}{4\sqrt{1-\theta^2}} < \frac{\sqrt{5}+\sqrt{7}}{4\sqrt{1-\varphi^2}},$$

and consequently

$$|1 + \lambda_1^{(SC)}(\theta)| < 1 + \frac{\sqrt{5}+\sqrt{7}}{4\sqrt{1-\varphi^2}}.$$

Introducing $\epsilon = \sqrt{1-\theta^2}$, $0 < \epsilon \le 1$, we get

$$|1 + \lambda_1^{(SC)}(\theta)| \ge \frac{|2 + 3\sqrt{1-\theta^2} + i\theta| - \sqrt{|-3 + 6\theta^2 - 4\sqrt{1-\theta^2} + i\theta(4 + 6\sqrt{1-\theta^2})|}}{4\sqrt{1-\theta^2}}$$

$$= \frac{\sqrt{5 + 12\sqrt{1-\theta^2} + 8(1-\theta^2)} - (25 + 24\sqrt{1-\theta^2})^{1/4}}{4\sqrt{1-\theta^2}}$$

$$= \frac{\sqrt{5 + 12\epsilon + 8\epsilon^2} - \sqrt{5}(1 + \frac{24}{25}\epsilon)^{1/4}}{4\epsilon} \ge \frac{\sqrt{5 + 12\epsilon + 8\epsilon^2} - \sqrt{5}(1 + \frac{6}{25}\epsilon)}{4\epsilon}$$

$$\ge \frac{6\sqrt{5}}{25} > \frac{15}{28}.$$

We also obtain

$$|\lambda_1^{(SC)}(\theta)| \ge \frac{\sqrt{|-3 + 6\theta^2 - 4\sqrt{1-\theta^2} + i\theta(4 + 6\sqrt{1-\theta^2})|} - |2 - \sqrt{1-\theta^2} + i\theta|}{4\sqrt{1-\theta^2}}$$

$$= \frac{(25 + 24\epsilon)^{1/4} - \sqrt{5 - 4\epsilon}}{4\epsilon} \ge \frac{\sqrt{5}(1 + \frac{6\epsilon}{25} - \frac{54\epsilon^2}{625}) - \sqrt{5}(1 - \frac{2\epsilon}{5} - \frac{2\epsilon^2}{25} - \frac{4\epsilon^3}{125})}{4\epsilon}$$

$$= \left(\frac{4}{25} + \frac{5\epsilon^2 - \epsilon}{625}\right)\sqrt{5} > \frac{5}{14}.$$

Finally, we have

$$\lim_{m_1\to\infty} |\lambda_{1,k}^{(SC)} - \lambda_{m_1,k}^{(SC)}| = \left| \frac{\sqrt{-3z_{k,\infty}^2 - 4z_{k,\infty}}}{z_{k,\infty} + z_{k,\infty}^{-1}} \right|$$

$$= \frac{\sqrt{|-3 + 6\theta_k^2 - 4\sqrt{1-\theta_k^2} + i\theta_k(4 + 6\sqrt{1-\theta_k^2})|}}{2\sqrt{1-\theta_k^2}}$$

$$= \frac{(25 + 24\epsilon_k)^{1/4}}{2\epsilon_k} \geq \frac{\sqrt{7}}{2}, \qquad k = 1, \ldots, m_2. \qquad \square$$

We make the following important observations:

(i) In the limit $m_1, m_2 \to \infty$, the restriction on the grid can be reduced to $\beta = m_2/m_1 = \varphi < 1$.

(ii) The curves $\lambda_{1,\infty}^{(SC)}(\theta)$, and hence the estimates in Corollary 1, are independent of α and γ.

(iii) The convergence of the eigenvalues $\lambda_{(1,m_1),k}^{(SC)}$ to the curves $\lambda_{1,\infty}^{(SC)}(\theta)$ essentially depends on that $d = (1 + \gamma m_1^{\alpha-1})/(1 - \gamma m_1^{\alpha-1}) \to 1$ and $R_{1,2}(z_k) \to 0$. Depending on the choice of parameters, these processes occur with different relative speed.

(iv) The interesting case $\alpha = 0$ is excluded from the theorem, since the remainder terms $R_1(z_k)$ and $R_2(z_k)$ will not vanish as $m_1 \to \infty$. This means that $\lambda_{(1,m_1),k}^{(SC)}$ will not converge to $\lambda_{1,\infty}^{(SC)}(\theta)$.

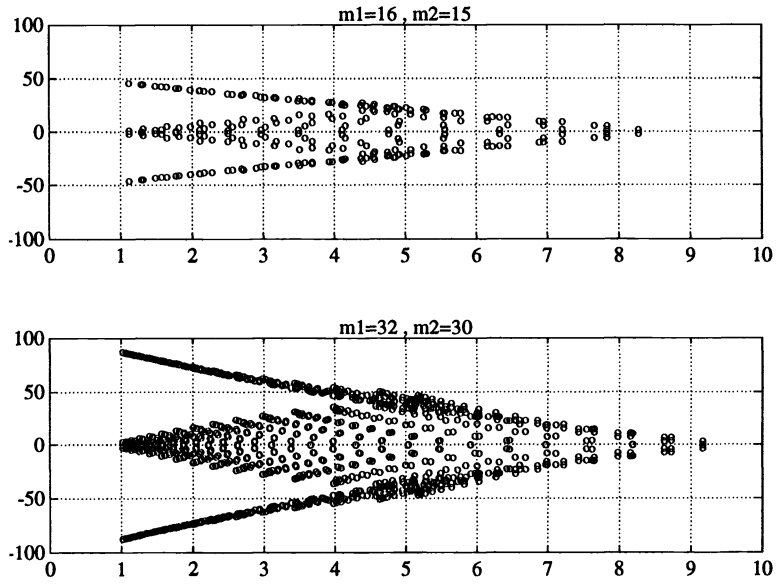


FIG. 1. *Spectra for B when* $\alpha = 0.1$ *and* $\gamma = 0.5$.

In Fig. 1 we show spectra of the unpreconditioned coefficient matrix B for two problem sizes. The parameters chosen for the artificial viscosity are $\alpha = 0.1$ and $\gamma = 0.5$. It is clear that the spectrum of B grows approximately linearly with the number of gridpoints in the fastest growing space direction, since B is a discretization of a first-order differential

operator. In Theorem 5 we have proved that this growth is removed by using the semicirculant preconditioner M. In Figs. 2–4 we show numerically computed spectra for three different choices of parameters. For each parameter set, the eigenvalues of $M^{-1}B$ for two different problem sizes are plotted together with the asymptotic curves $1 + \lambda_{1,\infty}^{(SC)}(\theta)$. The eigenvalues are computed using the formula in Theorem 2, where we have used "exact" eigenvalues of B_2. For all spectra below, we have used $\beta = \frac{15}{16} = 0.9375$. Hence, the curves $1 + \lambda_{1,\infty}^{(SC)}(\theta)$ (which are *not* straight lines) are identical in all graphs. We note that:

(i) The multiplicity of the eigenvalue at 1 is 3720 for the 64×60 problems and 244800 for the 512×480 problems.

(ii) For $\alpha = 0.9$, $R_{1,2}(z_k)$ converges to zero much faster than d converges to one. Hence, the spectrum quickly becomes two "simple" curves, which slowly converge to $1 + \lambda_{1,\infty}^{(SC)}(\theta)$.

(iii) For $\alpha = 0.1$, $R_{1,2}(z_k)$ converges to zero slower than d converges to one. Hence, the spectrum centers around $1 + \lambda_{1,\infty}^{(SC)}(\theta)$, but the oscillations of $R_{1,2}(z_k)$ are still present in the spectrum for the large problem.

(iv) For $\alpha = 0$, $R_{1,2}(z_k)$ will not converge to zero. However, in Fig. 4 we see that the spectrum is contained in a finite region well separated from zero, even for large problems. We have so far not been able to prove this.

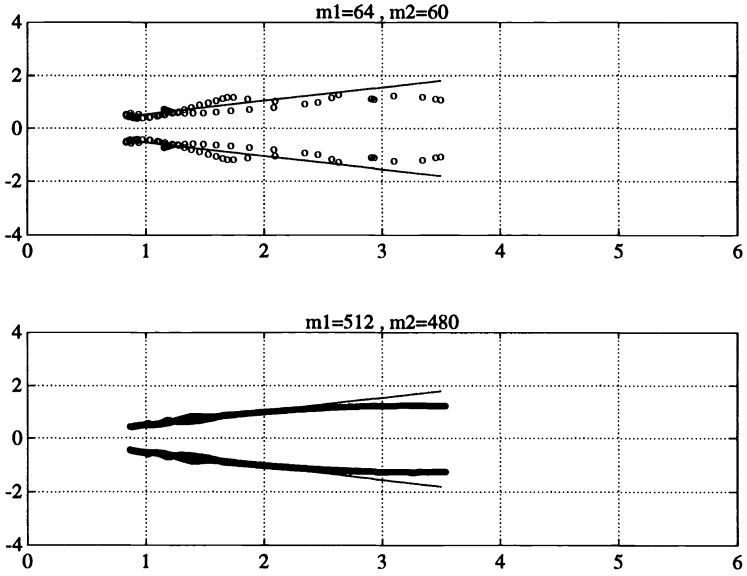


FIG. 2. *Spectra for* $M^{-1}B$ *when* $\alpha = 0.9$ *and* $\gamma = 0.05$.

7. The time-dependent problem. For comparison we restate here the results from [Otto92] using the theorems in §6. A semicirculant preconditioner, with parameters chosen as prescribed in (4), for the coefficient matrix in (7) is given by

$$M = I_{m_2} \otimes C_1 + \kappa_2 B_2 \otimes I_{m_1}, \tag{14}$$

where

$$C_1 = \text{ctrid}_{m_1}(-\kappa_1, 4, \kappa_1).$$

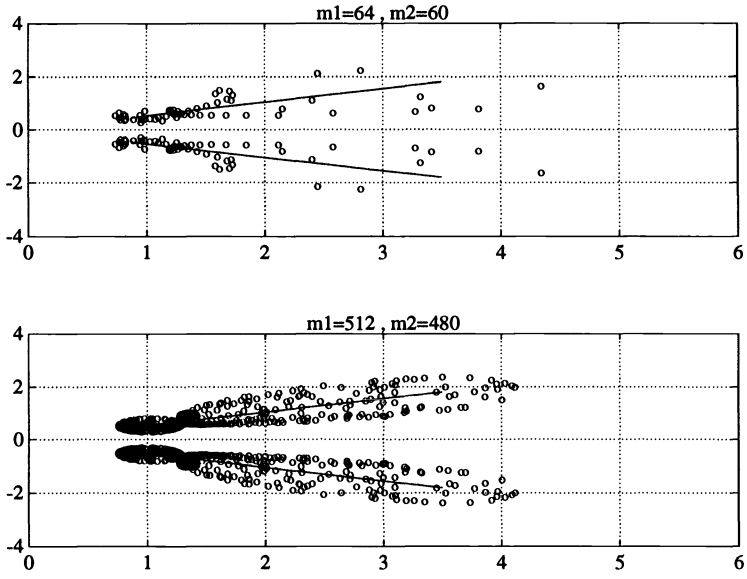


FIG. 3. *Spectra for* $M^{-1}B$ *when* $\alpha = 0.1$ *and* $\gamma = 0.5$.

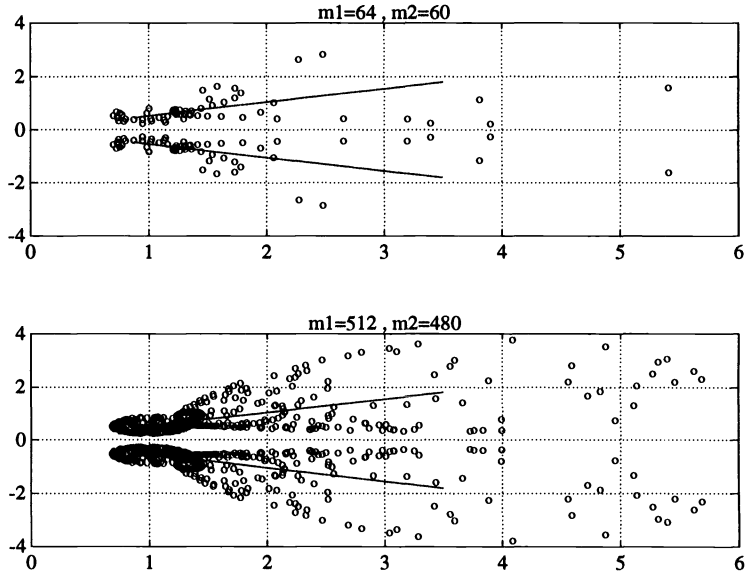


FIG. 4. *Spectra for* $M^{-1}B$ *when* $\alpha = 0$ *and* $\gamma = 0.5$.

We again define the error matrix $E \equiv B - M$. Now put $d = 1$ and define ζ_k by

$$\zeta_k \equiv \frac{4 + \kappa_2 \lambda_{k,2}}{2\kappa_1},$$

where $\lambda_{k,2}$ are the eigenvalues of B_2. Now Theorem 2 applies to this problem too. With some technical changes in the proof, Theorem 5 is also applicable provided

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

and

$$\frac{m_2(1+2m_2^{-3/2})}{m_1} \le \varphi\,, \qquad \varphi < 1.$$

The case $\alpha = 0$ could be allowed, since the remainder terms would be bounded.

8. Convergence rate. We solve (1) using GMRES(ℓ), i.e., the restarted GMRES iteration [SaadSch86]. If GMRES is not restarted it is a minimal residual iteration [FrGoNa91]. Hence, in iteration i we have

$$\|r_i\|_2 = \min_{p_i \in \mathcal{P}_i,\ p_i(0)=1} \|p_i(M^{-1}B)r_0\|_2,$$

where $r_i \equiv M^{-1}(Bu_i - b)$ and u_i is the approximation of u computed in iteration i. Since $M^{-1}B$ is diagonalizable by Theorem 3, we obtain

$$\frac{\|r_i\|_2}{\|r_0\|_2} \le \mathrm{cond}_2(W_{M^{-1}B}) \cdot \min_{p_i \in \mathcal{P}_i,\ p_i(0)=1} \max_{1 \le \ell \le n} |p_i(\lambda_\ell)| \equiv \mathrm{cond}_2(W_{M^{-1}B}) \cdot \varepsilon_i, \tag{15}$$

where $W_{M^{-1}B}$ is the eigenvector matrix and λ_ℓ the eigenvalues of $M^{-1}B$. $\mathcal{P}_i$ is the set of all polynomials of maximal degree i. For our model problems Theorem 2 yields that $\{\lambda_\ell\}_{\ell=1}^n = \{1\} \cup \{1 + \lambda^{(SC)}_{(1,m_1),k}\}_{k=1}^{m_2}$. It is normal to define the *asymptotic convergence factor* ϱ by

$$\varrho \equiv \lim_{i \to \infty} \varepsilon_i^{1/i}. \tag{16}$$

The points χ_1 and χ_2 defined in Corollary 1 determine a circle $\mathcal{C}(c, q)$ with center on the positive real axis. Here c denotes the distance from the origin to the center, and q is the radius. In Fig. 5 the curves $1 + \lambda^{(SC)}_{1,\infty}(\theta)$ are plotted for $\varphi = \frac{15}{16}$. The points $\chi_1, \bar{\chi}_1, \chi_2, \bar{\chi}_2$ are denoted by an asterisk (*), and the circle $\mathcal{C}(c, q)$ is drawn as a dotted line.

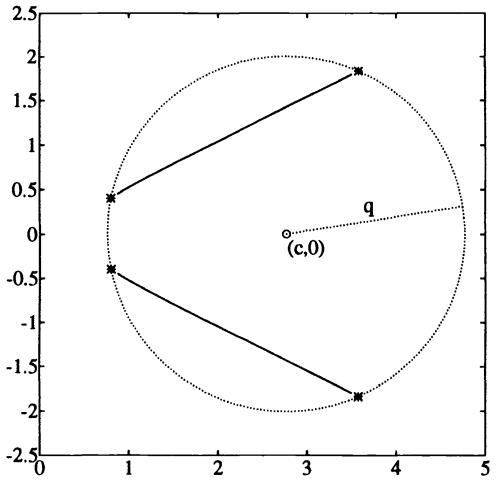


FIG. 5. *Circle* $\mathcal{C}(c, q)$.

Convincing results from numerical experiments [Otto92] corroborate that the curves $1 + \lambda^{(SC)}_{1,\infty}(\theta)$ lie inside $\mathcal{C}(c, q)$, i.e.,

$$|1 + \lambda^{(SC)}_{1,\infty}(\theta) - c| < q\,, \quad -\varphi \le \theta \le \varphi\,, \quad \varphi < 1. \tag{17}$$

In [Saad81] it is shown that, if the spectrum of $M^{-1}B$ is enclosed by $\mathcal{C}(c,q)$ then $\varrho \leq \frac{q}{c}$. Exploiting this result we can prove that, for sufficiently large problems, ϱ is bounded by a quantity less than one.

THEOREM 6. *Presume that the assumptions in Theorem* 5 *are fulfilled and* (17) *holds. Then for* $\epsilon > 0$ *arbitrarily small, there are integers* $\bar{m}_1(\epsilon)$ *and* $\bar{m}_2(\epsilon)$ *such that* ϱ *is bounded by*

$$\varrho < \varrho_\epsilon(\varphi) = \left(\frac{1 + (\frac{40}{225} + \frac{116}{25}\epsilon + \frac{36}{25}\epsilon^2)\sqrt{1-\varphi^2}}{1 + \frac{616}{225}\sqrt{1-\varphi^2}} \right)^{1/2},$$

when $m_1 > \bar{m}_1(\epsilon)$ *and* $m_2 > \bar{m}_2(\epsilon)$.

Proof. The proof follows the technique presented in [Otto92]. Using $\tau \equiv \sqrt{1-\varphi^2}$, $0 < \tau \leq 1$, we have

$$\chi_1 = \frac{4+2i}{5}, \qquad \chi_2 = \left(\tau^{-1} + \frac{7}{10}\right) + \left(\frac{1}{2}\tau^{-1} + \frac{4}{10}\right) i.$$

The circle $\mathcal{C}(c,q)$ is uniquely determined by $q^2 = \big(\Re e(\chi_1) - c\big)^2 + \big(\Im m(\chi_1)\big)^2 = \big(\Re e(\chi_2) - c\big)^2 + \big(\Im m(\chi_2)\big)^2$, which results in

$$c = \frac{25 + 36\tau - 3\tau^2}{4\tau(10-\tau)}, \qquad q = \sqrt{c^2 - \frac{8}{5}c + \frac{4}{5}}.$$

A simple analysis of c yields

$$c^{-1} \leq \frac{\frac{58}{25}\tau}{1 + \frac{616}{225}\tau} \leq \frac{18}{29}.$$

After some careful analysis we obtain

$$\frac{q^2}{c^2} = 1 - \frac{8}{5}c^{-1} + \frac{4}{5}c^{-2} = 1 - \frac{32\tau(10-\tau)(25+16\tau-\tau^2)}{5(25+36\tau-3\tau^2)^2} \leq \frac{1 + \frac{40}{225}\tau}{1 + \frac{616}{225}\tau},$$

implying that $q < c$. Choose $\epsilon > 0$ arbitrarily small. According to Theorem 5 and (17), there exist $\bar{m}_1(\epsilon)$ and $\bar{m}_2(\epsilon)$ such that, the spectrum of $M^{-1}B$ is enclosed by $\mathcal{C}(c, q+\epsilon)$ when $m_1 > \bar{m}_1$ and $m_2 > \bar{m}_2$. Some algebra yields

$$\frac{(q+\epsilon)^2}{c^2} < \frac{q^2}{c^2} + \left(2\epsilon + \frac{18}{29}\epsilon^2\right) \frac{\frac{58}{25}\tau}{1 + \frac{616}{225}\tau} \leq \frac{1 + (\frac{40}{225} + \frac{116}{25}\epsilon + \frac{36}{25}\epsilon^2)\tau}{1 + \frac{616}{225}\tau},$$

and the desired result follows. □

Directly we establish that, for $0 < \epsilon < (\sqrt{1417} - 29)/18 \approx 0.48$, $\varrho_\epsilon(\varphi) < 1$. So there is a margin to enclose the spectrum, when the number of gridpoints is large but still finite.

We now study the upper bound ψ_0 of the condition number reduction ψ defined in Theorem 4. Figure 6 shows ψ_0 for problems of size $2^p \times \frac{15}{16}2^p$, $p = 4, \ldots, 10$. The solid line represents problems with $\alpha = 0.9$ and $\gamma = 0.05$, the dashed line represents $\alpha = 0.1$ and $\gamma = 0.5$, and finally the dashed-dotted line represents $\alpha = 0$ and $\gamma = 0.5$. We see that for increasing problem size, ψ_0 is less than one and decreases dramatically for all values of α examined. Thus, the condition number in estimate (15) is lower for the preconditioned system than for the unpreconditioned system. Also, the gain increases with problem size. So it is clear that

applying the semicirculant preconditioner improves *both* the spectrum *and* the behavior of the eigenvectors of the iteration operator. On the other hand, Theorem 4 yields

$$(18)\quad \mathrm{cond}_2(W_{M^{-1}B}) \le \psi_0 \cdot \mathrm{cond}_2(V_2 \otimes \hat{V}_1) = \max_{1\le k\le m_2} \|\hat{U}_k\|_2 \cdot \max_{1\le k\le m_2} \|\hat{U}_k^{-1}\|_2 \cdot \mathrm{cond}_2(V_2).$$

An analysis shows that $\max_{1\le k\le m_2} \|\hat{U}_k^{-1}\|_2 \to \infty$ when $m_1, m_2 \to \infty$. However, numerical experiments indicate that this growth is slow, whereas $\mathrm{cond}_2(V_2)$ grows significantly with m_2. This is expected since V_2 is the eigenvector matrix of B_2, which corresponds to a discretization of a PDE lacking eigenfunctions. Therefore, the upper bound of $\mathrm{cond}_2(W_{M^{-1}B})$ grows mainly due to the "degenerate" eigenvectors of B_2.

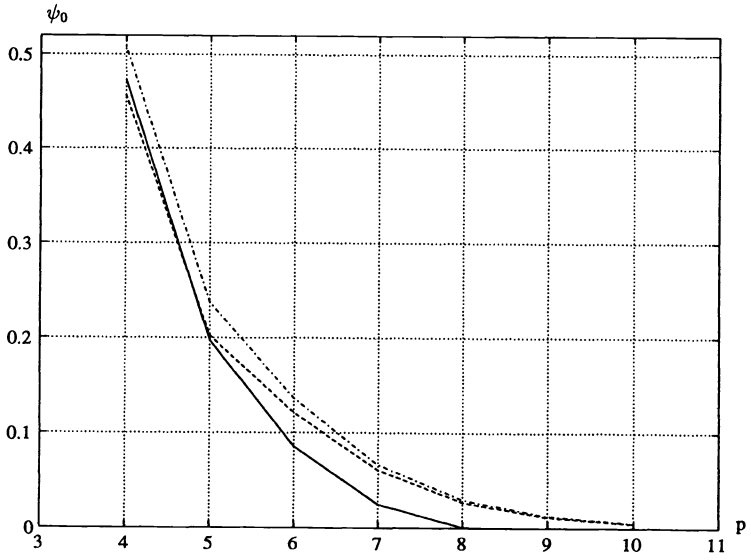


FIG. 6. ψ_0 *as a function of problem size.*

In Fig. 7 the solid line is the graph of the function $\varrho_0(\beta)$, which is the upper bound of ϱ and is defined in Theorem 6, in the limit $m_1, m_2 \to \infty$. The figure also shows the empirically derived convergence factors using the GMRES(20) iteration for some problems where $\alpha = 0.9$ and $\gamma = 0.05$. The symbols $\times$, $\circ$, and $*$ mark problems with $128 \times 128\beta$, $256 \times 256\beta$, $512 \times 512\beta$ unknowns, respectively. We see that, at least for large problems, a substantial improvement in the convergence factor is achieved when reducing β from 1 to ~ 0.9. Hence, we do not have to use many more points in the x_1-direction than in the x_2-direction to attain good convergence. The relatively high convergence factor, for small problems and low grid ratios, seems to be an effect of the finite problem size. Also, the growing upper bound of $\mathrm{cond}_2(W_{M^{-1}B})$ for increasing problem size seems to have little effect on the convergence factor. For practical computations, the bound on the asymptotic convergence factor derived in Theorem 6 governs the convergence.

For nonnormal matrices there is, to our knowledge, no theory that reduces the study of convergence for polynomial iterations with the matrix $M^{-1}B$ to a study of polynomials at $\mathcal{O}(n)$ "easily" computable points such as the eigenvalues $\{\lambda_i\}_{i=1}^n$. No strict convergence estimates are available for "closely nonnormal" or "asymptotically nonnormal" coefficient matrices.

9. Numerical experiments. We show the number of GMRES(6) iterations required to attain $\|r_i\|_2/\|r_0\|_2 \le 10^{-6}$ for the steady-state problem. The initial guess is $u_0 = 0$, and

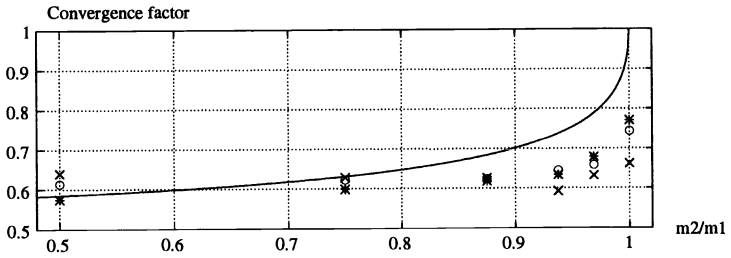


FIG. 7. *The convergence factor as a function of* $\beta = m_2/m_1$.

problems of size $2^p \times \frac{15}{16}2^p$, $p = 4, \ldots, 9$, are studied. We present the results for the same parameter sets as in Figs. 2–4. The solid line represents problems with $\alpha = 0.9$ and $\gamma = 0.05$, the dashed line represents $\alpha = 0.1$ and $\gamma = 0.5$, and finally the dashed-dotted line represents $\alpha = 0$ and $\gamma = 0.5$.

In Fig. 8 we see that for $\alpha > 0$, the number of iterations required does not grow with problem size for large problems. Similar results for the time-dependent problem presented in [Otto92] show that the number of iterations is indeed independent of problem size and κ. Note that also for $\alpha = 0$, there is no large growth in the number of iterations when the problem size is increased.

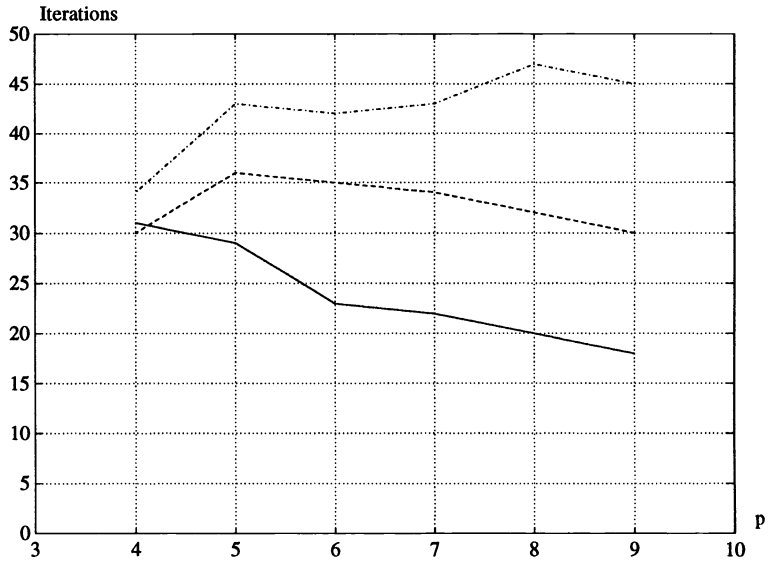


FIG. 8. *The number of* GMRES(6) *iterations required.*

We have also used other iterative methods such as CGS [Sonne89] and transpose-free quasi-minimal residual (TFQMR) [Freund91]. The performance (in terms of the number of matrix-vector multiplies and preconditioner solves) for these methods is similar to the performance for GMRES(6). The restarting length 6 gives a storage requirement for GMRES, which is comparable to those of CGS and TFQMR.

Each preconditioner solve requires quite a large number of arithmetic operations. Hence, we compare the performance obtained when using the semicirculant preconditioner to the performance without preconditioner. In Fig. 9 we show the speedup in CPU time obtained on

a Sun-4/470 using the semicirculant preconditioner combined with the GMRES(6) iteration compared to the unpreconditioned iteration. Again, problems of size $2^p \times \frac{15}{16}2^p$, $p = 4, \ldots, 9$, are studied. It is clear that the speedup is large for large problems. Note that the speedup is greater than one also for small problems. Heuristically, we expect the speedup to grow as $\mathcal{O}(m/\log_2 m)$, when $m = m_1 \approx m_2$ is large.

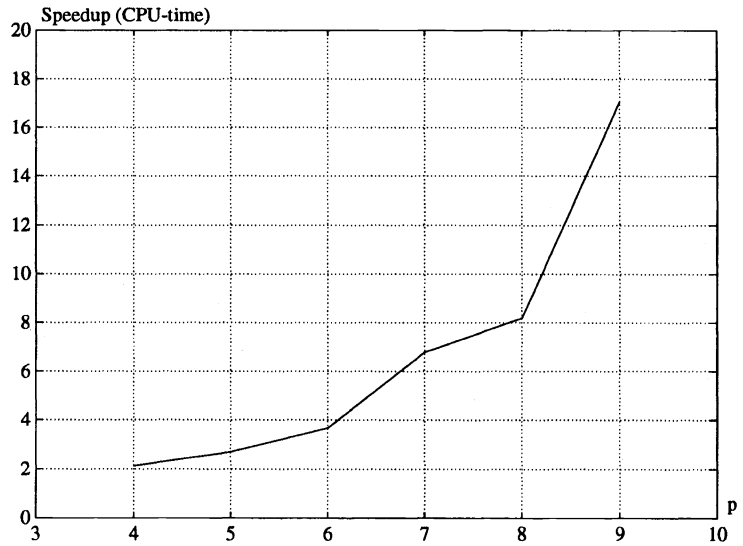


FIG. 9. *CPU time speedup as a function of problem size.*

10. Conclusions. We have used the restarted GMRES iteration combined with a semicirculant preconditioner for solving systems of equations arising from a time-independent hyperbolic PDE in two space dimensions. In the discretization a weak artificial viscosity is added in the x_1-direction. The numerical results indicate that the number of iterations required to solve the systems is independent of problem size. If no preconditioner is used, the number of iterations grows with problem size.

We have proved that, provided the ratio β of gridpoints in the x_2-direction to gridpoints in the x_1-direction is less than one, the spectrum of the preconditioned coefficient matrix asymptotically resides on two finite curve segments separated from zero. The spectrum of the unpreconditioned system asymptotically grows linearly with the number of gridpoints in the x_1-direction. Also, we have demonstrated that the eigenvectors of the preconditioned system are "improved" compared to the unpreconditioned system. The condition number of the eigenvector matrix is dramatically reduced. For practical computations, we have found that the convergence for the GMRES iteration is governed by a bound derived only from properties of the spectrum of the preconditioned coefficient matrix.

REFERENCES

[BaSw91] D. H. BAILEY AND P. N. SWARZTRAUBER, *The fractional Fourier transform and applications*, SIAM Rev., 33 (1991), pp. 389–404.

[RChan89] R. H. CHAN, *Circulant preconditioners for Hermitian Toeplitz systems*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 542–550.

[RChJin91] R. H. CHAN AND X.-Q. JIN, *Circulant and skew-circulant preconditioners for skew-Hermitian type Toeplitz systems*, BIT, 31 (1991), pp. 632–646.

[RChSt89] R. H. CHAN AND G. STRANG, *Toeplitz equations by conjugate gradients with circulant preconditioner*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 104–119.

[TChan88] T. F. CHAN, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.

[TChOl92] T. F. CHAN AND J. A. OLKIN, *Circulant preconditioners for Toeplitz-block matrices*, Report CAM 92-11, Dept. of Mathematics, Univ. of California at Los Angeles, 1992.

[Freund91] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, Tech. Report 91.18, RIACS, NASA Ames Research Center, Moffett Field, CA, 1991.

[FrGoNa91] R. W. FREUND, G. H. GOLUB, AND N. M. NACHTIGAL, *Iterative solution of linear systems*, Tech. Report 91.21, RIACS, NASA Ames Research Center, Moffett Field, CA, 1991.

[Holm93] S. HOLMGREN, *Fast Solvers for First-order PDE*, Ph.D. thesis, Dept. of Scientific Computing, Uppsala University, Uppsala, Sweden, 1993.

[HoOtto91] S. HOLMGREN AND K. OTTO, *Iterative solution methods and preconditioners for block-tridiagonal systems of equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 863–886.

[Otto92] K. OTTO, *Analysis of preconditioners for hyperbolic PDE*, Report No. 147, Dept. of Scientific Computing, Uppsala University, Uppsala, Sweden, 1992.

[Otto93] ———, *Construction and Analysis of Preconditioners for First-order PDE*, Ph.D. thesis, Dept. of Scientific Computing, Uppsala University, Uppsala, Sweden, 1993.

[Saad81] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105–126.

[SaadSch86] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[Sonne89] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[Tyr92] E. E. TYRTYSHNIKOV, *Optimal and superoptimal circulant preconditioners*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 459–473.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 408–426, March 1994

011

ON ADAPTIVE WEIGHTED POLYNOMIAL PRECONDITIONING FOR HERMITIAN POSITIVE DEFINITE MATRICES*

BERND FISCHER† AND ROLAND W. FREUND‡

Abstract. The conjugate gradient algorithm for solving Hermitian positive definite linear systems is usually combined with preconditioning in order to speed up convergence. In recent years, there has been a revival of polynomial preconditioning, motivated by the attractive features of the method on modern architectures. Standard techniques for choosing the preconditioning polynomial are based only on bounds for the extreme eigenvalues. Here a different approach is proposed that aims at adapting the preconditioner to the eigenvalue distribution of the coefficient matrix. The technique is based on the observation that good estimates for the eigenvalue distribution can be derived after only a few steps of the Lanczos process. This information is then used to construct a weight function for a suitable Chebyshev approximation problem. The solution of this problem yields the polynomial preconditioner. In particular, polynomial preconditioners associated with Bernstein–Szegö weights are studied. Results of numerical experiments are reported.

Key words. linear systems, Hermitian positive definite matrices, conjugate gradient algorithm, polynomial preconditioning, Chebyshev approximation problem, Bernstein–Szegö weights

AMS subject classifications. 65F10, 41A10, 65N22

1. Introduction.

One of the most powerful iterative schemes for solving Hermitian positive definite linear systems

$$Ax = b \tag{1.1}$$

is the conjugate gradient (CG) algorithm of Hestenes and Stiefel [16], especially when it is combined with preconditioning [5]. In recent years, there has been much interest in polynomial preconditioning. The basic idea is as follows: instead of solving the original system (1.1) by the CG algorithm, the CG iteration is applied either to

$$\psi(A)Ax = \psi(A)b \tag{1.2}$$

(left preconditioning) or to

$$A\psi(A)y = b, \qquad x = \psi(A)y \tag{1.3}$$

(right preconditioning). Here ψ is a suitably chosen polynomial of small degree. Moreover, it is required that none of the zeros of ψ coincides with an eigenvalue of A. This guarantees that the preconditioned systems (1.2) and (1.3) are both equivalent to (1.1).

Polynomial preconditioning goes back to the 1950s. It seems that Lanczos [21] was the first to mention the idea; interestingly, his paper is never referenced. Stiefel [28] used polynomial preconditioning techniques to accelerate eigenvalue computations. Rutishauser [25] proposed an inner-outer iteration process, with CG as the outer iteration and the Chebyshev semi-iterative method [14] as the inner recursion. The motivation for his approach was to reduce roundoff in the classical CG algorithm. In the 1980s, starting with the work of Johnson,

*Received by the editors May 18, 1992; accepted for publication (in revised form) January 7, 1993.

†Institut für Angewandte Mathematik, Universität Hamburg, Bundesstrasse 55, D-2000 Hamburg 13, Germany (na.fischer@na-net.ornl.gov).

‡AT&T Bell Laboratories, Room 2C-420, 600 Mountain Avenue, Murray Hill, New Jersey 07974-0636 (freund@research.att.com). The work of this author was supported by Cooperative Agreement NCC 2-387 between the National Aeronautics and Space Administration and the Universities Space Research Association while this author was in residence at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center, Moffett Field, California 94035.

Micchelli, and Paul [18], there has been a revival of Rutishauser's method and polynomial preconditioning in general; see [2], [9], [23], [26], and the references given there. The main reason for this renewed interest is that polynomial preconditioning is an attractive technique on vector and parallel computers; see, e.g., [27]. Each CG iteration involves the computation of inner products, which constitutes a bottleneck on many modern architectures. Polynomial preconditioning reduces the number of CG iterations and thus the total number of inner products, since the preconditioning step itself, i.e., multiplication by the preconditioner $\psi(A)$, does not require inner products.

For the Chebyshev iteration in Rutishauser's method, estimates α and β for the smallest and largest eigenvalues of A are needed. Often, good upper bounds β can be obtained easily, using simple techniques such as Gershgorin's theorem [26]. It is far more difficult to estimate the smallest eigenvalue. Saad [26] proposed a polynomial preconditioning technique that only requires an upper bound for the largest eigenvalue, while the trivial bound $\alpha = 0$ is used for the smallest eigenvalue of the positive definite matrix A. His technique is based on least-squares polynomials associated with the family of Jacobi weights [29]. Ideally, one would choose the weight function in the Jacobi class such that CG for the preconditioned systems (1.2) and (1.3) converges as fast as possible. However, the problem of finding such an optimal Jacobi weight is not addressed in [26]. Another way to avoid the computation of α and β is to use the CG polynomial itself as preconditioner. This approach was investigated by O'Leary [23]. The disadvantage of this technique is that the preconditioned system is not guaranteed to be positive definite. Ashby, Manteuffel, and Otto [2] demonstrated in a variety of numerical examples that the effectiveness of Chebyshev and least-squares polynomial preconditioners depends on the eigenvalue distribution of the coefficient matrix A.

Another option is to construct polynomial preconditioners via weighted Chebyshev approximation problems. This was proposed by Freund [9] who also suggested a heuristic for adapting the weight function to the eigenvalue distribution of A. The technique exploits the observation that eigenvalue distributions of Hermitian matrices can be surprisingly well estimated, using only a few steps of the Lanczos method. Actually, spectral estimation based on the Lanczos process is a widely used technique in applications; see, e.g., [24] and the references given there. The obtained estimated eigenvalue distribution is then used to construct a weight function, and finally the polynomial preconditioner is computed by solving the corresponding approximation problem.

In this paper, we further study polynomial preconditioning based on weighted Chebyshev approximation problems. In particular, we investigate the use of Bernstein–Szegö weights [29]. For such weights, the solutions of the associated approximation problems are known explicitly. Therefore, the construction of preconditioning polynomials based on Bernstein–Szegö weights does not involve the numerical solution of an approximation problem.

The remainder of this paper is organized as follows. In §2, we recall some basic properties of CG, and we discuss Chebyshev polynomial preconditioning. In §3, we present our approach to polynomial preconditioning based on weighted Chebyshev approximation problems, and we propose a procedure for obtaining a suitable weight function from the Lanczos process. This technique involves the construction of a monotone interpolant. In §4, we briefly describe a procedure for monotone piecewise cubic interpolation. In §5, we consider polynomial preconditioners based on Bernstein–Szegö weights. In §6, results of numerical experiments are reported. Finally, in §7, we make some concluding remarks.

Throughout the paper, it is always assumed that A in (1.1) is a Hermitian positive definite $N \times N$ matrix, with real or complex entries. As usual, M^H denotes the conjugate transpose of a matrix M, and the vector norm $\|x\| = \sqrt{x^H x}$ is the Euclidean norm. Finally, we denote by

$$\mathcal{P}_n := \{\varphi(\lambda) \equiv \sigma_0 + \sigma_1\lambda + \cdots + \sigma_n\lambda^n \mid \sigma_0, \sigma_1, \ldots, \sigma_n \in \mathbb{C}\}$$

the set of all complex polynomials of degree at most n.

2. CG and Chebyshev polynomial preconditioning. In this section, we collect some basic facts about CG, and we review Chebyshev polynomial preconditioning.

2.1. The CG algorithm. Let $x_0 \in \mathbb{C}^N$ be any initial guess for the solution of (1.1), and let $r_0 := b - Ax_0$ be the associated residual vector. The CG algorithm generates iterates of the form

$$x_n = x_0 + \chi_n(A)r_0, \quad \text{where } \chi_n \in \mathcal{P}_{n-1}, \quad n = 1, 2, \ldots . \tag{2.1}$$

The corresponding residual vectors are given by

$$r_n = \phi_n(A)r_0, \quad \text{where } \phi_n(\lambda) \equiv 1 - \lambda\chi_n(\lambda). \tag{2.2}$$

In exact arithmetic, the CG algorithm terminates after a finite number of steps with the exact solution $x_L = A^{-1}b$ of (1.1). In the sequel, L always denotes this termination index. We remark that L is just the minimal number of components in any expansion of r_0 into orthonormal eigenvectors v_j of A, and thus we have

$$r_0 = \sum_{j=1}^{L} \sigma_j v_j, \quad \text{where } \sigma_j \neq 0 \quad \text{for all } j. \tag{2.3}$$

In particular, $L \leq N$. In the following, we always assume that the vectors v_j have been scaled such that $\sigma_j > 0$ in (2.3). Furthermore, we denote by λ_j the eigenvalues corresponding to the eigenvectors v_j in (2.3), and thus we have

$$Av_j = \lambda_j v_j. \tag{2.4}$$

Since the number of terms in the representation (2.3) is minimal, it follows that the λ_j's are distinct. From now on, we assume that they are numbered in increasing order:

$$\lambda_1 < \lambda_2 < \cdots < \lambda_L. \tag{2.5}$$

The CG iterates are optimal, in the sense that $r_n^H A^{-1} r_n$ is minimal for all possible iterates of the form (2.1). This minimization property can be shown to be equivalent to the following orthogonality relations:

$$r_n^H r_k = 0 \quad \text{for all} \quad n, k = 0, 1, \ldots, L, \ n \neq k. \tag{2.6}$$

Using (2.2), (2.3), and (2.4), we can rewrite (2.6) in the form

$$\langle \phi_n, \phi_k \rangle = 0 \quad \text{for all } n, k = 0, 1, \ldots, L, \ n \neq k,$$

where

$$\langle \phi_n, \phi_k \rangle := r_0^H \phi_n(A)\phi_k(A)r_0 = \sum_{j=1}^{L} \sigma_j^2 \phi_n(\lambda_j)\phi_k(\lambda_j). \tag{2.7}$$

We remark that—even for complex Hermitian A—all CG polynomials ϕ_n have real coefficients, and hence no complex conjugation of ϕ_n is needed in (2.7).

It is well known (see, e.g., [13]) that the CG method and the Lanczos process [20] for tridiagonalizing A are mathematically equivalent. Using this connection, it can be shown that, for each fixed $n \in \{1, 2, \ldots, L\}$, the residual polynomials $\phi_0, \phi_1, \ldots, \phi_n$ are also orthogonal with respect to the inner product induced by J_n. Here, J_n denotes the $n \times n$ tridiagonal matrix generated by the first n steps of the Lanczos process. Before we state these orthogonality relations, we introduce some more notation. Let $n \in \{1, 2, \ldots, L\}$ be fixed, and let $\theta_j := \theta_j^{(n)}$, $j = 1, 2, \ldots, n$, denote the eigenvalues of J_n. The θ_j's are called the nth Ritz values of A. They are all distinct, and we assume that they are numbered in increasing order:

$$\theta_1 < \theta_2 < \cdots < \theta_n. \tag{2.8}$$

Let s_j be a set of corresponding orthonormal eigenvectors, i.e.,

$$J_n s_j = \theta_j s_j, \qquad \|s_j\| = 1.$$

Moreover, we assume that the s_j's are normalized such that their first components, $\tau_j := e_1^H s_j$, are all real and nonnegative. Here, e_1 denotes the first unit vector in $\mathbb{R}^n$. With these notations, the orthogonality relations can be stated as follows:

$$\langle \phi_i, \phi_k \rangle_n = 0 \quad \text{for all } i, k = 0, 1, \ldots, n, \ i \neq k,$$

where

$$\langle \phi_i, \phi_k \rangle_n := e_1^H \phi_i(J_n)\phi_k(J_n)e_1 = \sum_{j=1}^{n} \tau_j^2 \phi_i(\theta_j)\phi_k(\theta_j). \tag{2.9}$$

From now on, we assume that α and β are given real numbers satisfying

$$0 \leq \alpha \leq \lambda_1 \quad \text{and} \quad \lambda_L \leq \beta. \tag{2.10}$$

The condition (2.10) guarantees that the eigenvalues (2.5) of A and all Ritz values (2.8) are contained in the interval $[\alpha, \beta]$. In the sequel, we will make frequent use of the linear transformation

$$t := \ell(\lambda) \equiv \frac{2\lambda - (\beta + \alpha)}{\beta - \alpha}, \tag{2.11}$$

which maps $[\alpha, \beta]$ onto the unit interval $[-1, 1]$. In particular, it will turn out to be convenient to rewrite, by means of (2.11), the inner products (2.7) and (2.9) in terms of $[-1, 1]$. We set $\hat{t}_j := \ell(\lambda_j)$ and $t_j := \ell(\theta_j)$. From (2.7) and (2.9), we obtain

$$\begin{aligned} \langle \phi_n, \phi_k \rangle &= \sum_{j=1}^{L} \sigma_j^2 \phi_n(\lambda_j)\phi_k(\lambda_j) \\ &= \sum_{j=1}^{L} \sigma_j^2 \phi_n(\ell^{-1}(\hat{t}_j))\phi_k(\ell^{-1}(\hat{t}_j)) =: \int_{\mathbb{R}} \phi_n(\ell^{-1}(t))\phi_k(\ell^{-1}(t))\, d\sigma(t) \end{aligned} \tag{2.12}$$

and

$$\begin{aligned} \langle \phi_i, \phi_k \rangle_n &= \sum_{j=1}^{n} \tau_j^2 \phi_i(\theta_j)\phi_k(\theta_j) \\ &= \sum_{j=1}^{n} \tau_j^2 \phi_i(\ell^{-1}(t_j))\phi_k(\ell^{-1}(t_j)) =: \int_{\mathbb{R}} \phi_i(\ell^{-1}(t))\phi_k(\ell^{-1}(t))\, d\tau(t), \end{aligned} \tag{2.13}$$

respectively. The distribution functions $\sigma(t)$ and $\tau(t)$ in the Riemann–Stieltjes integrals defined in (2.12) and (2.13) are step functions, and they are given by

$$\sigma(t) \equiv \sum_{j=1}^{L} \sigma_j^2 H(t - \hat{t}_j), \quad \text{where } H(t) \equiv \begin{cases} 1 & \text{for } t \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{2.14}$$

and

$$\tau(t) \equiv \sum_{j=1}^{n} \tau_j^2 H(t - t_j), \tag{2.15}$$

respectively.

We now turn to polynomial preconditioning. For simplicity, we will focus only on right preconditioning (1.3), but all statements in this paper essentially remain true for left preconditioning (1.2). From now on, it is always assumed that $p > 1$ is a given integer, and we consider preconditioning polynomials $\psi \in \mathcal{P}_{p-1}$ in (1.3).

2.2. Chebyshev polynomial preconditioning. The standard approach for the design of preconditioners is to choose the polynomial ψ in (1.3) such that $A\psi(A)$ is, in some sense, as close as possible to the identity matrix I. For instance, one could attempt to minimize the Euclidean norm $\|I - A\psi(A)\|$. However, the solution of this problem would require the knowledge of all eigenvalues of A. Therefore, one usually substitutes for the spectrum of A an interval $[\alpha, \beta]$ that is known to contain all eigenvalues of A. In addition, it is required that $\alpha > 0$. This approach then leads to a Chebyshev approximation problem on the interval $[\alpha, \beta]$. After rewriting, by means of the transformation $\ell(\lambda)$ in (2.11), this problem in terms of the unit interval $[-1, 1]$, we obtain the Chebyshev approximation problem

$$\min_{\varphi \in \mathcal{P}_p:\, \varphi(\zeta)=1} \; \max_{t \in [-1,1]} |\varphi(t)|. \tag{2.16}$$

Here, $\zeta := \ell(0)$ and $\varphi(\ell(\lambda)) \equiv 1 - \lambda\psi(\lambda)$. It is well known that the optimal solution φ_p of (2.16) is just the suitably scaled pth Chebyshev polynomial of the first kind, T_p, and we have

$$\varphi_p(t) \equiv \frac{T_p(t)}{T_p(\zeta)}. \tag{2.17}$$

Note that the spectrum of the preconditioned matrix $A\psi_p(A)$ is contained in the interval

$$[\alpha_p, \beta_p], \quad \text{where } \alpha_p := 1 - \frac{1}{|T_p(\zeta)|} \quad \text{and} \quad \beta_p := 1 + \frac{1}{|T_p(\zeta)|}. \tag{2.18}$$

As in (2.2), let $r_{np} = \phi_{np}(A)r_0$ denote the residual vector obtained after np steps of CG applied to the original system (1.1). Similarly, let $r_n^{(\mathrm{P})} = \phi_n^{(\mathrm{P})}(A\psi_p(A))r_0$ be the nth residual vector generated by CG applied to the preconditioned system (1.3). In view of the optimality properties of the CG algorithm, we have

$$\begin{aligned} r_{np}^H A^{-1} r_{np} &= \min_{\phi \in \mathcal{P}_{np}:\, \phi(0)=1} (\phi(A)r_0)^H A^{-1} (\phi(A)r_0) \\ &\leq (\phi_n^{(\mathrm{P})}(A\psi_p(A))r_0)^H A^{-1} (\phi_n^{(\mathrm{P})}(A\psi_p(A))r_0) = (r_n^{(\mathrm{P})})^H A^{-1} r_n^{(\mathrm{P})}. \end{aligned}$$

As a result, n steps of CG applied to (1.3) can at best give the same residual vector as np steps of CG applied to (1.1). Therefore, Chebyshev polynomial preconditioning is indeed best possible, provided that the residual polynomials ϕ_{np} and $\phi_n^{(\mathrm{P})}$ satisfy

$$\phi_{np}(\lambda) \equiv \phi_n^{(\mathrm{P})}(\lambda\psi_p(\lambda)). \tag{2.19}$$

Next, we show that (2.19) is fulfilled if the distribution function $\sigma(t)$ defined in (2.14) corresponds to the worst-case distribution for the interval $[-1, 1]$.

The function $\sigma(t)$ is a step function with jumps at the translated eigenvalues $\hat{t}_j = \ell(\lambda_j)$ of A. For our discussion, we treat $\sigma(t)$ as a continuous function, and we denote by

$$\delta(t) \equiv \frac{d\sigma(t)}{dt} \tag{2.20}$$

the associated density function. Actually, since the dimension N of A (and thus L) is usually large, the step function "looks like" a continuous function; see Fig. 3.1 for an example. From potential theory, it is known that the worst-case distribution is just the equilibrium distribution, and for the case of the unit interval, it is given by

$$\sigma^{(\mathrm{E})}(t) \equiv \arcsin t, \qquad t \in [-1, 1]. \tag{2.21}$$

Furthermore, the orthogonal polynomials with respect to the corresponding inner product are the Chebyshev polynomials of the first kind, and we have, for all integers n, $k \geq 0$, $n \neq k$,

$$\int_{-1}^{1} T_n(t)T_k(t)\delta^{(\mathrm{E})}(t)\,dt = 0, \quad \text{where } \delta^{(\mathrm{E})}(t) \equiv \frac{1}{\sqrt{1-t^2}}. \tag{2.22}$$

We remark that (2.22) can be evaluated by means of Gaussian quadrature. This gives

$$\int_{-1}^{1} T_n(t)T_k(t)\delta^{(\mathrm{E})}(t)\,dt = \sum_{j=1}^{L} \hat{\sigma}_j^2 T_n(\hat{t}_j)T_k(\hat{t}_j), \qquad n, k < L.$$

In other words, for the distribution function defined by (2.21) the CG residuals correspond to Chebyshev polynomials. We remark that in this case the standard error bounds for the CG iterates are sharp; see [15]. Finally, we show that for the worst-case distribution (2.21) the relation (2.19) is satisfied, and hence Chebyshev polynomial preconditioning is optimal in this case.

LEMMA 2.1. *Let ℓ_p denote the linear mapping that maps $[\alpha_p, \beta_p]$ (cf. (2.18)) onto the unit interval $[-1, 1]$, and let $\zeta_p := \ell_p(0)$. Let $\phi_{np}(\lambda) \equiv T_{np}(\ell(\lambda))/T_{np}(\zeta)$ and $\phi_n^{(\mathrm{P})}(\lambda) \equiv T_n(\ell_p(\lambda))/T_n(\zeta_p)$ be the shifted and normalized Chebyshev polynomial on $[\alpha, \beta]$ and $[\alpha_p, \beta_p]$, respectively. Then the identity* (2.19) *is satisfied.*

Proof. From (2.18) and (2.17) one readily obtains

$$\phi_n^{(\mathrm{P})}(\lambda\psi_p(\lambda)) \equiv \frac{T_n(T_p(\ell(\lambda)))}{T_n(T_p(\zeta))}.$$

Equation (2.19) then follows from the well-known identity $T_{np}(t) \equiv T_n(T_p(t))$. □

3. Weighted polynomial preconditioning. As discussed in the previous section, Chebyshev polynomial preconditioning is optimal for matrices A, for which the function $\sigma(t)$ in (2.14) is the worst-case distribution. However, in practice, linear systems, especially those arising in the numerical treatment of partial differential equations, have eigenvalue distributions that are far from the worst case. For those, Chebyshev polynomial preconditioning is not optimal. Furthermore, it is known [1] that repeated application of Chebyshev polynomials will transform any given distribution into the worst-case distribution. This behavior is also reflected in Chebyshev polynomial preconditioning. If A has a favorable eigenvalue distribution, then the eigenvalue distribution of the preconditioned system is usually much closer to the worst case.

In this section, we propose a heuristic for adapting the preconditioning polynomial to the actual eigenvalue distribution of A.

3.1. Weighted Chebyshev approximation problems. Instead of (2.16), we now consider weighted Chebyshev approximation problems of the form

$$\min_{\varphi\in\mathcal{P}_p:\,\varphi(\zeta)=1}\ \max_{t\in[-1,1]}\ |w(t)\varphi(t)|, \quad \text{where } \zeta = \ell(0). \tag{3.1}$$

Here, w is a continuous weight function on the unit interval $[-1, 1]$, and it is always assumed that $w(t) > 0$ on the open interval $(-1, 1)$. Standard results from approximation theory (see, e.g., [22]) guarantee that there exists a unique optimal polynomial φ_p for (3.1). In general, φ_p is not known explicitly, and one needs to solve (3.1) numerically, for example, using the Remez algorithm; see, e.g., [22], [12]. Note that, for $w(t) \equiv 1$, the problem (3.1) reduces to (2.16), and the scaled Chebyshev polynomial (2.17) is the solution of (3.1).

The idea now is to use—instead of the Chebyshev polynomial (2.17)—the optimal solution φ_p of (3.1) as a polynomial preconditioner, where the weight function w is chosen based on an estimate for the density δ of the eigenvalue distribution of A.

It remains to give a heuristic for the choice of the weight function w in (3.1). If the estimated density δ of A happens to be the worst-case density $\delta^{(\mathrm{E})}$ defined in (2.22), then the solution of (3.1) should yield the optimal preconditioner for the worst case. As shown in §2, Chebyshev polynomials are optimal in this case. Therefore, the weight function should be constructed such that $w(t) \equiv 1$ if $\delta(t) \equiv 1/\sqrt{1-t^2}$. Motivated by this consideration, we propose the choice

$$w(t) \equiv \sqrt{\delta(t)\sqrt{1-t^2}}, \tag{3.2}$$

where $\delta(t)$ is an estimate for the eigenvalue density defined in (2.20). We stress that the connection between w and δ in (3.2) is very natural. Indeed, it is known that, for certain families of densities δ, the weighted L_2-approximation problem

$$\min_{\varphi\in\mathcal{P}_p:\,\varphi(\zeta)=1}\ \int_{-1}^{1} |\varphi(t)|^2\delta(t)\,dt$$

and the weighted Chebyshev approximation problem (3.1) with w given by (3.2) have the same optimal solution φ_p. For example, this is true for the worst-case density $\delta^{(\mathrm{E})}$ and for the more general class of densities associated with Bernstein–Szegö polynomials; this was first observed by Bernstein [3].

3.2. Estimating the distribution function. Usually, a good estimate for the eigenvalue distribution σ in (2.14) can be derived from the quantities generated by relatively few steps of the Lanczos process. Actually, since the Lanczos algorithm is equivalent to CG, we can extract all necessary information from a few steps of the CG algorithm applied to the original system (1.1).

Suppose we run CG for n steps. The process then has generated the entries of the $n \times n$ tridiagonal Lanczos matrix J_n. The Ritz values θ_j for J_n and the first component τ_j of the corresponding orthonormal eigenvectors define the inner product $\langle\cdot,\cdot\rangle_n$ in (2.9). It can be shown that $\langle\cdot,\cdot\rangle_n$ and the inner product $\langle\cdot,\cdot\rangle$ in (2.7) have the same moments up to degree $2n-1$, i.e., for all real polynomials ϕ of degree at most $2n-1$, it holds that

$$\langle 1, \phi\rangle_n = \langle 1, \phi\rangle.$$

This condition implies, roughly speaking, that the distribution $\tau(t)$ of the Ritz values has to be close to the eigenvalue distribution $\sigma(t)$. This statement can be made more precise, in the sense of the following theorem due to Karlin and Shapley [19, Thm. 22.2].

THEOREM 3.1. *Let $\sigma(t)$ and $\tau(t)$ denote the distribution functions defined in* (2.14) *and in* (2.15), *respectively. Then the difference function $\sigma(t) - \tau(t)$, if not identically zero, has exactly $2n - 1$ sign changes in the interval* $[-1, 1]$.

Next, we illustrate this result for the case of Example 6.1 described in §6. We only consider the case $h = \frac{1}{64}$, which leads to a matrix A of order $N = 3844$. In Fig. 3.1, the "continuous-looking" curve is the eigenvalue distribution $\sigma(t)$ of A, while the step function is the Ritz-value distribution $\tau(t)$ obtained after $n = 10$ steps of the CG algorithm. Note that the horizontal and vertical steps of $\tau(t)$ intersect $\sigma(t)$ exactly 19 times in $[-1, 1]$, in accordance with Theorem 3.1.

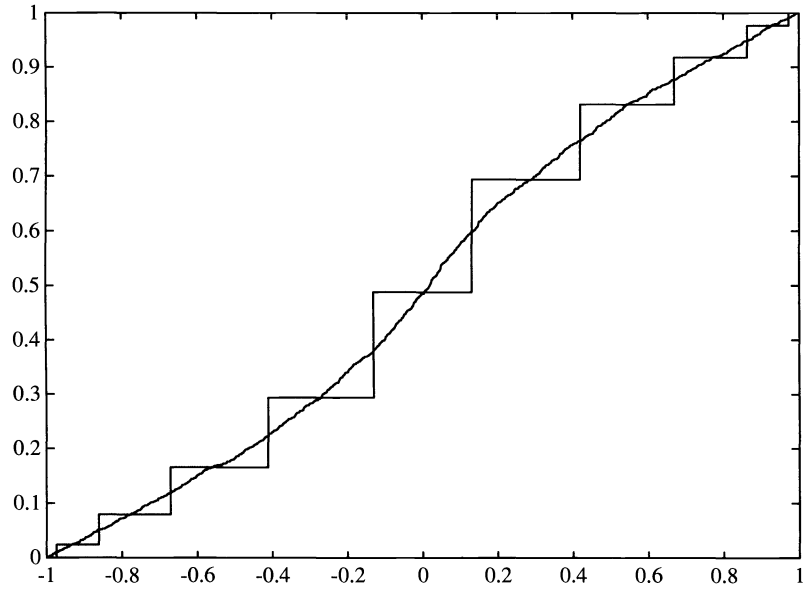


FIG. 3.1. *Distribution $\tau(t)$ of the Ritz values corresponding to $n = 10$ and eigenvalue distribution $\sigma(t)$ for Example* 6.1 *with $h = \frac{1}{64}$.*

We remark that, in Fig. 3.1, the midpoints of the vertical steps of $\tau(t)$ are very good approximations to points on the unknown curve $\sigma(t)$. This behavior is rather typical, and it was also observed by others [24, p. 6]. This suggests using interpolation at the vertical midpoints of $\tau(t)$ to construct an estimate, denoted by $f(t)$ in the sequel, for the true eigenvalue distribution $\sigma(t)$.

More precisely, we proceed as follows. First, we need to choose numbers α and β that satisfy (2.10). We stress that the lower bound α is only required to be nonnegative. In particular, the trivial choice $\alpha = 0$ is feasible, and this is one of the strengths of our approach. Of course, if there are better lower bounds for the smallest eigenvalue at hand, one should use these. However, we stress that overestimating the smallest eigenvalue may slow down the convergence rate. Typically, the largest Ritz value θ_n is a good approximation for the largest eigenvalue of A, and therefore we usually set $\beta := \theta_n$. This choice is not critical. Indeed, if there are a few eigenvalues larger than β, the preconditioning polynomial amplifies them, and the CG iteration will rapidly suppress components corresponding to these large eigenvalues.

Once α and β are chosen, we define the interpolating points by setting

$$t_j := \begin{cases} -1 & \text{for } j = 0, \\ \ell(\theta_j) & \text{for } j = 1, 2, \ldots, n-1, \\ 1 & \text{for } j = n, \end{cases}$$

and

$$\vartheta_j := \begin{cases} 0 & \text{for } j = 0, \\ \tau_j^2/2 + \sum_{i=1}^{j-1} \tau_i^2 & \text{for } j = 1, 2, \ldots, n-1, \\ 1 & \text{for } j = n. \end{cases}$$

Here, $\ell(\lambda)$ is again the linear map given by (2.11). Finally, the estimated distribution is then chosen as a monotone function $f \in C^1[-1, 1]$ satisfying

$$f(t_j) = \vartheta_j, \qquad j = 0, 1, \ldots, n. \tag{3.3}$$

4. Monotone piecewise cubic interpolation. One obvious choice for the interpolating function f is a monotone piecewise cubic interpolant. In this section, we briefly describe how to construct such a function. We follow the derivation of Fritsch and Carlson [11] and Fritsch and Butland [10].

Let $-1 = t_0 < t_1 < \cdots < t_n = 1$ and $0 = \vartheta_0 < \vartheta_1 < \cdots < \vartheta_n = 1$ be given. The goal is to construct a piecewise cubic function $f \in C^1[-1, 1]$ such that $f(t_i) = \vartheta_i$, $i = 0, 1, \ldots, n$, and f is monotone on $[-1, 1]$. To this end, one expresses f, on any subinterval $[t_i, t_{i+1}]$, in terms of the derivatives $d_i = f'(t_i)$, $i = 0, 1, \ldots, n$, as described in [11]. This leads to the representation

$$f(t) \equiv \left(\frac{d_i + d_{i+1} - 2\Delta_i}{h_i^2}\right)(t - t_i)^3 + \left(\frac{-2d_i - d_{i+1} + 3\Delta_i}{h_i}\right)(t - t_i)^2 + d_i(t - t_i) + \vartheta_i,$$

where $h_i := t_{i+1} - t_i$ and $\Delta_i := (\vartheta_{i+1} - \vartheta_i)/h_i$, $i = 0, 1, \ldots, n-1$.

By construction, any choice of the free parameters d_i leads to a function $f \in C^1[-1, 1]$ that fulfills (3.3). The remaining step is to adjust the d_i's such that f is monotone on $[-1, 1]$.

Note that the d_i's are not uniquely determined, and various choices have been discussed in the literature. Here, we use a formula proposed by Brodlie [4] and Fritsch and Butland [10]:

$$d_i := \begin{cases} \dfrac{\Delta_{i-1}\Delta_i}{\xi_i\Delta_i + (1 - \xi_i)\Delta_{i-1}} & \text{for } \Delta_{i-1}\Delta_i > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad i = 1, 2, \ldots, n-1, \tag{4.1}$$

where $\xi_i := (h_{i-1} + 2h_i)/(3(h_{i-1} + h_i))$. In addition to (4.1), we still need to choose the boundary conditions d_0 and d_n. Since, in our case, no information about the derivatives at the endpoints is available, we select a weak version of the so-called "not-a-knot" condition; see De Boor [6, p. 54]. Here one chooses d_0 and d_n such that f is twice continuously differentiable on $[t_0, t_2)$ and $(t_{n-2}, t_n]$, respectively. One obtains

$$\begin{aligned} d_0 &= \frac{h_0}{h_1}\big(3\Delta_1 - (2d_1 + d_2)\big) + 3\Delta_0 - 2d_1, \\ d_n &= \frac{h_{n-1}}{h_{n-2}}\big(3\Delta_{n-2} - (2d_{n-1} + d_{n-2})\big) + 3\Delta_{n-1} - 2d_{n-1}. \end{aligned} \tag{4.2}$$

However, this special choice of d_0 and d_n does not necessarily produce a monotone f on the subintervals $[t_0, t_1]$ and $[t_{n-1}, t_n]$, respectively. Note that the additional requirements $d_0 \in [0, 3\Delta_0]$ and $d_n \in [0, 3\Delta_{n-1}]$ will lead to a monotone function. Thus, if, for example, d_0 (computed by (4.2)) turns out to be negative or bigger than $3\Delta_0$, we simply set $d_0 = 0$ or $d_0 = 3\Delta_0$, respectively.

We would like to mention that FORTRAN codes for the described procedures are available in NETLIB (PCHIP package).

Finally, in analogy to (2.20), we set

$$\delta(t) \equiv f'(t), \tag{4.3}$$

where f is the computed monotone interpolant, and we define the weight function $w(t) \equiv w(t; f)$ by (3.2). Note that $w(t; f)$ is continuous on $[-1, 1]$. The desired polynomial preconditioner is then obtained by solving the resulting weighted Chebyshev approximation problem (3.1) numerically, using the Remez algorithm. One might ask why we did not use the simpler approach of constructing f as a monotone piecewise linear or quadratic interpolant. In both cases, the resulting weight function $w(t; f)$ would no longer be continuous in general. As a result, the standard Remez algorithm, which requires that the weight function is continuous, could not be used for solving (3.1). Therefore, we did not pursue the use of linear or quadratic interpolants.

5. Bernstein–Szegö weight functions. The approach described so far consists of two steps: first, one approximates the distribution function, and second, the polynomial preconditioner is obtained by solving the resulting approximation problem (3.1) numerically. In this section, we propose a different procedure that eliminates the second step. The idea is to restrict the weight functions to a class for which the solution of the Chebyshev problem (3.1) is known explicitly. We consider three classes of weight functions that fulfill this requirement.

Let ρ_k be any real polynomial of degree k with $\rho_k(t) > 0$ on $[-1, 1]$, and define

$$s_0(t) \equiv 1, \quad s_{1/2}(t) \equiv \sqrt{1+t}, \quad s_1(t) \equiv \sqrt{1-t^2}.$$

Then (3.1) can be solved explicitly for the so-called Bernstein–Szegö weight functions

$$w_j(t) \equiv \frac{s_j(t)}{\sqrt{\rho_k(t)}}, \qquad j \in \left\{0, \frac{1}{2}, 1\right\};$$

see Szegö [29], Freund [8], and the references given therein. More precisely, the solution φ_p of (3.1) with respect to w_j, $j \in \{0, \frac{1}{2}, 1\}$, is explicitly known for $p \geq p_j$, where

$$p_j = \begin{cases} 0 & \text{if } j = 1 \text{ and } k = 0, \\ \lfloor (k+1)/2 - j \rfloor & \text{otherwise.} \end{cases}$$

For convenience, in the sequel, we allow ρ_k to have simple zeros at the endpoints ± 1, i.e., the cases $j = 0, \frac{1}{2}$ are now included in the case $j = 1$. Therefore, we will only investigate weight functions of the form

$$w(t) \equiv w_1(t) \equiv \frac{\sqrt{1-t^2}}{\sqrt{\rho_k(t)}}. \tag{5.1}$$

From (4.3), (3.2), and (5.1), it follows that

$$f(x) \equiv \int_{-1}^{x} \delta(t)\, dt \equiv \int_{-1}^{x} \frac{w(t)^2}{\sqrt{1-t^2}}\, dt \equiv \int_{-1}^{x} \frac{\sqrt{1-t^2}}{\rho_k(t)}\, dt. \tag{5.2}$$

It remains to choose the polynomial ρ_k in (5.2) such that f fulfills the interpolatory conditions $f(t_j) = \vartheta_j$ stated in (3.3). Note that any f defined by (5.2) is "automatically" monotone on $[-1, 1]$, and that $f(-1) = 0$.

It turns out to be advantageous to express ρ_k in terms of its zeros:

$$\rho_k(t) \equiv a_{m+1} \prod_{j=1}^{m} (t - a_j) \prod_{j=1}^{l} (t - z_j)(t - \overline{z}_j), \qquad k = m + 2l.$$

Here, a_j, $j = 1, 2, \ldots, m$, are the real zeros, and $z_j = x_j + iy_j$, $y_j > 0$, $j = 1, 2, \ldots, l$, are the complex zeros in the upper half-plane. The partial-fractions expansion of (5.2) reads

$$\begin{aligned} f(x) &\equiv \frac{1}{a_{m+1}} \left(\sum_{j=1}^{m} A_j \int_{-1}^{x} \frac{\sqrt{1 - t^2}}{t - a_j} \, dt + \sum_{j=1}^{l} \int_{-1}^{x} \frac{(B_j t + C_j)\sqrt{1 - t^2}}{(t - z_j)(t - \overline{z}_j)} \, dt \right) \\ &\equiv F(x; a_1, \ldots, a_{m+1}, x_1, \ldots, x_l, y_1, \ldots, y_l), \end{aligned} \tag{5.3}$$

where

$$A_j = \frac{1}{\rho_k'(a_j)}, \qquad j = 1, 2, \ldots, m,$$

$$B_j t + C_j \equiv \frac{\rho_k'(\overline{z}_j)(t - \overline{z}_j) + \rho_k'(z_j)(t - z_j)}{\rho_k'(z_j)\rho_k'(\overline{z}_j)}, \qquad j = 1, 2, \ldots, l.$$

It is readily verified that all integrals in (5.3) have an explicit antiderivative. We omit these routine, but somewhat lengthy, calculations.

Now the interpolation conditions (3.3), together with the positivity requirement

$$\rho_k > 0 \quad \text{on} \quad (-1, 1), \tag{5.4}$$

lead to the following nonlinear interpolation problem with constraints:

Find real numbers $a_1, \ldots, a_{m+1}, x_1, \ldots, x_l, y_1, \ldots, y_l$ such that

$$\begin{aligned} F(t_j; a_1, \ldots, a_{m+1}, x_1, \ldots, x_l, y_1, \ldots, y_l) &= \vartheta_j, \qquad j = 1, 2, \ldots, n, \\ |a_j| &\geq 1, \qquad j = 1, 2, \ldots, m, \\ \operatorname{sgn}(a_{m+1}) &= (-1)^m \prod_{j=1}^{m} \operatorname{sgn}(a_j), \\ y_j &> 0, \qquad j = 1, 2, \ldots, l. \end{aligned} \tag{5.5}$$

Apart from the sign conditions, the problem (5.5) can be viewed as a nonlinear system of n equations in $k + 1 = m + 1 + 2l$ unknowns. Therefore, a natural choice of the polynomial degree is $k = n - 1$. Also, we would like to mention that (5.5) does not address the problem of the possible occurrence of multiple zeros at ± 1. However, in our experiments, we never encountered such multiple zeros.

The success of any nonlinear solver applied to (5.5) depends strongly on good starting values. We now describe a linear procedure for computing starting values. In view of (5.2), we have

$$\rho_k(x) f'(x) \equiv \sqrt{1 - x^2}$$

and, consequently,

$$(\rho_k(x) f(x))' \equiv \rho_k(x)' f(x) + \sqrt{1-x^2}. \tag{5.6}$$

By integrating (5.6) and using $f(-1) = 0$, we obtain the identity

$$\rho_k(x) f(x) \equiv \int_{-1}^{x} \rho_k(t)' f(t)\, dt + g(x), \tag{5.7}$$

where $g(x) \equiv \int_{-1}^{x} \sqrt{1-t^2}\, dt$. Now we parametrize ρ_k in the form

$$\rho_k(x) \equiv \sum_{j=0}^{k} b_j x^j, \quad \text{with } b_j \in \mathbb{R}, \tag{5.8}$$

and we set

$$\gamma_j(x;\, f) \equiv \int_{-1}^{x} t^{j-1} f(t)\, dt.$$

Then, the relation (5.7) can be rewritten as follows:

$$\sum_{j=0}^{k} b_j \left(x^j f(x) - j\gamma_j(x;\, f)\right) \equiv g(x). \tag{5.9}$$

We consider (5.9) only at the interpolation points $x = t_j$, $j = 1, 2, \ldots, n$. Together with the positivity condition (5.4), we obtain the following semi-infinite problem for computing the unknown coefficients $b_0, b_1, \ldots, b_n$ in (5.8):

Find real numbers b_j, $j = 0, 1, \ldots, k$, such that

$$\begin{aligned} G(t_i;\, b_0, \ldots, b_k) &= 0, \qquad i = 1, 2, \ldots, n, \\ \sum_{j=0}^{k} b_j x^j &> 0, \qquad x \in (-1, 1), \end{aligned} \tag{5.10}$$

where

$$G(t_i;\, b_0, \ldots, b_k) := \sum_{j=0}^{k} b_j \left(t_i^j \vartheta_i - j\gamma_j(t_i;\, f)\right) - g(t_i).$$

Note that problem (5.10) is linear, but there are infinitely many constraints. In addition, we must evaluate the integrals γ_j, which involve the unknown function f. To this end, we approximate f by a monotone piecewise cubic function S, as described in §4. Clearly, the resulting integrals $\gamma_j(x;\, S)$ are easy to compute.

A standard approach for solving semi-infinite problems is to replace the infinitely many constraints by a finite number, say M. Finally, after relaxing the interpolatory conditions slightly, we obtain the following linear programming problem:

Find real numbers b_j, $j = 0, 1, \ldots, k$, such that

$$\begin{aligned} \sum_{i=1}^{n} G(t_i;\, b_0, \ldots, b_k) &= \min!, \\ G(t_i;\, b_0, \ldots, b_k) &\geq 0, \quad i = 1, 2, \ldots, n, \\ \sum_{j=0}^{k} b_j \xi_i^j &> 0, \quad \xi_i \in (-1, 1), \quad i = 1, 2, \ldots, M. \end{aligned} \tag{5.11}$$

Here ξ_i, $i = 1, 2, \ldots, M$, are given distinct points in $(-1, 1)$. In our experiments, we always chose the ξ_i's as Chebyshev knots.

It is not hard to check that the feasible set of (5.11) is not empty, and hence (5.11) always has a solution $\rho_k^*(t; M) \equiv \sum_{j=0}^{k} b_j^* t^j$. The second parameter M indicates that ρ_k^* depends on the number M of positivity constraints. Unfortunately, it is possible that $\rho_k^*(t; M)$ has zeros in $(-1, 1)$. Here, one basically has to distinguish between the cases that $\rho_k^*(t; M)$ has one zero μ in the first (respectively, last) interval $(-1, \xi_1)$ (respectively, $(\xi_M, 1)$), or two zeros μ_1, μ_2 in (ξ_i, ξ_{i+1}), $i \in \{0, 1, \ldots, M\}$, where $\xi_0 := -1$ and $\xi_{M+1} := 1$. Such zeros cannot be used as starting values for the nonlinear problem (5.5). Therefore, we replace μ by -1 or 1, and the real zeros μ_1 and μ_2 by the complex zeros $z = (\mu_1 + \mu_2)/2 + i\varepsilon$ and $\overline{z}$, where ε is some small positive number. Notice that

$$(t - \mu_1)(t - \mu_2) - (t - z)(t - \overline{z}) = \frac{1}{4}(\mu_1 - \mu_2)^2 + \varepsilon^2 < |\xi_i - \xi_{i+1}|^2 + \varepsilon^2.$$

For sufficiently large M and sufficiently small ε, this "substitution of zeros" perturbs $\rho_k^*(t; M)$ only slightly.

In our experiments, we always chose $M := 200$. The corresponding linear solution $\rho_k^*(t; M)$ always yielded a good starting guess for the nonlinear solver applied to (5.5). In fact, quite frequently, the solution of (5.11) was such a good approximation that there was no need for applying the nonlinear solver.

6. Examples. In this section, we discuss some practical implementation details of the preconditioned CG method, and we describe some numerical examples.

6.1. Implementation. The basic outline of our preconditioned CG algorithm is as follows.

ALGORITHM 6.1 (Outline of CG with polynomial preconditioning).

1. *Start*: *choose* $x_0 \in \mathbb{C}^N$ *and compute* $r_0 = b - Ax_0$.
2. *Estimate the distribution function*:
   - *Perform n steps of the Lanczos algorithm starting with* $q_0 = r_0/\|r_0\|$.
   - *Compute the Lanczos distribution function* $\tau(t)$ *given by* (2.15).
   - *Set up the interpolation problem* (3.3).
3. *Determine the polynomial preconditioner* ψ_p :
   - *Compute the monotone piecewise cubic interpolant, as described in* §4.
   - *Either solve the associated weighted Chebyshev approximation problem* (3.1), *or compute the Bernstein–Szegö weight function, as described in* §5.
4. *Iterate*: *apply the CG method to the polynomial preconditioned system* $A\psi_p(A)y = b$.

Recall from (2.14) that the distribution function $\sigma(t)$ depends on the eigenvalues of A and the starting guess x_0. Therefore, in step 4 of Algorithm 6.1, we start the CG method with x_0, rather than the iterate x_n that can be constructed from quantities generated by the n Lanczos iterations in step 2. It is worth noticing that the n matrix-vector products with A, which are computed in the course of the n Lanczos steps, can be reused for the first $\lfloor n/p \rfloor$ iterations with the preconditioned CG method in step 4. However, this requires the storage of all $n + 1$ Lanczos vectors $q_0, q_1, \ldots, q_n$ generated in step 2.

In each step of the preconditioned CG algorithm, we must compute a matrix-vector product of the form $A\psi_p(A) \cdot v$. To this end, in our implementation, we represent the polynomial $\lambda\psi_p(\lambda)$ in Newton-form

$$\lambda\psi_p(\lambda) \equiv c_0 + \sum_{j=1}^{p} c_j \prod_{k=0}^{j-1}(\lambda - \xi_k). \tag{6.1}$$

Here, the interpolation points ξ_k are chosen as Chebyshev points shifted to the interval $[\alpha, \beta]$, and they are ordered using van der Corput's sequence. This leads to a stable implementation for computing $A\psi_p(A) \cdot v$, even for polynomials ψ_p of high degree; see, for example, [7]. Finally, the evaluation of $A\psi_p(A) \cdot v$ is performed by a Horner scheme applied to (6.1).

6.2. Performance. In all our examples, we use coefficient matrices A that result from a finite-difference discretization of the diffusion equation

$$-\frac{\partial}{\partial x}\left(\mu\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\nu\frac{\partial u}{\partial y}\right) = g \tag{6.2}$$

on the unit square in $\mathbb{R}^2$ with Dirichlet–Neumann boundary conditions. Equations of this type arise, for example, in connection with groundwater flow problems; see [17].

In order to ensure continuity of fluid flux even for highly discontinuous coefficient functions μ and ν, one uses the following five-point stencil:

$$-\eta_{i-1,j}u_{i-1,j} - \eta_{ij}u_{i+1,j} - \kappa_{i,j-1}u_{i,j-1} - \kappa_{ij}u_{i,j+1} + \rho_{ij}u_{ij} = h^2 g_{ij},$$

where h is the distance between adjacent nodes. Moreover, we set

$$\eta_{ij} = \delta(\mu_{ij};\, \mu_{i+1,j}),$$

$$\kappa_{ij} = \delta(\nu_{ij};\, \nu_{i,j+1}),$$

where

$$\delta(c;\, d) = \begin{cases} 2cd/(c+d) & \text{if } c + d \neq 0\,, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\rho_{ij} = \begin{cases} \eta_{i-1,j} + \eta_{ij} + \kappa_{i,j-1} + \kappa_{ij} & \text{if this sum} \neq 0, \\ 1 & \text{otherwise.} \end{cases}$$

Neumann boundary conditions are enforced by setting appropriate η and κ to zero.

We have tested the preconditioned CG method extensively for different cases. Here, we present three examples that show the typical behavior of the various polynomial preconditioners.

In addition to the Chebyshev polynomial preconditioner and the weighted Chebyshev preconditioner, we also implemented a preconditioner based on least-squares polynomials. For the latter, the Chebyshev norm in problem (3.1) is substituted by a weighted L_2-norm

$$\min_{\varphi \in \mathcal{P}_p:\, \varphi(\zeta)=1} \int_{-1}^{1} \varphi(t)^2 w(t)\, dt. \tag{6.3}$$

We have used the Legendre weight $w(1) \equiv 1$. For this special case the solution of (6.3) can be found in [18] (see also [2]).

All three examples were run for two different step sizes h: first with $h = \frac{1}{64}$ and then with half the step size $h = \frac{1}{128}$. In the following, numbers without brackets always correspond to $h = \frac{1}{64}$, while numbers for $h = \frac{1}{128}$ are given in brackets. For instance, the matrix in Example 6.1 is of order $N = 3844$ [15876] if $h = \frac{1}{64}$ $\left[\frac{1}{128}\right]$.

In all examples, the initial vector $x_0 = 0$ is chosen, and the right-hand side b is a random vector with entries $\in [-1, 1]$. The algorithm is stopped as soon as the kth residual satisfies $\|r_k\|/\|r_0\| \leq 10^{-10}$ $[10^{-6}]$. All computations were carried out in double-precision arithmetic, i.e., with approximately 16 significant decimal digits.

For the definition of μ and ν, we decompose the unit square into four parts, $R0$, $R1$, $R2$, and $R3$, as indicated in Fig. 6.1.

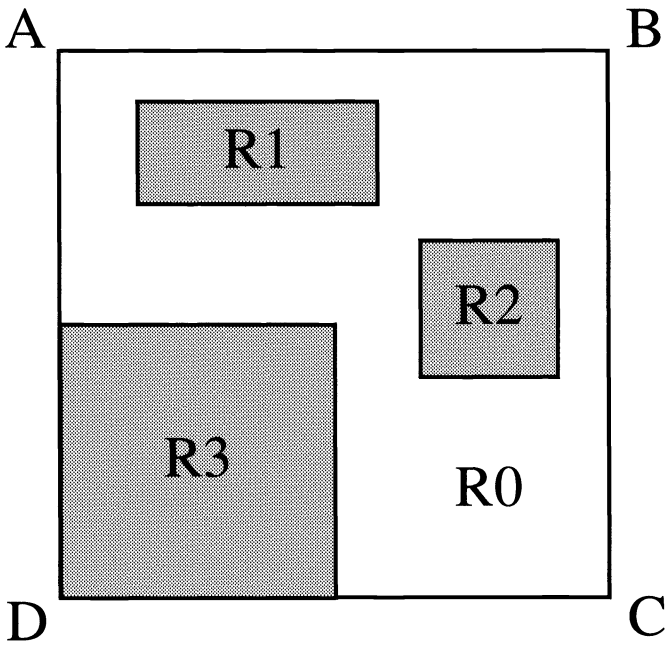


FIG. 6.1. *Decomposition of the unit square.*

In step 2 of Algorithm 6.1, we always performed $n = 20$ Lanczos steps. Note that, in this case, the Bernstein–Szegö preconditioner is then known explicitly for $p \geq 9$, where p is the degree of $t\psi_p(t)$. The weighted Chebyshev preconditioner (referred to as Remez or Bernstein, respectively) and the least-squares polynomial preconditioner (referred to as Legendre) are computed for the interval $[0, \theta_n]$, where θ_n is the largest Ritz value. The Chebyshev polynomial preconditioner (referred to as Chebyshev) is based on the "true" spectrum $[\lambda_1, \lambda_L]$.

Example 6.1. Here we consider (6.2) with $\mu \equiv \nu \equiv 1$ on $R0 \cup R1 \cup R2 \cup R3$ and Dirichlet boundary condition $u \equiv$ const on $\overline{AB}$, $\overline{BC}$, $\overline{CD}$, and $\overline{DA}$. Note that, in this case, (6.2) just reduces to Poisson's equation. The order of the resulting system is $N = 3844$ [15876], and the Euclidean condition number of A is about 1600 [6560]. The CG method, without any preconditioning, converged after 229 [286] steps. The iteration counts for the various polynomial preconditioners are listed in Table 6.1.

For the case $p = 5$ and $h = \frac{1}{64}$, the Chebyshev polynomial $T_5(t)/T_5(\zeta)$ (dashed line), the solution $\varphi_5(t)$ of (3.1) (solid line), and the functions $\pm d_5/w(t)$ (dotted lines) are plotted

TABLE 6.1
Polynomial preconditioning for Example 6.1.

p	5		10		15	
Bernstein	–	[–]	31	[33]	23	[27]
Remez	62	[74]	32	[41]	25	[30]
Chebyshev	95	[116]	48	[59]	33	[40]
Legendre	121	[149]	86	[104]	70	[85]

in Fig. 6.2. The number d_5 denotes the minimal deviation of problem (3.1) with corresponding Bernstein–Szegö weight function. The Chebyshev polynomial is based on [0.005, 7.995] and $\varphi_5(t)$ is based on [0, 7.995]. The underlying distribution function $\sigma(t)$ is shown in Fig. 3.1.

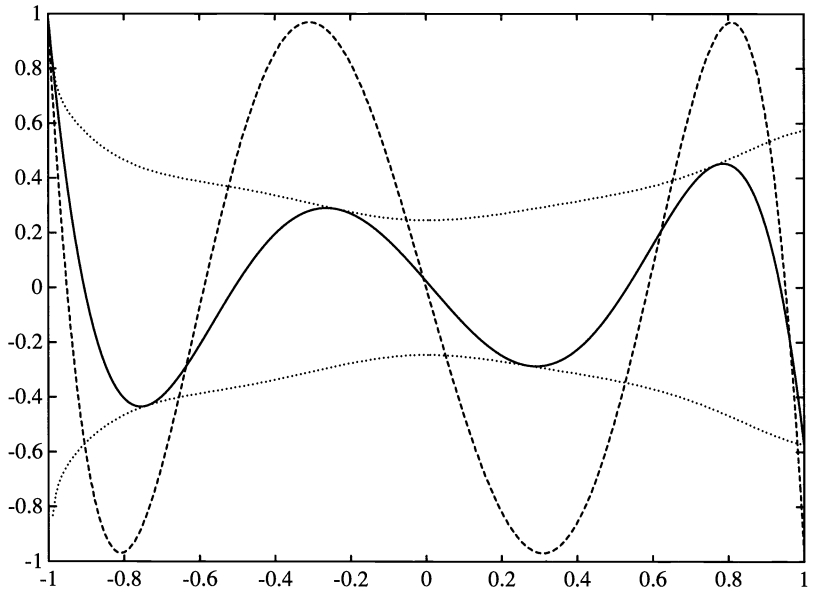


FIG. 6.2. *Chebyshev polynomial, weighted Chebyshev polynomial, and Bernstein–Szegö weight function for Example* 6.1 *and* $p = 5$.

Example 6.2. In this example, we choose continuous nonconstant coefficient functions in (6.2). More precisely, we set $\mu(x, y) \equiv \nu(x, y) \equiv x + y$ on $R0 \cup R1 \cup R2 \cup R3$. Furthermore, we have Dirichlet boundary conditions on $\overline{DA}$ and no-flow Neumann boundary conditions on $\overline{AB}$, $\overline{BC}$, and $\overline{CD}$. The matrix A has been symmetrically scaled to have unit main diagonal. The order of A is $N = 4032$ [16256], and the condition number of A is about 21000 [56180]. The CG method, without any preconditioning, converged after 370 [569] steps. The iteration counts for the various polynomial preconditioners are listed in Table 6.2.

Example 6.3. In this example, the coefficient functions in (6.2) are highly discontinuous. They have different values on all four components of the unit square:

TABLE 6.2
Polynomial preconditioning for Example 6.2.

p	5		10		15	
Bernstein	–	[–]	54	[65]	35	[49]
Remez	100	[148]	55	[80]	38	[58]
Chebyshev	334	[367]	168	[184]	113	[124]
Legendre	198	[300]	141	[211]	115	[172]

$$\mu(x, y) \equiv \nu(x, y) \equiv \begin{cases} 1 & \text{for } (x, y) \in R0, \\ 1000 & \text{for } (x, y) \in R1, \\ 0 & \text{for } (x, y) \in R2, \\ 100 & \text{for } (x, y) \in R3. \end{cases}$$

Again, we choose Dirichlet boundary conditions on $\overline{DA}$ and no-flow Neumann boundary conditions on $\overline{AB}$, $\overline{BC}$, and $\overline{CD}$. The matrix A has been symmetrically scaled to have unit main diagonal. The order of A is $N = 4032$ [16256], and the condition number of A is about 450000 [1905837]. The unpreconditioned CG method converged after 395 [678] steps. The iteration counts for the various polynomial preconditioners are listed in Table 6.3.

TABLE 6.3
Polynomial preconditioning for Example 6.3.

p	5		10		15	
Bernstein	–	[–]	57	[92]	40	[65]
Remez	104	[176]	55	[96]	42	[70]
Chebyshev	>1000	[>1000]	769	[958]	526	[648]
Legendre	210	[357]	148	[252]	121	[205]

The results demonstrate the effectiveness of the proposed weighted "Bernstein–Szegö" and "Remez" polynomial preconditioners. Moreover, they show that these preconditioners are usually superior to the other considered preconditioners. In particular, both preconditioners perform very well for ill-conditioned problems. Furthermore, we note that, for the case $h = \frac{1}{128}$, the "Bernstein–Szegö" preconditioner is consistently better than the "Remez" preconditioner.

7. Concluding remarks. On modern architectures, it is attractive to combine the CG algorithm with polynomial preconditioning. In this paper, we have presented an approach for adapting polynomial preconditioners to the actual eigenvalue distribution of the coefficient matrix of the linear system. Our technique is based on the observation that good estimates for the eigenvalue distribution can be derived after only a few steps of the Lanczos process. We then use this information to construct a weight function for a suitable Chebyshev approximation problem. The solution of this problem yields the polynomial preconditioner. We have explored the use of Bernstein–Szegö weights.

We have presented some numerical examples that demonstrate the effectiveness of the proposed polynomial preconditioner. Our results suggest that the adaptive weighted polynomial preconditioner is superior to other polynomial preconditioners.

Acknowledgment. We would like to thank Youcef Saad for bringing reference [21] to our attention.

REFERENCES

[1] R. L. ADLER AND T. J. RIVLIN, *Ergodic and mixing properties of Chebyshev polynomials*, Proc. Amer. Math. Soc., 15 (1964), pp. 794–796.

[2] S. F. ASHBY, T. A. MANTEUFFEL, AND J. S. OTTO, *A comparison of adaptive Chebyshev and least squares polynomial preconditioning for Hermitian positive definite linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1–29.

[3] S. BERNSTEIN, *Sur une classe de polynomes d'écart minimum*, C. R. Acad. Sci. Paris, 190 (1930), pp. 237–240.

[4] K. W. BRODLIE, *A review of methods for curve and function drawing*, in Mathematical Methods in Computer Graphics and Design, K. W. Brodlie, ed., Academic Press, London, 1980, pp. 1–37.

[5] P. CONCUS, G. H. GOLUB, AND D. P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–332.

[6] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[7] B. FISCHER AND L. REICHEL, *Newton interpolation in Fejér and Chebyshev points*, Math. Comp., 53 (1989), pp. 265–279.

[8] R. W. FREUND, *On some approximation problems for complex polynomials*, Constr. Approx., 4 (1988), pp. 111–121.

[9] ———, *Polynomial preconditioners for Hermitian and certain non-Hermitian matrices*, Minisymposium Presentation, 1989 SIAM Annual Meeting, San Diego, July 1989.

[10] F. N. FRITSCH AND J. BUTLAND, *A method for constructing local monotone piecewise cubic interpolation*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 300–304.

[11] F. N. FRITSCH AND R. E. CARLSON, *Monotone piecewise cubic interpolation*, SIAM J. Numer. Anal., 17 (1980), pp. 238–246.

[12] G. H. GOLUB AND L. B. SMITH, *Chebyshev approximation of continuous functions by a Chebyshev system of functions*, Collect. Algorithms CACM, 414 (1971), pp. P1–P10.

[13] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[14] G. H. GOLUB AND R. S. VARGA, *Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods*, Numer. Math., 3 (1961), pp. 147–168.

[15] A. GREENBAUM, *Comparison of splittings used with the conjugate gradient algorithm*, Numer. Math., 33 (1979), pp. 181–194.

[16] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–436.

[17] B. HOLTER ANDA G. VANDENBERGHE, *A comparison of vectorized methods for solving the two-dimensional diffusion equation: multigrid versus polynomial preconditioned conjugate gradient*, Appl. Math. Comput., 40 (1990), pp. 77–103.

[18] O. G. JOHNSON, C. A. MICCHELLI, AND G. PAUL, *Polynomial preconditioners for conjugate gradient calculations*, SIAM J. Numer. Anal., 20 (1983), pp. 362–376.

[19] S. KARLIN AND L. S. SHAPLEY, *Geometry of Moment Spaces*, Memoirs of the American Mathematical Society, 12, Amer. Math. Soc., Providence, RI, 1953.

[20] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 255–282.

[21] ———, *Chebyshev polynomials in the solution of large-scale linear systems*, in Proceedings of the Association for Computing Machinery, Toronto, Sauls Lithograph Co., Washington, DC, 1953, pp. 124–133.

[22] G. MEINARDUS, *Approximation of Functions: Theory and Numerical Methods*, Springer-Verlag, New York, 1967.

[23] D. P. O'LEARY, *Yet another polynomial preconditioner for the conjugate gradient algorithm*, Linear Algebra Appl., 154–156 (1991), pp. 377–388.

[24] W. P. REINHARDT, *L^2 discretization of atomic and molecular electronic continua: moment, quadrature and J-matrix techniques*, Comput. Phys. Comm., 17 (1979), pp. 1–21.

[25] H. RUTISHAUSER, *Theory of gradient methods*, in Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, M. Engeli, Th. Ginsburg, H. Rutishauser, and E. Stiefel, eds., Birkhäuser, Basel, 1959, pp. 24–49.

[26] Y. SAAD, *Practical use of polynomial preconditionings for the conjugate gradient method*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 865–881.

[27] ———, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1200–1232.

[28] E. L. STIEFEL, *Kernel polynomials in linear algebra and their numerical applications*, U.S. National Bureau of Standards, Applied Mathematics Series, 49 (1958), pp. 1–22.

[29] G. SZEGÖ, *Orthogonal Polynomials*, 4th ed., Amer. Math. Soc., Providence, RI, 1975.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 427–439, March 1994

012

A ROBUST GMRES-BASED ADAPTIVE POLYNOMIAL PRECONDITIONING ALGORITHM FOR NONSYMMETRIC LINEAR SYSTEMS*

WAYNE JOUBERT†

Abstract. In this study a hybrid generalized minimal residual (GMRES) / polynomial preconditioning algorithm for solving nonsymmetric systems of linear equations is defined. The algorithm uses the results from cycles of restarted GMRES to form an effective polynomial preconditioner, typically resulting in decreased work requirements. The algorithm has the advantage over other hybrid algorithms in that its convergence behavior is well understood: the new algorithm converges for all starting vectors if and only if restarted GMRES converges. The results of numerical experiments with the algorithm are presented.

Key words. iterative methods, nonsymmetric linear systems, GMRES, polynomial preconditioning

AMS subject classification. 65F10

1. Introduction.

A desirable method for solving the linear system

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

where $A \in \mathbb{C}^{N\times N}$ is nonsingular, is the minimal residual method

$$u^{(n)} = u^{(0)} + q_{n-1}(A)r^{(0)}, \quad \deg q_{n-1} \le n-1, \quad q_{n-1} \text{ such that } ||r^{(n)}|| \text{ minimized}, \tag{2a}$$

or, alternatively,

$$u^{(n)} \in u^{(0)} + \mathbf{K}_n(r^{(0)}, A), \qquad Ar^{(n)} \perp \mathbf{K}_n(r^{(0)}, A), \tag{2b}$$

where $\{u^{(i)}\}_{i\ge 0}$ denote the iterates, $r^{(i)} = b - Au^{(i)}$ denotes the associated residuals, $\mathbf{K}_n(v, A) = \text{span}\{A^i v\}_{i=0}^{n-1}$ is the associated Krylov space, and $||\cdot||$ is the standard 2-norm of vectors. We also let $P_n(z) = 1 - zq_{n-1}(z)$, so that $r^{(n)} = P_n(A)r^{(0)}$, which condition identifies (2a), (2b) as a polynomial method.

When A is Hermitian positive definite (HPD) or possesses a similar property, the polynomial q_{n-1} may be determined by only n matrix-vector products $A \cdot v$ along with $2n$ inner products. This fact leads to effective iterative methods such as the conjugate residual method. On the other hand, to form q_{n-1} in the general case requires at least $n^2/2$ inner products, a requirement that appears insurmountable and that renders standard iterative methods such as full GMRES for performing (2) prohibitively costly due to long recurrence relations [2], [3], [12], and [9, §2.5].

A typical remedy to the order-n^2 work problem is to restart (2) periodically every s iterations. This is the basis of the GMRES(s) algorithm, for example, [18]; see [11] for a survey of restarted methods. Choosing s large improves the convergence of the method; however, an increased average work per iteration results (see [10]).

A further remedy to the long recurrence problem comes in the form of hybrid methods. These methods typically apply GMRES until sufficient information is extracted from A, typically in the form of eigenvalue information, so that an effective polynomial method such as polynomial preconditioning or Chebyshev acceleration may be applied based on this information (see, e.g., [15], [1], [16]). This approach has the advantage of using short recurrences

*Received by the editors May 18, 1992; accepted for publication (in revised form) April 12, 1993. This work was supported in part by National Science Foundation grant DCR-8518722, and Department of Energy grant DE-FG05-87ER25048, with the University of Texas at Austin. This work was also supported in part by Department of Energy grant W-7405-ENG-36 with Los Alamos National Laboratory.

†Los Alamos National Laboratory, Los Alamos, New Mexico 87545 (wdj@lanl.gov).

for the sections of the algorithm not using GMRES. On the other hand, the composite nature of these algorithms often makes it difficult to prove rigorous convergence results for the algorithms or to predict their behavior compared to other iterative methods.

In this study, an alternative hybrid algorithm is proposed. For this algorithm, the basic framework of restarted GMRES is maintained, but for each restart cycle the $(s-1)$ degree polynomial that would normally be calculated by (2) is optionally replaced by a polynomial preconditioner of the same degree that is based on information from previous GMRES cycles. The new algorithm has the advantage of converging for exactly the same class of matrices as restarted GMRES, and often at significantly less cost. The savings are even greater on parallel machines and other architectures for which inner product computations are particularly expensive.

The remainder of this paper is organized as follows. In §2 we consider the theoretical feasibility of replacing GMRES cycles with polynomial preconditioning. Then in §3 the new algorithm is defined. In §4 the results of numerical experiments are presented.

2. The feasibility of polynomial preconditioning. In this section we consider the relative convergence behavior of (2) compared to the best fixed polynomial preconditioning that can be applied to solve (1). We assume throughout that A is nonsingular.

We begin with definitions of basic convergence bounds for polynomial methods. Following [10], for $\mathbb{K}$ denoting the reals $\mathbb{R}$ or the complex numbers $\mathbb{C}$, and $A \in \mathbb{K}^{N\times N}$ and $n \geq 0$, let

$$\psi_{n,\mathbb{K}}(A) = \sup_{\substack{r\in\mathbb{K}^N \\ r\neq 0}} \inf_{\substack{\deg P_n \leq n \\ P_n(0)=1}} \frac{||P_n(A)r||}{||r||} = \sup_{\substack{r^{(0)}\in\mathbb{K}^N \\ r^{(0)}\neq 0}} \frac{||r^{(n)}||}{||r^{(0)}||},$$

$$\varphi_{n,\mathbb{K}}(A) = \inf_{\substack{\deg P_n \leq n \\ P_n(0)=1}} \sup_{\substack{r\in\mathbb{K}^N \\ r\neq 0}} \frac{||P_n(A)r||}{||r||},$$

where $r^{(n)}$ refers to the nth residual generated by the minimal residual method (2), and $P_n(z) = 1 - zq_{n-1}(z)$ is a polynomial over $\mathbb{C}$. When the field $\mathbb{K}$ is omitted in the notation, it is assumed to be $\mathbb{C}$. Here and throughout, $||v|| = \sqrt{v^*v}$, where v^* denotes the conjugate transpose of v, and $||M||$ denotes the induced matrix norm.

The quantities $\psi_{n,\mathbb{K}}(A)$ and $\varphi_{n,\mathbb{K}}(A)$ are bounds that denote the possible residual norm decrease for two particular polynomial methods. They differ in the following important respect. The first quantity gives a sharp bound for the convergence of GMRES or any other algorithm implementing (2). In this case the polynomial P_n is chosen as a function of $r^{(0)}$. On the other hand, the second quantity denotes the performance of the best polynomial preconditioner (defined here as q_{n-1}) that is chosen independent of $r^{(0)}$ and that must be effective for all such vectors in $\mathbb{K}^N$. The main concern of this section is the relationship between these two quantities; in particular, it is of interest to determine when they are equal.

Basic results on these quantities are set forth in [10]. Of interest to us here are the following properties for $A \in \mathbb{K}^{N\times N}$:

(i) $0 \leq \psi_{n,\mathbb{K}}(A) \leq \varphi_{n,\mathbb{K}}(A) \leq 1$ for all n.

(ii) $\psi_{0,\mathbb{K}}(A) = \varphi_{0,\mathbb{K}}(A) = 1$, while $\psi_{n,\mathbb{K}}(A) = \varphi_{n,\mathbb{K}}(A) = 0$ for any n satisfying $n \geq d(A) = \min\{\deg P : P \text{ monic}, P(A) = 0\}$.

(iii) When $A \in \mathbb{R}^{N\times N}$, $\psi_{n,\mathbb{R}}(A) \leq \psi_{n,\mathbb{C}}(A)$, while $\varphi_{n,\mathbb{R}}(A) = \varphi_{n,\mathbb{C}}(A)$.

(iv) For fixed n, $\psi_{n,\mathbb{K}}(A)$ and $\varphi_{n,\mathbb{K}}(A)$ are each continuous in A.

(v) $\psi_{n,\mathbb{K}}(A)^2 = \sup_{r\in\mathbb{K}^N} F_n(r)$ holds for $A \in \mathbb{K}^{N\times N}$, where $F_n(r)$ is defined to be zero for r satisfying $d(r, A) < n$ and elsewhere $F_n(r)$ is defined to be

$$1 - r^* A K_n(r, A)[K_n(r, A)^* A^* A K_n(r, A)]^{-1} K_n(r, A)^* A^* r / r^* r. \tag{3}$$

Here, $d(v, A) = \min\{\deg P : P \text{ monic}, P(A)v = 0\}$ denotes the degree of a vector, and $K_n(v, A) = \begin{bmatrix} v & Av & \dots & A^{n-1}v \end{bmatrix}$ is the standard Krylov basis matrix. Note that $F_n(r^{(0)}) = ||r^{(n)}||^2/||r^{(0)}||^2$, where $r^{(n)}$ is defined by (2), which gives the square of the residual decrease from n steps of GMRES applied to a vector.

(vi) The map F_n of (3) is the zero map when $d(A) \le n$. Otherwise it is continuous, and in fact it is C^∞ on the complement of the closed measure-zero set $\{r : d(r, A) < n\}$, where F_n is considered as a map on $\mathbb{R}^{2N}$. The latter fact follows directly from the fact that $F_n(r)$ is a rational function of the entries of $\operatorname{Re} r$ and $\operatorname{Im} r$.

From the standpoint of iterative methods, it is desirable that one or both of the following conditions be satisfied.

Condition 1. For given $A \in \mathbb{K}^{N\times N}$, $\psi_{s,\mathbb{K}}(A) < 1 \Leftrightarrow \varphi_{s,\mathbb{K}}(A) < 1$. If this property holds, then a convergent polynomial preconditioner of degree $s - 1$ necessarily exists (in the sense that $\varphi_{s,\mathbb{K}}(A) < 1$) if and only if GMRES($s$) is guaranteed to converge for all $r^{(0)} \in \mathbb{K}^N$.

Condition 2. For given $A \in \mathbb{K}^{N\times N}$, $\psi_{s,\mathbb{K}}(A) = \varphi_{s,\mathbb{K}}(A)$. If this stronger condition holds, then not only does a convergent polynomial preconditioner exist if and only if GMRES(s) must converge, but one exists that has the same convergence bound $||r^{(ms)}||/||r^{(0)}|| \le \psi_s(A)^m$ as GMRES(s).

It is not clear whether generally $\psi_{n,\mathbb{K}}(A) = \varphi_{n,\mathbb{K}}(A)$. However, the following results do hold.

THEOREM 1. *For* $\mathbb{K} = \mathbb{R}$ *or* $\mathbb{C}$, *and* $A \in \mathbb{K}^{N\times N}$, $\psi_{1,\mathbb{K}}(A) = \varphi_{1,\mathbb{K}}(A)$.

Proof. Let P be a minimizing polynomial for $\varphi_{1,\mathbb{K}}(A)$. As shown in [10], it is possible to choose P with coefficients in $\mathbb{K}$.

We first show that for any nonzero $\alpha \in \mathbb{K}$, there exists nonzero v within V, V here defined to be the eigenspace (over $\mathbb{K}$) associated with the maximal eigenvalue of $P(A)^*P(A)$, such that $\operatorname{Re}\alpha^* v^* A^* P(A)v \ge 0$. Note that for any v and α, and any $\epsilon > 0$, $||[P(A) + \epsilon\alpha A]v||^2 = ||P(A)v||^2 + 2\epsilon \operatorname{Re}[\alpha^* v^* A^* P(A)v] + \epsilon^2|\alpha|^2||Av||^2$. If the result is false, then for some α, $\operatorname{Re}\alpha^* v^* A^* P(A)v < 0$ for all nonzero $v \in V$, in which case there exists $\tilde{c}_1 > 0$ with $2\operatorname{Re}\alpha^* v^* A^* P(A)v \le -\tilde{c}_1||P(A)v||^2$ for all $v \in V$.

Let $\tilde{P}(z) = P(z) + \epsilon\alpha z$ and $w = v\cos\theta + v^\perp \sin\theta$, $v \in V$, $v^\perp \in V^\perp$, $||v|| = ||v^\perp|| = 1$. We seek to show for some ϵ, $||\tilde{P}(A)w|| < ||P(A)||$ for all w, a contradiction to P being a minimizer. Note that

$$\begin{aligned}
||\tilde{P}(A)w||^2 &= \cos^2\theta||\tilde{P}(A)v||^2 + 2\cos\theta\sin\theta \operatorname{Re} v^*\tilde{P}(A)^*\tilde{P}(A)v^\perp + \sin^2\theta||\tilde{P}(A)v^\perp||^2 \\
&= \cos^2\theta||P(A)v||^2 + 2\epsilon\cos^2\theta \operatorname{Re}\alpha^* v^* A^* P(A)v \\
&\quad + 2\epsilon\cos\theta\sin\theta \operatorname{Re}[\alpha v^* P(A)^* A v^\perp + \alpha^* v^* A^* P(A)v^\perp] \\
&\quad + \sin^2\theta||P(A)v^\perp||^2 + 2\epsilon\sin^2\theta \operatorname{Re}\alpha^* v^{\perp *} A^* P(A)v^\perp + \mathcal{O}(\epsilon^2).
\end{aligned}$$

Thus there exist constants c_2, c_3, c_4, and c_5 (with $c_5 > 0$, such that the condition $||P(A)v^\perp||^2 \le ||P(A)v||^2 - c_5$ is satisfied, by definition of V), such that for any v, $v^\perp$ as defined above, $||\tilde{P}(A)w||^2$ is bounded by

$$\begin{aligned}
&\cos^2\theta(1 - \epsilon\tilde{c}_1)||P(A)v||^2 + \epsilon c_2\cos\theta\sin\theta + \epsilon c_3\sin^2\theta + \sin^2\theta||P(A)v^\perp||^2 + \epsilon^2 c_4 \\
&\quad \le ||P(A)v||^2 - \epsilon c_1\cos^2\theta + \epsilon c_2\cos\theta\sin\theta + \epsilon c_3\sin^2\theta - c_5\sin^2\theta + \epsilon^2 c_4,
\end{aligned}$$

where $c_1 = \tilde{c}_1||P(A)||^2$. It is sufficient to find $\epsilon > 0$ such that for all θ,

$$\epsilon(-c_1\cos^2\theta + c_2\cos\theta\sin\theta + c_3\sin^2\theta) - c_5\sin^2\theta + \epsilon^2 c_4 < 0.$$

But the left-hand side of this inequality equals

$$-c_5 \sin^2\theta + \epsilon^2 c_4 - \epsilon c_1 + \epsilon \sin\theta(c_1 \sin\theta + c_2 \cos\theta + c_3 \sin\theta)$$
$$\leq -c_5 \sin^2\theta + \epsilon^2 c_4 - \epsilon c_1 + \epsilon c_6 \sin\theta$$

for some c_6, for all θ. Letting $z = \sin\theta$ yields the quadratic $-c_5 z^2 + \epsilon c_6 z + (\epsilon^2 c_4 - \epsilon c_1)$, which can be made negative for all real z by choosing ϵ sufficiently small.

This establishes that for any $\alpha \in \mathbb{K}$ there exists a nonzero $v \in V$ such that $\mathrm{Re}\,\alpha^* v^* A^* \cdot P(A)v \geq 0$. We now show that for any $c \in \mathbb{K}$ there is nonzero $v \in V$ such that $\mathrm{Re}[c^* v^* A^* P(A)v] = 0$. Otherwise, letting $\alpha = \pm c$ yields v_1, $v_2 \in V$ such that $\mathrm{Re}[c^* v_1^* A^* P(A)v_1] > 0$ and $\mathrm{Re}[c^* v_2^* A^* P(A)v_2] < 0$. Note v_1 and v_2 must be linearly independent; otherwise, $v_1 = \xi v_2$, $\xi \in \mathbb{K}$, yields $v_1^* A^* P(A)v_1 = |\xi|^2 v_2^* A^* P(A)v_2$, a contradiction. Setting $v_t = t v_1 + (1-t)v_2$, $0 \leq t \leq 1$, yields by a continuity argument some t such that $\mathrm{Re}[c^* v_t^* A^* P(A)v_t] = 0$, giving the result.

Let $\underline{V}$ be a matrix whose columns form a basis over $\mathbb{K}$ of V. We have shown for any $c \in \mathbb{K}$ there exists nonzero $u \in \mathbb{K}^{\mathrm{DimV}}$ such that $\mathrm{Re}[c^* u^* \underline{V}^* A^* P(A)\underline{V}u] = 0$. When $\mathbb{K} = \mathbb{R}$, take $c = 1$ to give $r = \underline{V}u \neq 0$ with $r^* A^* P(A)r = 0$. When $\mathbb{K} = \mathbb{C}$, we have shown $0 \in F(H(c^* \underline{V} A^* P(A)\underline{V}))$ for any $c \in \mathbb{C}$, where $H(M) = (M + M^*)/2$ denotes the Hermitian part of a matrix M and $F(M) = \{x^* M x : x \in \mathbb{C}^N, ||x|| = 1\}$ the field of values [8]. This implies that $0 \in F(\underline{V} A^* P(A)\underline{V})$: otherwise, by the convexity of the field of values, there exists $c = e^{i\theta}$ such that $H(c^* \underline{V} A^* P(A)\underline{V})$ is positive definite, i.e., $0 \notin F(H(c^* \underline{V} A^* P(A)\underline{V}))$. Thus there exists nonzero $r = \underline{V}u \in V$ such that $r^* A^* P(A)r = 0$.

Thus, for either the real or complex case, $Ar \perp P(A)r$ for nonzero $r \in V$. It is easily seen that this $P(z) = 1 - z q_0(z)$ satisfies (2b) for this r, and in fact $\psi_{1,\mathbb{K}}(A)^2 \geq ||P(A)r||^2/||r||^2 = ||P(A)||^2 \geq \varphi_{1,\mathbb{K}}(A)^2$. □

THEOREM 2. *For $A \in \mathbb{R}^{N\times N}$, $\psi_{2,\mathbb{R}}(A) < 1 \Leftrightarrow \varphi_{2,\mathbb{R}}(A) < 1$.*

Proof. Let $n = 2$. It is clear from (3) that $\psi_{n,\mathbb{R}}(A) < 1$ implies that for all real v satisfying $d(v, A) \geq n$ and $||v|| = 1$, $v^* A K_n(v, A) \neq 0$. The same is true when $d = d(v, A) < n$ and $||v|| = 1$: in this case, we have

$$0 = v^* v - v^* A K_d(v, A)[K_d(v, A)^* A^* A K_d(v, A)]^{-1} K_d(v, A)^* A^* v,$$

so that $0 \neq v^* A K_d(v, A)$, a subvector of $v^* A K_n(v, A)$. Defining

$$\mathcal{F}_{n,\mathbb{K}}(A) = \{v^* A K_n(v, A) : v \in \mathbb{K}^N, ||v|| = 1\} \subseteq \mathbb{K}^n$$

for $A \in \mathbb{K}^{N\times N}$, we thus have that $0 \notin \mathcal{F}_{n,\mathbb{R}}(A)$. According to [7, p. 86], $\mathcal{F}_{2,\mathbb{R}}(A)$ is convex in $\mathbb{R}^2$ when $N \geq 3$.

When $N \leq 2$, $\psi_{2,\mathbb{R}}(A) = \varphi_{2,\mathbb{R}}(A) = 0$. When $N \geq 3$, since $0 \notin \mathcal{F}_{n,\mathbb{R}}(A)$, by the Hahn–Banach theorem there is a hyperplane through zero that does not intersect $\mathcal{F}_{2,\mathbb{R}}(A)$. Letting $c = [c_1\ c_2]^T$ denote the appropriate normal vector to this hyperplane, we have that $c^* \mathcal{F}_{2,\mathbb{R}}(A)$ consists only of positive numbers. Thus $c_1 v^* A v + c_2 v^* A^2 v$ is bounded above zero for all real v satisfying $||v|| = 1$. Now let $P(z) = 1 - \alpha(c_1 z + c_2 z^2)$. Then $||P(A)v||^2/||v||^2 = 1 - 2\alpha v^*(c_1 A v + c_2 A^2)v/v^* v + \alpha^2 ||(c_1 A v + c_2 A^2)v||^2/v^* v$ which, for α sufficiently small, is bounded beneath one for all nonzero v. □

THEOREM 3. *For $A \in \mathbb{R}^{N\times N}$ HPD, $\psi_{n,\mathbb{R}}(A) = \varphi_{n,\mathbb{R}}(A)$.*

Proof. See [4]. □

THEOREM 4. *For $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$ and $A \in \mathbb{K}^{N\times N}$ normal, $\psi_{n,\mathbb{K}}(A) = \varphi_{n,\mathbb{K}}(A)$.*

Proof. See [5] for an alternate proof for the $\mathbb{K} = \mathbb{R}$ case. If $n \geq d(A)$ then we are done. Otherwise for $r, s \in \mathbb{K}^N$ and $\epsilon > 0$ small, let

$$r' = (1 - \epsilon^2)^{1/2} r + \epsilon s = (1 - \epsilon^2/2)r + \epsilon s + \mathcal{O}(\epsilon^4)$$

with $d(r, A) \geq n$. Also let $||r|| = ||s|| = 1$ and $r \perp s$, so $||r'|| = 1$. Let P_r be the least squares polynomial (2a), (2b) of degree not greater than n associated with r, and let $K_r = K_n(r, A)$, $K_s = K_n(s, A)$. Note that $P_r(A)r = r - AK_r[K_r^*A^*AK_r]^{-1}K_r^*A^*r$, and P_r is a polynomial over $\mathbb{K}$. Note also that:

(i) $K_{r'} \equiv K_n(r', A) = (1 - \epsilon^2/2)K_r + \epsilon K_s + \mathcal{O}(\epsilon^3)$;

(ii) $K_{r'}^*A^*AK_{r'} = (1-\epsilon^2)K_r^*A^*AK_r + \epsilon K_r^*A^*AK_s + \epsilon K_s^*A^*AK_r + \epsilon^2 K_s^*A^*AK_s + \mathcal{O}(\epsilon^3)$;

(iii) $[K_{r'}^*A^*AK_{r'}]^{-1} = (1 + \epsilon^2)K_r^*A^*AK_r^{-1}$
$-\epsilon K_r^*A^*AK_r^{-1}[K_r^*A^*AK_s + K_s^*A^*AK_r]K_r^*A^*AK_r^{-1}$
$+\epsilon^2 K_r^*A^*AK_r^{-1}[K_r^*A^*AK_s + K_s^*A^*AK_r]$
$\cdot K_r^*A^*AK_r^{-1}[K_r^*A^*AK_s + K_s^*A^*AK_r]K_r^*A^*AK_r^{-1}$
$-\epsilon^2 K_r^*A^*AK_r^{-1}[K_s^*A^*AK_s]K_r^*A^*AK_r^{-1} + \mathcal{O}(\epsilon^3)$.

Then, since $s - AK_s[K_r^*A^*AK_r]^{-1}K_r^*A^*r = P_r(A)s$, we have after some manipulation

$$F_n(r') = [r^*P_r(A)^*P_r(A)r] + 2\epsilon \operatorname{Re}[s^*P_r(A)^*P_r(A)r]$$
$$+\epsilon^2\Big[s^*P_r(A)^*P_r(A)s - r^*P_r(A)^*P_r(A)r - [K_r^*A^*P_r(A)s + K_s^*A^*P_r(A)r]^*$$
$$\cdot[K_r^*A^*AK_r]^{-1}[K_r^*A^*P_r(A)s + K_s^*A^*P_r(A)r]\Big] + \mathcal{O}(\epsilon^3).$$

If r is a (global) maximizer for F_n (and thus a local maximizer), then F_n is smooth near r, so we must have

$$\text{(4a)} \qquad \operatorname{Re} s^*P_r(A)^*P_r(A)r = 0,$$

$$\text{(4b)} \qquad \begin{aligned} &s^*P_r(A)^*P_r(A)s - r^*P_r(A)^*P_r(A)r \\ &\quad - [K_r^*A^*P_r(A)s + K_s^*A^*P_r(A)r]^*[K_r^*A^*AK_r]^{-1} \\ &\quad \cdot [K_r^*A^*P_r(A)s + K_s^*A^*P_r(A)r] \leq 0 \end{aligned}$$

for any $s \perp r$.

Now $\operatorname{Re} s^*P_r(A)^*P_r(A)r = 0$ for any $s \perp r$. If $\mathbb{K} = \mathbb{R}$, then $s^*P_r(A)^*P_r(A)r = 0$; otherwise, by replacement of s with the quantity (is), we obtain $\operatorname{Im} s^*P_r(A)^*P_r(A)r = 0$, and thus $s^*P_r(A)^*P_r(A)r = 0$ for all $s \perp r$. Thus $P_r(A)^*P_r(A)r$ is a scalar multiple of r, namely,

$$P_r(A)^*P_r(A)r = \frac{r^*P_r(A)^*P_r(A)r}{r^*r}r.$$

Thus r is a (right) singular vector of $P_r(A)$.

If r is a maximal singular vector (in the sense that its associated singular value is maximal among the singular values), then for all $v \in \mathbb{K}^N$,

$$\varphi_{n,\mathbb{K}}(A) \leq \frac{||P_r(A)v||}{||v||} \leq \frac{||P_r(A)r||}{||r||} = \psi_{n,\mathbb{K}}(A) \leq \varphi_{n,\mathbb{K}}(A),$$

and thus P_r is a minimizer for $\varphi_{n,\mathbb{K}}(A)$ and we are done. Otherwise, there exists $s \perp r$ a maximal singular vector for $P_r(A)$, and r is not a maximal singular vector.

Since A is normal, then $A = U\Lambda U^*$, with U unitary and Λ diagonal. Then U^*r is an eigenvector of $\Lambda^*\Lambda$, and is thus composed of a linear combination of standard unit basis vectors e_{i_j}, and similarly U^*s is composed of a set of vectors e_{i_k} all distinct from e_{i_j}. Since then r and s are contained in the linear spans of distinct (orthogonal) eigenspaces of A, it follows that $A^l r \perp A^m s$ for any l, m. Then (4b) reduces to

$$s^*P_r(A)^*P_r(A)s - r^*P_r(A)^*P_r(A)r \leq 0$$

or

$$r^* P_r(A)^* P_r(A) r \geq s^* P_r(A)^* P_r(A) s$$

which contradicts the maximality of s over r. □

The results of this section have the following significance. In certain cases such as when A is normal, a fixed polynomial preconditioning of degree $(s-1)$ exists independent of $r^{(0)}$, which has the same convergence bound as a cycle of GMRES(s). In such cases it is reasonable to seek a polynomial preconditioner to replace (2) so that inner products need not be computed. Of course, since (2) finds the best polynomial for each $r^{(0)}$, the *average* behavior of (2) may be better than the average behavior of the best polynomial preconditioner, though the worst case behavior for a cycle applied to a given vector is the same.

It is not clear whether $\psi_{n,\mathbb{K}}(A) = \varphi_{n,\mathbb{K}}(A)$ holds for general A, though numerical experiments suggest that it does hold for cases other than A normal and $n = 1$. The exact behavior of these functions is an open area of research.

The algorithm presented in §3 takes advantage of the cases when a fixed polynomial preconditioning of degree s can be found that yields a convergence rate as good as that of the restarted GMRES algorithm with restart frequency s.

3. Definition of the hybrid algorithm. We wish to define a modification of the restarted GMRES algorithm that allows for alternate means of calculating the polynomial P_n of (2) besides the least squares minimization of (2a), (2b).

To make use of the relevant inner product information from the Krylov spaces obtained from GMRES cycles in the history of a run, we define a fixed polynomial basis to be used to represent the Krylov space at each cycle. The use of a fixed polynomial basis simplifies the recording of Krylov space inner products for each cycle. In particular, we define the polynomials $\{p_i\}_{i=0}^{s}$ that are fixed independent of the cycle number, and use these to represent the Krylov space of each cycle by the matrix $\mathcal{P}_{s+1}(r) = [p_0(A)r \ldots p_s(A)r]$ and the associated inner product matrix $\mathcal{P}_{s+1}(r)^* \mathcal{P}_{s+1}(r)$. Means of choosing the polynomials so that the basis is well conditioned are described in [11]. Here we employ Chebyshev polynomials which possess a three-term recurrence.

The idea of the algorithm is as follows: given $r^{(ms)}$, the Krylov basis vectors $\mathcal{P}_{s+1}(r^{(ms)})$ may be generated, and then a decision can be made as to whether $r^{(ms+s)}$ is calculated by (2) or alternatively by some polynomial preconditioner represented in the basis $\mathcal{P}_{s+1}(r^{(ms)})$. The polynomial preconditioner may be based on information from the history of the run, in particular, the previous cycles that were performed in GMRES mode (2).

The structure of the algorithm is shown below. Here m denotes the cycle number, $\mathcal{S}$ denotes the set of cycles performed in GMRES mode, and the parameter t is a user-supplied tolerance, $0 \leq t \leq 1$.

1. Initialize. $m \leftarrow 0$.
2. Perform the stopping test.
3. Compute $r^{(s)}$ via a cycle of GMRES. $m \leftarrow 1$.
4. Determine well-conditioned basis polynomials $\{p_i\}$.
5. Determine $\mathcal{P}_{s+1}(r^{(0)})^* \mathcal{P}_{s+1}(r^{(0)})$ from the GMRES Arnoldi vector information from the first cycle (see below). Initialize $\mathcal{S} \leftarrow \{0\}$.
6. Perform the stopping test.
7. Select a polynomial preconditioner q, based on the information in $\{\mathcal{P}_{s+1}(r^{(is)})^* \cdot \mathcal{P}_{s+1}(r^{(is)})\}_{i \in \mathcal{S}}$, and represented in the basis $\{p_i\}$.
8. Compute $\tilde{r}^{(ms+s)} = [I - Aq(A)]r^{(ms)}$ and its norm.
9. Test: If $||\tilde{r}^{(ms+s)}||/||r^{(ms)}|| \leq [\max_{i \in \mathcal{S}} ||r^{(is+s)}||/||r^{(is)}||] \cdot (1-t) + [1] \cdot (t)$, then set $r^{(ms+s)} = \tilde{r}^{(ms+s)}$; else, add m to $\mathcal{S}$, calculate the inner products

$\mathcal{P}_{s+1}(r^{(ms)})^*\mathcal{P}_{s+1}(r^{(ms)})$, and then compute $r^{(ms+s)}$ via the minimal residual method (2) using the existing basis vectors $\mathcal{P}_{s+1}(r^{(ms)})$.

10. Form $u^{(ms+s)}$ corresponding to $r^{(ms+s)}$.
11. Increment m; go to 6.

The details of executing the minimal residual component of step 9 are found in [11]. In particular,

$$u^{(ms+s)} = u^{(ms)} + \mathcal{P}_s(r^{(ms)})[\mathcal{P}_s(r^{(ms)})^* A^* A \mathcal{P}_s(r^{(ms)})]^{-1} \mathcal{P}_s(r^{(ms)})^* A r^{(ms)},$$

$$r^{(ms+s)} = r^{(ms)} - A\mathcal{P}_s(r^{(ms)})[\mathcal{P}_s(r^{(ms)})^* A^* A \mathcal{P}_s(r^{(ms)})]^{-1} \mathcal{P}_s(r^{(ms)})^* A r^{(ms)}.$$

The parameter t controls how good the polynomial preconditioner is required to be in order to be used. If $t = 0$, then the polynomial preconditioner applied to the current cycle must do at least as well as the worst GMRES cycle in the history of the run; t set closer to 1 allows a worse preconditioner to be tolerated, while any $t < 1$ insures convergence unless GMRES(s) can stagnate for this matrix (i.e., $\psi_{s,\mathbb{K}}(A) = 1$). This equivalence of the convergence behavior of the new method to that of GMRES(s) assumes, of course, that the Krylov basis for each cycle is well conditioned; for a full discussion of such considerations, see [11].

The calculation of the inner product information of step 5 from the GMRES information of that first cycle requires a more detailed description. Let $AQ_s = Q_{s+1}H_{s+1}$, where H_{s+1} is the upper Hessenberg matrix from the Arnoldi sequence of the first GMRES cycle, with Q_{s+1} the associated matrix of Arnoldi vectors, $Q_{s+1}^* Q_{s+1} = I$. We seek a new basis $\tilde{Q}_s \equiv \mathcal{P}_s(r^{(0)})$ such that $A\tilde{Q}_s = \tilde{Q}_{s+1}T_{s+1}$, where T_{s+1} is upper Hessenberg (for the Chebyshev basis case, tridiagonal) and defines the set of recurrences associated with $\{p_i\}$ (see [11] for complete details). Let $\tilde{Q}_{s+1} = Q_{s+1}\tilde{T}$, so that $\tilde{T}$ is the change of basis matrix. We seek $\tilde{Q}_{s+1}^* \tilde{Q}_{s+1} = \tilde{T}^* \tilde{T}$. Note $\tilde{T}$ is upper triangular. After some manipulation we have

$$\tilde{T}T_{s+1} = H_{s+1}\begin{bmatrix} I_s \\ 0 \end{bmatrix}^* \tilde{T}\begin{bmatrix} I_s \\ 0 \end{bmatrix},$$

where I_s is the identity. By applying the standard unit basis vector e_i to the right side of each side of the equation, we obtain a recursion for the columns of $\tilde{T}$. Specifically, we let $\tilde{t}_i = \tilde{T}e_i$ and $T_{s+1} = \{t_{i,j}\}$, and then

$$t_{i+1,i}\tilde{t}_{i+1} = H_{s+1}\begin{bmatrix} I_s \\ 0 \end{bmatrix}^* \tilde{t}_i - \sum_{j=1}^{i} t_{j,i}\tilde{t}_j.$$

Finally, $\tilde{t}_1$ is a multiple of e_1 based on the scaling relationship between the initial basis vectors $Q_{s+1}e_1$ and $\tilde{Q}_{s+1}e_1$.

It should be emphasized that the algorithm presented thus far makes no restriction on the type of preconditioner used. Any type of preconditioner may be applied, and the mechanism of the algorithm insures that if the residual decrease is not adequate, then the cycle can be completed using GMRES. Furthermore, the cost of the algorithm when in GMRES mode is only slightly greater (about one SAXPY operation per step) than a standard GMRES-type algorithm for computing (2), due to the computation of $\tilde{r}^{(ms+s)}$.

We now define a particular preconditioner to be used in this study. This preconditioner is based on the inner product information from past GMRES cycles of the run and approximates the minimizing polynomial of $\varphi_{s,\mathbb{K}}(A)$.

Given a set of vectors $\{r_i\} \subseteq \mathbb{K}^N$, for $A \in \mathbb{K}^{N\times N}$, it is assumed that the inner product information contained in the matrices $K_{s+1}(r_i, A)^* K_{s+1}(r_i, A)$ is known. Then we note

$$\inf_{e_1^*c=1} \sup_{r_i} \frac{||K_{s+1}(r_i, A)c||}{||r_i||} \leq \inf_{e_1^*c=1} \sup_{r\in\mathbb{K}^N} \frac{||K_{s+1}(r, A)c||}{||r||}$$
$$= \inf_{P_s(0)=1} \sup_{r\in\mathbb{K}^N} \frac{||P_s(A)r||}{||r||} = \varphi_{s,\mathbb{K}}(A),$$

where $P_s(z) = \sum_{i=0}^{s}(e_{i+1}^*c)z^i$. Since each $F_i(c) = ||K_{s+1}(r_i, A)c||/||r_i||$ is convex (and in fact F_i^2 is quadratic), the function $F(c) = \sup_i F_i(c)$ is convex. Optimization routines may be used to find the minimizer of F numerically. Furthermore, when the set $\{r_i\}$ is sufficiently expanded, $\inf_{e_1^*c=1} F(c)$ becomes closer to $\varphi_{s,\mathbb{K}}(A)$.

The algorithm considered here, then, uses a polynomial based on the minimizer of $F = \sup_i F_i$, which yields a polynomial that in some sense is optimal for all previous GMRES cycles.

Let us consider for a moment the computational cost of this algorithm. Following the analysis techniques of [10], let w_{mv}, w_{sax}, and w_{dot} represent the computational cost on a given computer of the matrix-vector product $A \cdot v$, SAXPY $y \leftarrow y + \alpha x$, and dot product u^*v, respectively. The cost for an s-step cycle in polynomial preconditioning mode using Chebyshev basis polynomials is $[sw_{mv} + w_{\text{dot}} + (4s + 2)w_{\text{sax}}]$, while a cycle in GMRES mode adds $[(s + 1)(s + 2)/2]w_{\text{dot}} + (s + 1)w_{\text{sax}}$ to this. The number of cycles to converge to $||r^{(ms)}||/||r^{(0)}|| \leq \zeta$ is approximately $\log(\zeta)/\log((1 - t)\varphi_{s,\mathbb{K}}(A) + t)$, assuming $0 \leq t \leq 1$ for this analysis, since $||r^{(ms+s)}|| \leq ((1 - t)\varphi_{s,\mathbb{K}}(A) + t)||r^{(ms)}||$ is assured. If l cycles in GMRES mode are required, then the total work for solution is approximately

$$l\left[\frac{(s + 1)(s + 2)}{2}w_{\text{dot}} + (s + 1)w_{\text{sax}}\right]$$
$$+[sw_{mv} + w_{\text{dot}} + (4s + 2)w_{\text{sax}}]\frac{\log(\zeta)}{\log[(1 - t)\varphi_{s,\mathbb{K}}(A) + t]}. \tag{5}$$

For comparison purposes we consider the case when A is HPD. Then $\varphi_{s,\mathbb{K}}(A)$ may be approximated using Chebyshev polynomials to yield

$$\varphi_{s,\mathbb{K}}(A) \doteq \frac{1}{\cosh\left(s \log\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)\right)}, \tag{6}$$

where $\kappa = ||A|| \cdot ||A^{-1}||$ is the condition number of A. This approximation is most valid when the spectrum $\sigma(A)$ does not have large gaps.

For the purpose of this analysis we assume l is fixed independently of κ; this may not hold in practice, depending in part on the strategy used to determine the preconditioner. Let $s = c\kappa^{\xi}$ and assume κ is large. Now, if $\xi \geq \frac{1}{2}$, then (5) grows at least as fast as a multiple of κ. Otherwise, when $0 < \xi < \frac{1}{2}$, we expand by Taylor series for κ large and after some manipulation and dropping of low order terms, we obtain from (5) the estimated asymptotic total cost,

$$w_\infty \equiv ls^2\frac{w_{\text{dot}}}{2} + \frac{\kappa}{s}[w_{mv} + 4w_{\text{sax}}]\frac{\log\zeta^{-1}}{2(1-t)}$$
$$= lc^2\kappa^{2\xi}\frac{w_{\text{dot}}}{2} + \frac{\kappa^{1-\xi}}{c}[w_{mv} + 4w_{\text{sax}}]\frac{\log\zeta^{-1}}{2(1-t)}. \tag{7}$$

It is clear that setting $\xi = \frac{1}{3}$ minimizes the maximal exponent of κ and thus the growth rate of this quantity. In this case, s grows at a rate proportional to $\kappa^{1/3}$, and the total work estimate

w_∞ grows at a rate proportional to $\kappa^{2/3}$. Under these assumptions, minimizing the asymptotic total work estimate (7) over choices of the constant c yields after some manipulation

$$s = \left[\frac{w_{mv} + 4w_{\text{sax}}}{2w_{\text{dot}}} \frac{\log \zeta^{-1}}{l(1-t)} \kappa \right]^{1/3},$$

$$w_\infty = \frac{3}{2}(l w_{\text{dot}})^{1/3} \left[(w_{mv} + 4w_{\text{sax}}) \frac{\log \zeta^{-1}}{2(1-t)} \right]^{2/3} \kappa^{2/3}.$$

These results show that for the HPD case under these assumptions, it is desirable to let the restart frequency s for this algorithm grow in proportion to $\kappa^{1/3}$ to get best performance, and the resulting total work required (measured in vector operations) grows in proportion to $\kappa^{2/3}$.

This compares favorably with a total work growth rate proportional to κ for standard restarted GMRES with the best choice of s (see [10]), while it is not as good as the conjugate residual method whose growth rate is proportional to $\kappa^{1/2}$ due to its short recurrence. In other words, the new method under the given assumptions is able to recover part of the effective performance of short recurrence methods.

The algorithm presented here is similar to the hybrid GMRES algorithm of [16] but has significant differences. Both algorithms apply a cycle of GMRES to the linear problem and use the resulting polynomial as a preconditioner. However, if the polynomial is not adequate, the algorithm of [16] increases the value of s, in an attempt to form a better polynomial. On the other hand, the algorithm presented here keeps s fixed but uses more GMRES cycles to improve the preconditioner.

4. Numerical experiments. We now present the results of numerical experiments with the new algorithm. For these experiments we consider the model problem

$$-u_{xx}(x, y) - u_{yy}(x, y) + Du_x(x, y) = g(x, y) \quad \text{on } \Omega = [0, 1]^2,$$

$$u(x, y) = 1 + xy \quad \text{on } \partial\Omega.$$

Here D is constant, and g is chosen so that the true solution is $u(x, y) = 1 + xy$ on Ω. Five-point central differencing is applied, with uniform mesh spacing $h = 1/n_h$ in each direction. No preconditioning is used.

The mesh size for these experiments is $h = \frac{1}{128}$. Initial guess $u^{(0)} = 0$ is used, and a vector of random entries is used for b. In practice, a random vector may be used for $u^{(0)}$, as described in [16]. The stopping test $||r^{(n)}||/||r^{(0)}|| < \zeta = 10^{-8}$ is used.

The experiments are run in double precision on a Sun 4/330 Sparcstation with 25 Mhz clock speed. The optimization to find the polynomial preconditioner is performed by the GRG2 package of [13], [14], which uses a generalized reduced gradient technique for convex optimization. For each run, the approximation to the best polynomial for the previous cycle is used as the initial guess to the next call of the optimizer package to find an improved value.

The timings in Tables 1–6 are in seconds and the optimizer time is not counted. The case of $t = -\infty$ denotes the case when GMRES mode forced for every cycle. In these instances, the least squares polynomial (2) is used for each cycle, and the method reduces to the GMRES/Chebyshev basis algorithm described in [11], which typically gives the same iterates as standard GMRES at reduced cost. The performance values for these instances are indicative of the performance for the standard restarted GMRES algorithm for these cases.

Several conclusions may be drawn from these results (cf. [16], [17]):

1. In most cases only a couple of GMRES cycles are required to form an adequate preconditioner.

TABLE 1
Timings; case of $s = 20$.

Meth \ Dh:	0	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
$t = -\infty$	1703	405	255	196	200	199	213	199	174	273
$t = .2$	724	446	262	192	163	143	122	122	150	236
$t = .5$	559	446	255	186	159	139	118	118	146	228

TABLE 2
Total cycles/GMRES cycles; case of $s = 20$.

Meth \ Dh:	0	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
$t = -\infty$	143/143	33/33	21/21	16/16	16/16	16/16	17/17	16/16	14/14	22/22
$t = .2$	86/25	61/3	35/2	25/2	21/2	18/2	15/2	15/2	19/2	31/2
$t = .5$	62/24	63/2	35/2	25/2	21/2	18/2	15/2	15/2	19/2	31/2

TABLE 3
Timings; case of $s = 40$.

Meth \ Dh:	0	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
$t = -\infty$	1354	498	406	477	412	412	421	360	354	385
$t = .2$	748	404	294	253	229	201	188	187	201	244
$t = .5$	681	418	302	262	238	207	195	191	201	246

TABLE 4
Total cycles/GMRES cycles; case of $s = 40$.

Meth \ Dh:	0	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
$t = -\infty$	40/40	15/15	12/12	14/14	12/12	12/12	12/12	10/10	10/10	11/11
$t = .2$	37/12	25/2	17/2	14/2	12/2	10/2	9/2	9/2	10/2	13/2
$t = .5$	33/10	25/2	17/2	14/2	12/2	10/2	9/2	9/2	10/2	13/2

TABLE 5
Timings; case of $s = 60$.

Meth\ Dh:	0	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
$t = -\infty$	1343	603	665	740	798	758	763	617	604	671
$t = .2$	798	443	423	403	423	423	402	402	261	323
$t = .5$	733	424	383	320	301	281	240	241	261	323

TABLE 6
Total cycles/GMRES cycles; case of $s = 60$.

Meth \ Dh:	0	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
$t = -\infty$	21/21	9/9	10/10	11/11	12/12	11/11	11/11	9/9	9/9	10/10
$t = .2$	21/8	15/2	14/2	13/2	14/2	14/2	13/2	13/2	8/1	11/1
$t = .5$	31/1	16/1	14/1	11/1	10/1	9/1	7/1	7/1	8/1	11/1

2. The run times for the new method are typically better than those for the pure GMRES method (denoted by $t = -\infty$), sometimes by as much as a factor of three, which is a substantial savings. This is coupled with the fact that the GMRES/Chebyshev basis algorithm [11] used here for the $t = -\infty$ case is up to twice as fast as the standard Arnoldi-based GMRES implementation, leading to solution time reductions of up to a factor of six.

3. It is easier to determine a good polynomial when s is chosen larger. This is consistent with the results of [16]. Of course, if $s = d(A) = d(r^{(0)}, A)$, then the polynomial from the first cycle matches the spectrum of A exactly (in exact arithmetic).

4. For the cases for which only one GMRES cycle is performed, the result of the algorithm applied here is identical to that of the hybrid algorithm of [16], assuming the value of s is forced to be fixed for that algorithm. For the cases for which more than one GMRES cycle is performed, the algorithms are different. In these cases, the polynomial P_s from the first cycle of GMRES is found by this algorithm to be inadequate, and if used would lead to slower convergence or to divergence (cf. [17]). This is because a single GMRES cycle may yield insufficient information to insure that $||P_s(A)r||/||r|| \leq 1$ for all r, even though $||P_s(A)r^{(0)}||/||r^{(0)}|| \leq 1$ is imposed. Such problems may occur when $r^{(0)}$ lacks a representative distribution of eigencomponents. In such cases, the algorithm of [16] with fixed s may diverge.

5. In the HPD cases, many GMRES cycles are required to obtain an effective polynomial. This is due in part to the fact that for the HPD case it is especially critical to estimate the smallest eigenvalue accurately, which is difficult. Furthermore, the polynomial preconditioner used here does not take advantage of the fact that for the HPD case, effective polynomials may be found based solely on the endpoints of the spectrum. Therefore, the algorithm spends a large amount of time making the polynomial small throughout the interior of the spectrum. On the other hand, when the matrix is nonnormal, the behavior of polynomials on the spectrum is less determinative of the convergence behavior of the polynomial preconditioner, and the approach used here has greater merit.

6. For these problems, $s = 20$ generally gave the best timings. This indicates a tradeoff between choosing s large to keep small the required number of GMRES cycles for the preconditioner calculation, and choosing s small to avoid the need to do even a single very large GMRES cycle. Furthermore, the aspect of GMRES which makes its average performance better than its worst case performance may be more pronounced when s is large.

7. These experiments do not shed light on the growth rate of the best value of s as the problem size grows, since only one problem size is used. The earlier asymptotic analysis suggests that at least for the HPD case, the best value of s grows for larger problem sizes. However, this may not always occur in practice due to such factors as superlinear convergence of GMRES due to gaps in the spectrum of A, or the need for differing numbers of GMRES cycles as the problem size grows, factors not accounted for in the model given earlier.

8. Though the timings for the new algorithm are generally less than those for the $t = -\infty$ (pure GMRES(s)) case, the difference is not as dramatic as might be expected. This is because more overall cycles are often required by the hybrid algorithm: the true minimal residual algorithm for each cycle has an *average* performance which may be better than the average performance of a good polynomial preconditioner, though the worst case bounds may be the same. The reasons for this are a topic of further study.

Next, in Table 7 the execution times for the convex optimizer are compared to the total run times, for the $s = 20$ case shown in Table 2.

It should be emphasized that this study does not fully investigate the issue of fast convex minimization algorithms for this problem. Alternate algorithms might give faster results; this is a topic for further research. On the other hand, it can be expected that for sufficiently large problems, the time for the optimization relative to the time for the rest of the algorithm will be small, since the optimization does not require the use of long vectors of size N. Thus, for many practical problems, an extremely fast minimizer may not be necessary.

The optimization times shown in Table 2 are fairly small compared to the total time, except for the HPD ($Dh = 0$) case. Even for the $s = 60$ case, the optimizer time did not exceed approximately 20% of the total time, except for the HPD case. On the other hand, when A is HPD, as noted earlier, many cycles are executed in GMRES mode, requiring a large number of calls to the optimizer to revise the preconditioner. One possible remedy to avoid

TABLE 7
Optimizer times / total times (seconds); case of $s = 20$.

Meth \ Dh:	0	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
$t = .2$	407/1131	10/456	5/267	8/200	5/168	5/148	7/129	8/130	10/160	7/243
$t = .5$	2055/2614	6/452	5/260	7/193	5/164	5/144	7/125	8/126	10/156	7/235

such cases is to set an upper limit on the number of optimizations done, and when this number is exceeded the algorithm could revert to using standard restarted GMRES for every cycle. In any case, it should be emphasized that performing even a few optimizations yields a better polynomial than the simple first-cycle polynomial, which is employed by the fixed-s version of the algorithm of [16].

In short, the algorithm shows benefit in terms of CPU savings over restarted GMRES, particularly for large values of s. Even greater savings may be expected on computer architectures such as distributed memory machines for which inner product operations are particularly costly.

5. Conclusions. In this paper we have shown results on the effectiveness of replacing GMRES with a polynomial preconditioning, defined an algorithm for safely adapting between GMRES and polynomial preconditioning, and defined a new preconditioning which approximates the minimax polynomial preconditioning for a given matrix.

The algorithm is shown to be effective compared to restarted GMRES for a number of cases. Further research is needed to shed more light on the behavior of polynomial methods for nonsymmetric problems.

REFERENCES

[1] H. C. ELMAN, Y. SAAD, AND P. SAYLOR, *A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 840–855.

[2] 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.

[3] ———, *Orthogonal error methods*, SIAM J. Numer. Anal., 24 (1987), pp. 170–187.

[4] A. GREENBAUM, *Comparison of splittings used with the conjugate gradient algorithm*, Numer. Math. 33 (1979), pp. 181–194.

[5] A. GREENBAUM AND L. GURVITS, *Max-Min properties of matrix factor norms*, SIAM J. Sci. Comput., 15 (1994), pp. 348–358.

[6] A. GREENBAUM AND L. N. TREFETHEN, *GMRES/CR and Arnoldi/Lanczos as matrix approximation problems*, SIAM J. Sci. Comput., 15 (1994), pp. 359–368.

[7] R. A. HORN AND C. R. JOHNSON, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.

[8] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1964.

[9] W. JOUBERT, *Iterative methods for the solution of nonsymmetric systems of linear equations*, Report CNA-242, Center for Numerical Analysis, The Univ. Texas at Austin, Feb. 1990.

[10] ———, *On the convergence behavior of the restarted GMRES algorithm for solving nonsymmetric linear systems*, J. Numer. Linear Algebra Appl., to appear.

[11] W. D. JOUBERT AND G. F. CAREY, *Parallelizable restarted iterative methods for nonsymmetric linear systems. Part* I: *Theory*, *Part* II: *Parallel implementation*, Internat. J. Computer Math., 44 (1992), pp. 243–267.

[12] W. D. JOUBERT AND D. M. YOUNG, *Necesssary and sufficient conditions for the simplification of generalized conjugate gradient algorithms*, Linear Algebra Appl., 88/89 (1987), pp. 449–485.

[13] L. S. LASDON AND A. D. WARREN, GRG2 *User's Guide*, unpublished report, Department of Management Science and Information Systems, The University of Texas at Austin, Dec. 1989.

[14] L. S. LASDON, A. D. WARREN, A. JAIN, AND M. RATNER, *Design and testing of a generalized reduced gradient code for nonlinear programming*, ACM Trans. Math. Software, 4 (1978), pp. 34–50.

[15] T. A. MANTEUFFEL, *Adaptive procedure for estimation of parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 187–208.

[16] 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. 796–825.

[17] D. P. O'LEARY, *Yet another polynomial preconditioner for the conjugate gradient algorithm*, Linear Algebra Appl., 154–158 (1991), pp. 377–388.

[18] 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.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 440–459, March 1994

013

A COMPARISON OF PRECONDITIONED NONSYMMETRIC KRYLOV METHODS ON A LARGE-SCALE MIMD MACHINE*

JOHN N. SHADID† AND RAY S. TUMINARO‡

Abstract. Many complex physical processes are modeled by coupled systems of partial differential equations (PDEs). Often, the numerical approximation of these PDEs requires the solution of large sparse nonsymmetric systems of equations. In this paper the authors compare the parallel performance of a number of preconditioned Krylov subspace methods on a large-scale multiple instruction multiple data (MIMD) machine. These methods are among the most robust and efficient iterative algorithms for the solution of large sparse linear systems. In this comparison, the focus is on parallel issues associated with preconditioners within the generalized minimum residual (GMRES), conjugate gradient squared (CGS), biconjugate gradient stabilized (Bi-CGSTAB), and quasi-minimal residual CGS (QMRCGS) methods. Conclusions are drawn on the effectiveness of the different schemes based on results obtained from a 1024 processor nCUBE 2 hypercube.

Key words. linear systems, nonsymmetric, parallel algorithms, Krylov methods, preconditioners, multilevel methods, MIMD

AMS subject classifications. 35, 35Q35, 65, 65F10, 65F50, 68M20

1. Introduction.

A wide variety of complex physical systems are modeled by three-dimensional partial differential equations (PDEs). Often, numerical formulations of these PDE problems give rise to systems of large sparse nonsymmetric equations. Unfortunately, the slow asymptotic performance and large storage requirements of direct solvers make them impractical for the increasingly larger systems that scientists need to solve. For these systems robust and fast iterative solvers must be used.

In this paper, we focus on performance and implementation issues for preconditioned Krylov methods on large-scale multiple instruction multiple data (MIMD) machines. These machines utilize many independent processors, each with their own local memory and a message-passing capability for communicating between processors. The Krylov methods we consider (GMRES, CGS, Bi-CGSTAB, and QMRCGS) are relatively easy to program and work well on a wide variety of meaningful problems. A number of highly parallel preconditioners are implemented including classical methods such as Jacobi, Gauss–Seidel, and polynomial preconditioners, as well as more sophisticated (though sometimes more problematic) preconditioners such as multigrid and multilevel filtering. Based on results obtained on a 1024 processor nCUBE 2, a number of general conclusions are drawn about the effectiveness of the underlying Krylov methods as well as the parallel performance of the preconditioners.

2. Iterative solvers.

Krylov algorithms are among the fastest and most robust iterative solvers for a wide variety of applications (e.g., computational fluid dynamics). In this paper, a representative group of nonsymmetric Krylov solvers are considered: conjugate gradient squared (CGS) [26], biconjugate gradient stabalized (Bi-CGSTAB) [29], generalized minimum residual method, (GMRES) [21], and a transpose-free variant of the quasi-minimum residual method (QMR) [8] referred to in this paper as QMRCGS.

The basic process in each of the Krylov solvers is the projection of the original set of equations onto a subspace. The resulting set of equations is then solved and the subspace

*Received by the editors May 11, 1992; accepted for publication (in revised form) February 16, 1993. 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 contract DE-AC04-76DP00789.

†Parallel Computational Sciences Division, Sandia National Laboratories, Albuquerque, New Mexico 87185 (jnshadi@cs.sandia.gov).

‡Applied and Numerical Mathematics Division, Sandia National Laboratories, Albuquerque, New Mexico 87185.

solution is used to correct the solution to the original problem. The Bi-CGSTAB, CGS, and QMRCGS methods are based on the Lanczos biorthogonalization method. Their corresponding projection operators are computed using short recurrence relations. Unfortunately, the resulting projection operator is not orthogonal. By contrast, GMRES is based on the Arnoldi method. Its projection operator is much more expensive to compute (in terms of time and storage) as short recurrences cannot be used. However, the corresponding projection operator is orthogonal. To avoid excessive growth in storage and computations, a restarted version of GMRES is considered. Specifically, the iteration is terminated after a modest number of steps and then reinitiated using the previous solution as an initial guess. We note that all the methods considered in this paper are transpose-free schemes. In many engineering applications, the matrix-vector product corresponds to a complex and lengthy set of subroutines (where the the matrix is not explicitly stored). Thus, methods that do not require the use of the transpose of a matrix are more generally applicable. Figures 1–4 depict preconditioned (see §3) versions of CGS, Bi-CGSTAB, GMRES, and QMRCGS methods corresponding to our implementations.

```
CGS(x_0, A, b )
    r_0 = b − Ax_0;    q_0, p_{-1} = 0;
    Choose r̃_0 so that r_0^T r̃_0 ≠ 0;
    ρ_{-1} = 1;    n = 0;
    while (not converged) do
        ρ_n = r̃_0^T r_n;    β = ρ_n / ρ_{n-1};
        u = r_n + βq_n;
        p_n = u + β(q_n + βp_{n-1});
        Solve: M p̂ = p_n;        /* M is the preconditioning matrix. */
        v = A p̂;
        α = ρ_n / (r̃_0^T v);
        q_{n+1} = u − αv;
        Solve: M û = (u + q_{n+1});
        r_{n+1} = r_n − αA û;
        x_{n+1} = x_n + α û;
        n = n + 1;
    end
end
```

FIG. 1. CGS *Algorithm.*

2.1. Parallel issues. In this comparison we report on the relative performance of parallel solvers developed for systems of equations arising from PDE discretizations on regular two-dimensional domains. Specifically, parallelization is accomplished by spatially decomposing the computational domain and assigning subdomains to individual processors. Each processor updates only those variables internal to its subdomain. In addition to these internal variables, each processor contains internal boundary components. The internal boundaries are copies of components that are updated on other processors, but which are needed to compute local operations. Currently, we have implemented only rectangular subdomains. In this case, each processor contains an extra border around its subdomain that contains the additional variables. Similar to the variables, the discrete equations are also partitioned among the processors. In particular, the matrix is stored using a parallel generalization of the standard modified sparse row (MSR) format [20]. This format requires that two arrays be maintained: one array to store the nonzeros of the matrix and the other array to describe their location. A parallel version of this format is defined as follows. Let us assume (for clarity of presentation) that $\frac{N}{P}$ variables have been assigned to P processors. We decompose the matrix into P submatrices (each with $\frac{N}{P}$ rows) such that each processor is assigned a submatrix whose nonzero columns correspond

```
Bi-CGSTAB(x_0, A, b)
    r_0 = b − Ax_0;    v_0, p_0 = 0;
    Choose r̃_0 so that r_0^T r̃_0 ≠ 0;
    ρ_0, α, ω_0 = 1;    n = 1;
    while (not converged) do
        ρ_n = r̃_0^T r_{n−1};    β = (ρ_n/ρ_{n−1})(α/ω_{n−1});
        p_n = r_{n−1} + β(p_{n−1} − ω_{n−1} v_{n−1});
        Solve: M p̂ = p_n;        /* M is the preconditioning matrix. */
        v_n = A p̂;
        α = ρ_n/(r̃_0^T v_n);
        s = r_{n−1} − α v_n;
        Solve: M ŝ = s;
        t = A ŝ;
        ω_n = (t^T s)/(t^T t);
        x_n = x_{n−1} + α p̂ + ω_n ŝ;
        r_n = s − ω_n t;
        n = n + 1;
    end
end
```

FIG. 2. Bi-CGSTAB *Algorithm.*

```
GMRES(x_0, m, A, b)
    while (not converged) do
        /* Arnoldi process */
        r_0 = (b − Ax_0);
        β = √(r_0^T r_0);    v_1 = r_0/β;
        for j = 1, ..., m  do
            w = AM^{−1} v_j        /* M is the preconditioning matrix */
            for i = 1, ..., j  do    h_{i,j} = v_i^T w;    end
            v̂_{j+1} = w − Σ_{i=1}^{j} h_{i,j} v_i;
            h_{j+1,j} = √(v̂_{j+1}^T v̂_{j+1});    v_{j+1} = v̂_{j+1}/h_{j+1,j};
        end
        Let y_m be the value of y minimizing ||βe_1 − Hy||_2;
                                    /* H is (m + 1) × m upper Hessenberg
                                       matrix with nonzeros given by h_{i,j}
                                       and e_1 = [1, 0, ..., 0]^T. */
        x_0 = x_0 + M^{−1} V_m y_m        /* V_m = [v_1, ..., v_m]. */
    end
```

FIG. 3. *Restarted* GMRES *Algorithm (Gram–Schmidt orthogonalization).*

to the variables stored in the processor. Each submatrix is then stored using an MSR format in conjunction with a local numbering of the variables. More details of the distributed sparse matrix and the subdomain partitioning can be found in [24].

The majority of the solvers are implemented using only basic linear algebra subroutine (BLAS) kernels such as dot products and vector addition routines (supplied and optimized by the vendor, nCUBE). Additionally, a matrix-vector kernel is needed. Since this kernel is only a few lines long, it was written in assembly code (with the help of the vendor). Our implementation of this kernel runs at 1.3 Mflops per node. This performance for an indirectly addressed sparse matrix-vector multiply compares well with the maximum nCUBE 2 double

QMRCGS(x_0, A, b)
$r_0^{\mathrm{CGS}} = b - Ax_0$;
$p_0 = u_0 = M^{-1} r_0^{\mathrm{CGS}}$; /* M is the preconditioning matrix */
$v_0 = Ap_0$; $\tau_0 = ||r_0^{\mathrm{CGS}}||$;
Choose $\tilde{r}_0$ so that $\tilde{r}_0^T r_0^{\mathrm{CGS}} \neq 0$;
$\rho_0 = \tilde{r}_0^T r_0^{\mathrm{CGS}}$;
$d_0 = 0$; $\mathcal{V}_0 = 0$; $\eta_0 = 0$;
for $n = 1, 2, \ldots$ **do**
  $\sigma_{n-1} = \tilde{r}_0^T v_{n-1}$; $\alpha_{n-1} = \rho_{n-1}/\sigma_{n-1}$;
  $q_n = u_{n-1} - \alpha_{n-1} M^{-1} v_{n-1}$;
  $r_n^{\mathrm{CGS}} = r_{n-1}^{\mathrm{CGS}} - \alpha_{n-1} A(u_{n-1} + q_n)$;
  for $m = 2n - 1, 2n$ **do**
    if m even **then**
      $\omega_{m+1} = ||r_n^{\mathrm{CGS}}||$; $y_m = q_n$;
    else
      $\omega_{m+1} = \sqrt{||r_{n-1}^{\mathrm{CGS}}||\,||r_n^{\mathrm{CGS}}||}$; $y_m = u_{n-1}$;
    endif
    $\mathcal{V}_m = \omega_{m+1}/\tau_{m-1}$; $c_m = 1/\sqrt{1 + \mathcal{V}_m^2}$;
    $\tau_m = \tau_{m-1} \mathcal{V}_m c_m$; $\eta_m = c_m^2 \alpha_{n-1}$;
    $d_m = y_m + (\mathcal{V}_{m-1}^2 \eta_{m-1}/\alpha_{n-1}) d_{m-1}$;
    $x_m = x_{m-1} + \eta_m d_m$;
    if (converged)**stop**;
  end
  $\rho_n = \tilde{r}_0^T r_n^{\mathrm{CGS}}$; $\beta_n = \rho_n/\rho_{n-1}$;
  $u_n = M^{-1} r_n^{\mathrm{CGS}} + \beta_n q_n$;
  $p_n = u_n + \beta_n(q_n + \beta_n p_{n-1})$;
  $v_n = Ap_n$;
end

FIG. 4. QMRCGS *Algorithm.*

precision computation rate of about 2.5 Mflops per node[1] [23], [24]. The only communication procedures are a local exchange border routine (used by the sparse matrix-vector multiply requiring four messages to be exchanged per processor) and a global summation routine (used in the dot product algorithm requiring $\log_2(P)$ messages to be exchanged per processor on a hypercube with P total processors).

Parallel versions of Bi-CGSTAB, CGS, and the QMRCGS require only parallel implementations of the above-mentioned kernels. However, for GMRES the orthogonalization procedure needs to be reevaluated. Typically, GMRES orthogonalization is performed by a modified Gram–Schmidt process on scalar and vector computers. This choice is motivated by the superior stability properties of the modified scheme relative to the classical Gram–Schmidt method. However, from a parallel efficiency perspective, the classical Gram–Schmidt algorithm is superior. Specifically, $m \log_2 P$ messages per processor are needed to orthogonalize

[1] Unless noted otherwise, the results we present are for double precision computations using the assembly-coded matrix-vector multiply routine.

a vector with respect to m previous vectors by a modified Gram–Schmidt method. In contrast, the classical Gram–Schmidt method requires only $\log_2 P$ messages per processor (though these messages are now of size m). For large Krylov subspace dimensions, m, this difference can be significant. Both orthogonalization schemes have been implemented on the nCUBE 2.

Finally, we note that redundant work is performed (on all processors) in our implementation of the Hessenberg solver. That is, each processor performs the plane rotations corresponding to the factorization of the Hessenberg matrix along with the resulting triangular solve. Though this might appear to be inefficient, the actual time required for these operations is insignificant (for example less than .3% of the total solution time for the results given in Table 1, §4) and thus there is no advantage in parallelizing this step.

3. Preconditioners. It is well known that the overall performance of Krylov methods can be substantially improved when one uses preconditioning [6]. The basic idea is that instead of solving the system $Ax = b$, the system $AM^{-1}y = b$ is solved where M^{-1} is an approximation to A^{-1} and is easily computed. Since only matrix-vector products are needed, it is not necessary to explicitly form AM^{-1} (only a subroutine to solve $Mv = y$ is needed). We note that the preconditioning described here corresponds to "right" preconditioning and that it is also possible to precondition on the "left" (i.e., $M^{-1}A$). In this paper, only right preconditioning is considered as the comparisons are somewhat more straightforward. Specifically, when left preconditioning is used the computed residual corresponds to a preconditioned residual. Thus, if convergence is based on the size of the residual, changing the preconditioner effectively changes the convergence criteria.

Currently, a wide variety of preconditioners are available. The proper choice of preconditioner is both computer (or architecture) dependent as well as problem dependent. This research continues in a number of areas. It is, therefore, impossible to give a complete survey of all preconditioners. Instead, we attempt to investigate some of the more common preconditioners currently in use. It is expected that further advances in preconditioners will yield improvements in terms of both robustness and performance.

3.1. Local preconditioners. In this section we consider preconditioners that use information only within a local neighborhood to update the value of the solution at a grid point. This local data dependency is the result of the numerical approximation method (e.g., finite difference or finite element techniques) as well as the particular iterative procedure that is used as a preconditioner.

Classical iterative schemes. Two classical schemes are considered in this paper: Jacobi and Gauss–Seidel. Jacobi preconditioning amounts to rescaling the matrix and is thus easily parallelizable. For Gauss–Seidel, however, the ordering of the points must be considered. On sequential computers, points are often updated in an order that corresponds to counting across the rows of the grid (natural ordering). On vector and parallel machines this ordering is inefficient due to data dependencies. Instead, a "multicolor" ordering of the grid points is used that partitions the grid points into disjoint sets [1], [31]. These sets are selected so that update operations for each grid point are independent of all other grid points in the same set; thus points within a set can be updated in parallel. Of course, different orderings imply different numerical convergence properties. Thus, there is often a tradeoff between parallelism and rate of convergence.

Polynomial preconditioners. These methods use a polynomial expansion in the matrix A to approximate A^{-1}. Since these methods require only the standard matrix-vector kernel, no new communication routines are necessary to produce a parallel polynomial preconditioner. The major difficulty is choosing the polynomial coefficients properly. One choice based on

$$\min_{\alpha_i's} \; ||I - P(A)A||_2, \tag{1}$$

where

$$M^{-1} = P(A) = \sum_{i=0}^{n} \alpha_i A^i$$

leads to the least squares polynomial. The α's are easily approximated when an approximation to the convex hull of the spectrum of A, $\sigma(A)$, is available (e.g., within GMRES) [18]. Since the convex hull is not always easily available, we approximate (1) by assuming that all the eigenvalues of A lie on the real axis between 0 and $||A||_\infty$. When A is close to symmetric positive definite (SPD), this approximation is quite good.

A second choice for the $\alpha's$ corresponds to first defining

$$G \equiv I - \frac{1}{\beta} A \tag{2}$$

and using a truncated Neumann series

$$A^{-1} = \frac{1}{\beta}(I - G)^{-1} \approx \frac{1}{\beta} \sum_{i=0}^{n} G^i \tag{3}$$

[7], [30]. To ensure convergence of the infinite series, β must be chosen so that the spectral radius of G is less than one. For a real positive definite matrix A, this corresponds to

$$\beta > \frac{1}{2} \max_{\lambda \in \sigma(A)} \frac{|\lambda|^2}{\mathrm{Re}(\lambda)}, \tag{4}$$

where $\mathrm{Re}(\lambda)$ is the real part of λ. When A is SPD, (4) reduces to $\beta > \frac{\lambda_{\max}}{2} \approx \frac{||A||_\infty}{2}$. Rather than computing β from (4) for the nonsymmetric case, we use $\beta = 1.1||A||_\infty/2$, and have found this suitable for a wide range of problems.

Finally, we note that in our implementation, a row sum scaling is first performed (i.e., $a_{ij} = a_{ij} / \sum_{j=1}^{N} ||a_{ij}||$) before the polynomial preconditioner is constructed for the resulting matrix.[2]

3.2. Global preconditioners. Global preconditioners use information across the domain to update each point. In PDE discretizations this corresponds to using information from distant grid points to update an unknown. Since A^{-1} is dense in most applications (i.e., the solution at a point depends on information from all points), it is not surprising that global schemes often converge in many fewer iterations than local ones. Unfortunately, global preconditioners are often much harder to implement and have a somewhat more limited range of applicability. In this paper, we discuss alternating line Gauss–Seidel, incomplete LU factorization (ILU), multigrid, and multilevel filtering.

Alternating line Gauss–Seidel. We consider the alternating line Gauss–Seidel method [31]. Specifically, the grid is colored into red and black horizontal lines and one iteration of line Gauss–Seidel is performed. The procedure is then repeated with vertical lines. To update a line, a matrix containing all coupling within that line is factored. For five-point discretization of a single PDE this corresponds to factoring a tridiagonal matrix. Thus, a series of tridiagonal (or block tridiagonal) matrices must be solved in one direction and then solved in the other direction. Our parallel implementation proceeds as follows. Consider a

[2]Row sum scaling is explicitly performed on the matrix before using all of the preconditioners except the Jacobi scheme (where diagonal scaling is already incorporated) and the multigrid method where consistent scaling must be used on different grid levels to properly implement multigrid.

$p \times p$ array of processors that corresponds to an $n \times n$ computational grid. We perform p line solves in parallel (assuming $n \geq 2p$) until all the line solves in the x direction are complete. We then repeat the procedure for the y direction solves. Thus, each line solve is assigned p processors. A multitude of parallel tridiagonal algorithms have been studied [10], [13]. In our implementation we use block cyclic reduction. This algorithm proceeds by eliminating all the even unknowns within a line. The remaining unknowns are renumbered and the resulting even unknowns are eliminated. This process is repeated until only two unknowns remain (which can be solved directly). Since each of the $\log_2 n$ stages in the elimination process requires two messages to be communicated, a total of $2 \log_2 n$ messages are sent during the process. It should be noted that our implementation of the tridiagonal solvers is fairly crude. In part, this explains the poor results that are presented in §4. In particular, a number of issues must be addressed to improve the tridiagonal solver: the slower serial performance of cyclic reduction, inactive processors, and high communication requirements. There are, of course, many other possibilities for solving tridiagonal systems in parallel. One could keep a full line in each processor and solve all the tridiagonals in parallel. However, this approach would require a matrix transpose algorithm to perform the line solves in the other direction as well as require that the grid has sufficiently many points in each direction (e.g., at least 2048 grid lines in each direction when using a system with 1024 processors). Overall, we believe that improvements can be made to our tridiagonal solver (perhaps even by a factor of three or more). Even with significant improvements, however, the line scheme would still lag behind the other preconditioners (see §4).

ILU. We consider incomplete factorization as a preconditioner [15]. An incomplete factorization can be thought of as performing a Gaussian elimination procedure without storing all of the elements (i.e., restricting the fill-in). In our implementation we require the approximate LU factors to have the same sparsity pattern as in A. Most ILU schemes use a natural ordering of the unknowns.[3] Since standard ILU does not parallelize well, we consider a local ILU method (LILU) in which the boundary interface values for each subdomain are taken as known quantities. Each processor then appends a set of boundary interface equations to its local sparse matrix. These equations have unit diagonal with a right-hand side corresponding to the boundary interface value. An ILU factorization is performed on the expanded matrix and the triangular solves are used to generate an approximate inverse within each subdomain. It should be noted that the LILU method requires the same type of communication as the polynomial preconditioners, i.e., at each iteration updated interface values from neighboring processors must be communicated. One disadvantage of this approach is that the resulting preconditioner may yield slower convergence as it now uses information only within its subdomain. More sophisticated techniques that globally couple the interface equations exist (e.g., [25], [27]). We are currently investigating these approaches.

Multigrid. Multigrid methods are among the fastest iterative solvers for elliptic PDE problems [3]. One iteration of a simple multigrid V cycle consists of smoothing steps (e.g., Jacobi, Gauss–Seidel) along with projection and interpolation operations on a hierarchy of grids with different mesh sizes. In our Krylov solvers we use one multigrid V cycle as a preconditioner (recursively coarsening the grid down to a 4×4 grid). Coarse grid discretizations are obtained using the same scheme (in fact the same subroutine) as for the fine grid and storing the coarse grid operators using the MSR format. Within the V cycle, bilinear interpolation and full weighting restriction is used to transfer between grids. Unless otherwise stated, it should be assumed that one Gauss–Seidel iteration is used before and another Gauss–Seidel

[3] Multicolor orderings are also possible. However, the overall convergence is often much less reliable for hard problems than with the natural ordering scheme [4].

iteration is used after the coarse grid correction.[4]

Multilevel filtering. Motivated by the success of multigrid and hierarchical basis preconditioners, the multilevel filtering preconditioner [14] (similar to the BPX preconditioner, see [2]) approximately decomposes the residual into different frequency components, selectively scales these components, and then recombines them to form the preconditioned residual. In particular, a projection operator is used to project the residual onto a series of coarser grids. The projected residual on each grid is then scaled by differing amounts. This scaling represents an approximate inverse in Fourier space and is chosen optimally for the Poisson equation. Finally, the residual components are interpolated and combined to form the preconditioned residual. From an implementation perspective, multilevel filtering corresponds to a multigrid V cycle where the relaxation operation is replaced by a subroutine that scales the right-hand side, and the projection and interpolation operators are applied to the right-hand side (as opposed to the residual and solution).

In both the multigrid and the multilevel filtering preconditioners, each grid is processed in sequence. Each grid is partitioned among the processors corresponding to a particular decomposition of the domain. On coarser grids (where there are fewer grid points than processors) the subdomain assigned to a processor may not contain grid points. In this case the processor simply performs redundant work, in parallel, to the useful work proceeding on other processors. It should be noted that when a binary reflected Gray code is used in conjunction with the multigrid algorithm, not only are adjacent subdomains assigned to neighboring processors, but on coarse grids (with fewer points than processors) it is possible to shuffle the data (using nearest neighbor communication) so that only nearest neighbor communication is needed to exchange boundary information between those processors that contain valid grid points. This is due to the extra interconnections present on a hypercube that are not available on a mesh-connected architecture [5].

4. Numerical results. The remainder of this paper considers the performance of the solvers and preconditioners, discussed in the previous sections, on massively parallel machines. In this examination we focus on two questions:

- Which methods and preconditioners are best suited to parallel machines?
- What performance can be expected from current massively parallel MIMD machines?

To perform this evaluation, a number of test problems have been solved. In the process of our numerical experiments, a significant effort has been made to accurately reflect the relative performance of the different schemes. That is, numerous tests and variants (different algorithms, implementations, and convergence criteria) have been examined in an effort not to favor any one method over another. However, given the nature of the task, these results (as with any comparisons of this kind) should only be taken as an approximate guide of expected performance on similar problems. A 1024 processor nCUBE 2 hypercube was used for all of the tests. This machine is a distributed-memory MIMD machine with 4 Mbytes of memory per node and a maximum double precision computation rate of about 2.5 Mflops per node. We describe below the four test problems used in this paper.

PROBLEM 1.

$$-\frac{\partial^2 u}{\partial x^2}+\frac{\partial u}{\partial x}+(1+y^2)\left(-\frac{\partial^2 u}{\partial y^2}+\frac{\partial u}{\partial y}\right)=f(x,y) \tag{5}$$

defined on the unit square with Dirichlet boundary conditions and right-hand side such that

$$u(x,y)=e^{x+y}+x^2(1-x)^2\ln(1+y^2). \tag{6}$$

[4]Numerous experiments were performed with different smoother combinations. For isotropic problems, two Gauss–Seidel iterations are typically among the most efficient. Only one V cycle was chosen for preconditioning. Obviously, as more V cycles are used, the more the method behaves like pure multigrid.

This convection-diffusion PDE is taken from [28] and is discretized using central differences. The resulting algebraic system is positive definite and represents a relatively easily solved nonsymmetric system.

PROBLEM 2.

$$(7)\quad -[b(x,y)u_x]_x - [c(x,y)u_y]_y + [b(x,y)\ c(x,y)u]_x + [d(x,y)u]_y + e(x,y)u = f$$

defined on the unit square with

$$\begin{aligned} b(x,y) &= e^{-xy}, & c(x,y) &= \beta(x+y), \\ d(x,y) &= \gamma(x+y), & e(x,y) &= (1+xy)^{-1}, \\ \beta &= .01, & \gamma &= .01, \end{aligned}$$

Dirichlet boundary conditions and right-hand side such that the solution is given by

$$(8)\qquad u(x,y) = xe^{xy}\sin(\pi x)\sin(\pi y).$$

This PDE is anisotropic in that the amount of diffusion varies significantly in the different coordinate directions. In addition, by varying the magnitude of the γ parameter it is possible to produce a more strongly convected system. The problem is taken from [26] and is discretized via central differences.

PROBLEM 3.

$$(9)\quad \begin{aligned} \frac{\partial^2\Omega}{\partial x^2} + \mathrm{Ax}^2\frac{\partial^2\Omega}{\partial y^2} &= \Pi_1\frac{\partial\Phi}{\partial x} + \Pi_2\frac{\partial\Phi}{\partial y}, \\ \frac{\partial^2\Psi}{\partial x^2} + \mathrm{Ax}^2\frac{\partial^2\Psi}{\partial y^2} &= \frac{-\mathrm{Ax}^2}{4}\Omega, \\ \mathrm{Ax}\frac{\partial\Psi}{\partial y}\frac{\partial\Phi}{\partial x} - \mathrm{Ax}\frac{\partial\Psi}{\partial x}\frac{\partial\Phi}{\partial y} &= \frac{\partial^2\Phi}{\partial x^2} + \mathrm{Ax}^2\frac{\partial^2\Phi}{\partial y^2}. \end{aligned}$$

This PDE system corresponds to the incompressible Navier–Stokes and thermal energy equations (given in stream-vorticity formulation) describing buoyancy-induced natural convection in an inclined two-dimensional rectangular cavity over the computational domain $(-1, 1) \times (-1, 1)$. In the above equations,

$$(10)\qquad \Pi_1 \equiv -\mathrm{Ax}\frac{\mathrm{Ra}\cos\Theta}{2}, \qquad \Pi_2 \equiv (\mathrm{Ax})^2\frac{\mathrm{Ra}\sin\Theta}{2},$$

where Ax is the aspect ratio of a rectangular cavity, Θ is the angle of inclination, and Ra is the Rayleigh number. In this formulation the Prandtl number has been assumed to be infinite so that the inertial terms in the momentum equation can be neglected. In our specific example problem, we take $\mathrm{Ax} = 1$, $\Theta = \frac{\pi}{2}$, and $\mathrm{Ra} = 10^3$. Dirichlet boundary conditions are used for the stream function, Ψ, a mixed Dirichlet/Neumann condition for temperature, Φ, and a first-order approximation of the stream function-vorticity relationship at the boundary is used for the vorticity boundary conditions (see [22]). A discrete nonlinear set of equations is obtained using central difference approximations. The resulting nonlinear system is solved by Newton's method requiring a series of linear subproblems to be solved. As the Rayleigh number is increased the system becomes more strongly convected and nonlinear.

PROBLEM 4.

$$(11)\qquad -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} + \gamma\left(x^2\frac{\partial u}{\partial x} + y^2\frac{\partial u}{\partial y}\right) - \beta u = f(x,y)$$

defined on the unit square with Dirichlet boundary conditions and right-hand side such that the vector of all ones is the exact solution. This problem is discretized using centered differences on a uniform mesh, the parameter $\gamma = 100$, and the parameter β is varied to make the problem more indefinite.

4.1. Krylov solvers. In this subsection we focus on the four Krylov solvers. Due to the large number of possible preconditioner/solver combinations, we limit ourselves to polynomial preconditioning and use the notation LSn to denote an n term least-squares polynomial expansion. A similar notation is used for the Neumann series. Studies of the various preconditioners are deferred to the next section.

Before comparing Krylov solvers, it is necessary to first address two algorithmic issues associated with the GMRES method. The first issue (unique to the parallel environment) is the choice of orthogonalization procedure: classical Gram–Schmidt or modified Gram–Schmidt. The second issue corresponds to choosing the Krylov subspace size so that the convergence rate and the time per iteration are balanced to yield the fastest overall solution time (see §2). Figure 5 illustrates the GMRES results using both modified and classical Gram–Schmidt orthogonalization with a variety of subspace sizes for Problem 1. The results shown in the figure are typical of GMRES behavior. Specifically, we have not encountered a PDE problem that required significantly fewer GMRES iterations using modified Gram–Schmidt as opposed to classical Gram–Schmidt.[5] Thus, when comparing CPU times, the classical Gram–Schmidt procedure requires less time per iteration and therefore outperforms the modified Gram–Schmidt procedure (1.5 times faster for $m = 150$ in Fig. 5). Unless otherwise stated, the GMRES results in the remainder of the paper correspond to using classical orthogonalization. When we wish to distinguish between the two schemes, we use the notation GMRES^{C} and GMRES^{M} to denote the use of classical and modified Gram–Schmidt, respectively. While our examples do not indicate a need for a more stable orthogonalization procedure, it may be worthwhile to consider some combination of classical and modified Gram–Schmidt to obtain better stability properties without significantly increasing the time per iteration. We, however, do not pursue this topic any further.

Unfortunately, the situation is not as clear for the choice of Krylov subspace size. In particular, the optimal size is problem dependent, preconditioner dependent, and grid size dependent. Thus, it is difficult to find a nearly optimal subspace dimension without experimentation. For the problem corresponding to Fig. 5, the optimal subspace size is approximately 50 for the least-squares preconditioner, while the Neumann series works best with a subspace size of 90. From our experience we have generally found that a Krylov subspace size of 64 works well on a variety of problems. However, there are some situations that require bigger subspace sizes to reduce CPU time and in some cases to obtain convergence. In the remainder of the comparison we attempt to present representative results for the restarted GMRES method by selecting appropriate values for m.

We now consider the performance of the different Krylov algorithms using representative preconditioners. In these examples we select the initial guess as the zero vector and take the $\tilde{r}_0$ vector as a random vector in the CGS, Bi-CGSTAB, and QMRCGS methods. To determine convergence we require the residual normalized by $||A||_\infty$ to be less than 10^{-6}. This choice was influenced by the nonlinear problem, since requiring a significant reduction in the initial residual can be problematic when the initial guess is quite accurate. In the CGS and Bi-CGSTAB solvers we use the naturally occurring $||r||_2$ measure of the residual, and in the GMRES method we use the estimate suggest by Saad and Schultz [21]. In QMRCGS, we use

[5] It is possible, however, to create linear systems illustrating the superior numerical properties of modified Gram–Schmidt.

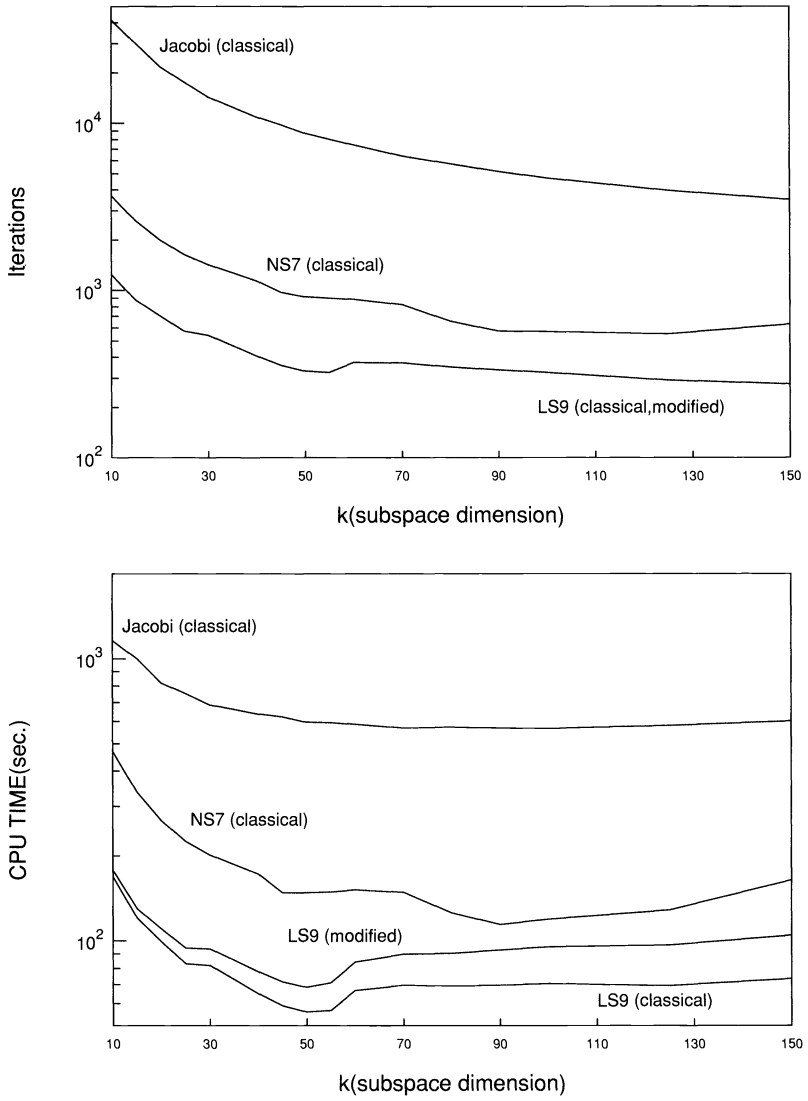


FIG. 5. *Convergence of* GMRES *for Problem* 1 *on a* 256 × 256 *grid with* 256 *processors.*

the bound $||r_m|| \leq \sqrt{m+1}\tau_m$ to estimate this quantity at the mth iteration as suggested by Freund [8]. After satisfying the above criteria we compute the value of the true residual to verify convergence of the solver.

In Tables 1–4 we present timings for the four parallel Krylov methods on the different PDE test problems. The times given in the tables correspond to the time needed for the solution algorithm as well as the scaling. However, it should be noted that the time to create the matrices (whose entries are computed locally in parallel on each processor) is insignificant. With the possible exception of GMRES using modified Gram–Schmidt orthogonalization, all the solvers are highly parallelizable. In Table 1, speedups are given corresponding to Problem 1. Most of the solvers attain speedups between 5 and 6 where speedup is defined relative to the 128-processor run time as

$$S_p = \frac{T_{128}}{T_p}$$

and T_p is the time required using p processors. These speedups compare quite favorably with the theoretical maximum of 8 (especially when one takes into account that the grid size is quite modest compared to a three-dimensional application). Unfortunately, the GMRES method using modified Gram–Schmidt attains a speedup of only 2.6 due to the large number of message startups. In the remainder of the tables we consider only the classical GMRES method. It should also be noted that the QMRCGS, CGS, and the Bi-CGSTAB methods are rather sensitive to roundoff error. As a consequence, the number of iterations required for convergence changes as the number of processors changes. This behavior explains the apparent superlinear speedup shown in Table 1. If the speedups for CGS and Bi-CGSTAB are calculated using the time per iteration (instead of entire solution time), the superlinear speedups do not appear.

TABLE 1

Time (CPU *seconds*) *and speedups* (*relative to* 128 *processors*) *using* LS3 *preconditioning on a* 512 × 512 *grid for Problem* 1. GMRES *using a Krylov subspace size of* 64 *before restarting.*

Procs\ Solver	CGS		Bi-CGSTAB		QMRCGS		$GMRES^C$		$GMRES^M$	
	Time	S_p	Time	S_p	Time	S_p	Time	S_p	Time	S_p
128	72.1	1.0	59.5	1.0	66.0	1.0	181.9	1.0	215.3	1.0
256	34.5	2.1	28.2	2.1	35.3	1.9	96.2	1.9	135.6	1.6
512	19.8	3.6	17.9	3.3	20.6	3.2	53.9	3.4	98.9	2.2
1024	12.1	6.0	10.9	6.0	13.2	5.0	32.5	5.6	82.8	2.6

TABLE 2

Time (CPU *seconds*) *and iterations using* LS1 *preconditioning for Problem* 2 *on a* 256 × 256 *grid with* 256 *processors. The* GMRES *method uses a Krylov subspace size of* 128 *before restarting.*

γ\ Solver	CGS		Bi-CGSTAB		QMRCGS		GMRES	
	Time	Its.	Time	Its.	Time	Its.	Time	Its.
0.01	7.0	342	4.7	207	7.0	291	14.8	291
0.1	6.8	331	5.0	221	7.7	322	16.3	325
1.0	16.2	792	10.2	445	17.9	754	32.2	620
10.0	18.3	897	18.8	832	22.0	931	62.5	1216
100.0	17.6	861	30.5	1258	16.7	707	68.4	1327

TABLE 3

Time (CPU *seconds*) *and iterations using* LS9 *preconditioning for Problem* 3 *on a* 256 × 256 *grid with* 1024 *processors. The* GMRES *method uses a Krylov subspace size of* 175 *before restarting.*

Ra \ Solver	CGS		Bi-CGSTAB		QMRCGS		GMRES	
	Time	Its.	Time	Its.	Time	Its.	Time	Its.
10^3	90.8	872	63.8	590	75.8	697	308.1	3254
10^4	215.8	2094	179.4	1692	176.9	1647	619.9	6219
5×10^4	658.1	6436	681.1	6496	571.0	5374	1823.2	18271

TABLE 4
Time (CPU *seconds*) *and iterations using* LS3 *preconditioning for Problem* 4 *on a* 256 × 256 *grid with* 256 *processors. The* GMRES *method uses a Krylov subspace size of* 175 *before restarting.*

β\ Solver	CGS		Bi-CGSTAB		QMRCGS		GMRES	
	Time	Its.	Time	Its.	Time	Its.	Time	Its.
100.0	7.5	216	13.3	363	7.9	209	18.4	268
200.0	10.8	313	16.9	461	9.8	259	21.7	305
300.0	22.5	654	18.4	499	22.3	591	38.7	516
400.0	22.9	665	26.8	731	22.5	597	50.6	682
500.0	38.7	1124	45.1	1237	25.1	666	> 600	> 9000

In terms of overall solver performance GMRES is usually the slowest. In fact, it is often two or three times slower than the other methods. Additionally, large Krylov subspace sizes are sometimes needed to obtain any convergence. For example, in Table 3 the solution times are given for the nonlinear convection problem with increasing Rayleigh numbers. Increasing the Rayleigh number effectively makes the problem more nonlinear and more convective in nature. Even for the smaller Rayleigh numbers, a Krylov subspace size greater than 100 is needed to obtain convergence in a reasonable number of iterations. Furthermore, for a Rayleigh number of 5×10^4, convergence is still not obtained using a Krylov subspace size of 175. That is, the GMRES method is not only slower, but it often does not converge when the subspace size is not large enough. We should note, however, that the time differential between GMRES and the other solvers drops substantially (see Table 10) as more effective preconditioners are used.

For many problems we have observed that the Bi-CGSTAB method is slightly faster than both the CGS and the QMRCGS schemes. However, there are situations where Bi-CGSTAB performs noticeably worse. For example, in Table 4, we see that the Bi-CGSTAB method deteriorates as the magnitude of β is increased. This increase in β effectively makes more of the eigenvalues associated with the discrete operator negative. The net result is that all the methods degrade as β increases. However, the degradation is most pronounced with the Bi-CGSTAB scheme. Similar behavior is observed in Table 2 where Bi-CGSTAB is almost a factor of two slower than QMRCGS for $\gamma = 100.0$. In this case, the large value of γ produces an unstable discretization. While this instability does not greatly affect CGS and QMRCGS, it causes a large deterioration in the Bi-CGSTAB performance. Though unstable discretizations should generally be avoided, this example illustrates the greater sensitivity of the Bi-CGSTAB method to the specific nature of the equations.

Overall, there is no clear winner between the iterative solvers. For the most part, the GMRES method is the slowest. Typically, Bi-CGSTAB is slightly faster than CGS and QMRCGS (though it can sometimes be a lot slower on some hard problems). Finally, the run times of CGS and QMRCGS are usually quite close and in the worst case are only slightly longer than the fastest overall run time.

4.2. Preconditioning. The overall performance of a variety of preconditioners is depicted in Tables 5–8 using the Bi-CGSTAB method.[6] In the tables, the following notations are used for different preconditioners: Jac, Jacobi; RBGS, red-black Gauss–Seidel; mg[rbgs1], V cycle multigrid with one pre- and post-Gauss–Seidel sweep; mf, multilevel filtering; LILU,

[6]Only the Bi-CGSTAB method is used to simplify the presentation. The Bi-CGSTAB results are representative of preconditioner performance. However, it should be noted that the performance of the GMRES method is usually improved more substantially by preconditioning than the other schemes.

local ILU; and lineGS, alternating red-black line Gauss–Seidel. The time for the Jacobi preconditioned system essentially corresponds to the time for just using Bi-CGSTAB without preconditioning since the time required to diagonally scale the matrix is insignificant. While the results vary significantly, it is still possible to draw some general conclusions. The most important of these is that preconditioning *does* strongly influence the overall solution time. Overall, the global multilevel schemes frequently provide the "best" performance boost. In particular, the multigrid and multilevel filtering schemes often reduce solution times drastically when compared to local schemes (even though they require more communication and are somewhat more difficult to efficiently parallelize). This reduction is evident in Table 5 where the run time for Problrm 1 drops from 961.3 to 66.0 seconds when comparing Jacobi versus multigrid preconditioning on 64 processors.[7] Unfortunately, the parallel efficiency associated with multigrid preconditioning can be rather low on problems with small grids. For example, the time per iteration using multigrid preconditioning in Table 7 is quite high relative to that for Jacobi. This is essentially due to the inherent inefficiency associated with processing the coarse grids. Additionally, multilevel methods attain their speed by making assumptions about the underlying structure and behavior of the differential operator (e.g., that the PDE operator is elliptic). Even for elliptic problems with irregular behavior, the performance can drop off significantly unless more "specialized" techniques can be incorporated (e.g., near boundaries, interfaces, boundary layers, singularities, etc.). Therefore, it is not surprising that the multiple level methods do not perform well on problems such as Problem 2 (Table 6) for large γ and Problem 4 for large β (Table 8). Further line relaxation techniques are often used to produce more robust multigrid algorithms. Unfortunately, implementation of an efficient parallel tridiagonal solver (see §3) is quite difficult. Thus, the resulting multigrid method with line relaxation is slow (see Table 6, $\gamma \leq 1.0$) as the time per iteration is large. We are currently looking into some promising new multigrid algorithms which make use of semicoarsening techniques (see [16] and [17]) to produce a more efficient parallel robust multigrid method.

TABLE 5
Time (CPU *seconds*) *for* Bi-CGSTAB *on a* 1024 × 1024 *grid for Problem* 1.

Precond\procs	64	256	1024	1024*
Jac	961.3	325.2	73.7	114.3
RBGS	1324.4	346.0	108.0	129.2
LS1	984.0	279.6	61.3	97.9
LS3	735.9	185.0	53.9	89.4
LS9	774.7	188.4	51.5	89.0
NS1	1011.9	252.5	61.7	94.7
LILU	–	355.2	101.2	116.7
mg[rbgs1]	66.0	18.5	6.1	6.5
mf	99.4	27.9	8.9	9.5
– Not enough memory * Without assembly coded matrix-vector multiply				

Although local preconditioners are sometimes much slower than the multilevel schemes, their performance is much more uniform over a wider range of problems; they are easier to implement for a broad class of problems; and they can still reduce the overall run time by a substantial amount. For example, in Tables 6–8, it is apparent that a number of the local schemes perform well in comparison with the standard multilevel methods on these difficult problems. In addition, from Table 5 it is clear that as the number of processors increases, the

[7] There is some debate as to whether multigrid is best used alone or as a preconditioner. In this paper, we ignore this debate and simply illustrate the multigrid preconditioning results.

TABLE 6
CPU *time (iterations) for* Bi-CGSTAB *for Problem* 2 *on a* 256 × 256 *grid with* 256 *processors.*

Precond\γ	0.01	1.0	10.0	100.0
Jac	6.8(443)	14.0(895)	22.1(1468)	zero inner*
RBGS	15.5(445)	11.9(351)	23.1(689)	38.2(1110)
LS0	5.9(395)	12.5(815)	21.8(1460)	91.7(5508)
LS1	4.7(207)	10.2(445)	18.8(832)	30.5(1258)
LS9	5.3(67)	9.4(119)	18.2(232)	zero inner
NS1	4.8(194)	10.5(425)	18.5(762)	30.3(1171)
LILU	9.8(180)	14.9(277)	17.4(323)	12.2(224)
LineGS	50.8(70)	90.0(124)	97.9(135)	93.6(129)
mg[rbgs1]	7.7(29)	10.8(41)	35.8(136)	failed
mg[LineGS]	11.1(4)	19.4(7)	failed	failed
mf	7.1(49)	14.8(104)	32.3(226)	failed

*Indicates that the method breaks down with a zero inner product resulting in a divide by zero. Freund and Nachtigal [9] have proposed look-ahead techniques that overcome this problem within the QMR algorithm.

TABLE 7
CPU *time (iterations) for* Bi-CGSTAB *for Problem* 3 *on a* 128 × 128 *grid with* 256 *processors.*

Precond\ Ra	10^3	10^4	5×10^4	10^5
Jac	37.6(1736)	82.8(4101)	128.7(6540)	318.9(16000)
RBGS	38.4(879)	73.3(1705)	140.9(3315)	283.9(6722)
LS9	41.7(384)	112.2(1059)	failed	failed
NS1	35.8(1160)	105.8(3570)	413.3(14193)	732.0(25386)
LILU	77.9(928)	98.4(1492)	182.5(2813)	234.5(3655)
mg[rbgs1]	24.6(62)	40.1(118)	failed	failed

TABLE 8
Time (iterations) for Bi-CGSTAB *for Problem* 4 *on a* 256 × 256 *grid with* 256 *processors.*

Precond\β	100.0	200.0	300.0	400.0	500.0
Jacobi	22.2(1511)	48.9(3329)	64.0(4340)	112.1(7617)	failed
RBGS	24.4(732)	39.4(1181)	58.5(1755)	68.3(2043)	77.9(2339)
LS0	17.9(1214)	58.4(3962)	49.3(3326)	failed	failed
LS1	14.1(627)	28.0(1243)	36.3(1616)	63.5(2796)	91.1(4049)
LS9	11.6(147)	11.3(143)	25.3(322)	26.1(333)	zero inner
NS1	11.9(4911)	25.6(1059)	36.2(1496)	48.9(2023)	57.2(2371)
LILU	28.1(531)	39.4(747)	76.5(1446)	78.5(1492)	151.5(2894)
LineGS	269.0(371)	313.2(432)	412.5(569)	514.1(709)	933.7(1288)
mg[rbgs1]	16.4(62)	104.4(397)	269.4(1025)	111.5(424)	89.5(340)
mg[LineGS]	27.7(10)	102.1(37)	140.8(51)	failed	failed

CPU time reduction relative to the local methods decreases for a fixed size problem. This decrease in relative performance is due to the higher communication costs associated with the global exchange of information at each iteration for the global preconditioners. In this sense the local methods can be seen to scale more effectively to large number of processors for a fixed size computational problem.

Among local preconditioners, least-squares polynomials perform the best over a wide range of problems despite the simplified criteria used to determine the polynomial coefficients (see §3.1). This can be seen in Tables 5–8 as well as in Fig. 6 where the time and number of

iterations for Problem 1 are given as a function of polynomial degree. As Fig. 6 illustrates, the use of high-order polynomials greatly reduces the total iterations (and hence the number of global inner products). This reduction, however, comes at the cost of increasing the number of local matrix-vector operations. Thus, the overall performance is a function of the cost differential between local matrix-vector computations and global inner products. On massively parallel machines such as the nCUBE2, the matrix-vector product is relatively inexpensive and hence high-order polynomial preconditioners are often effective. In the parallel computing environment it is definitely an advantage to have the ability to easily alter the balance of local and global data flow to suit parallel machines with differing computation/communication characteristics. Additionally, we remark that the polynomial preconditioners generally outperform Gauss–Seidel preconditioning. This is partially a function of code optimization and partially a function of communication requirements. In particular, an assembly coded matrix-vector multiply routine is used for the polynomial methods, but only a standard C code is used for the Gauss–Seidel routines. In the optimization of the sparse matrix-vector routine, the best standard C code performed at about .8 Mflops and the assembly code ran at 1.3 Mflops per node. There are essentially three reasons that we believe it is justified to use of the assembly coded matrix-vector multiply routine in our results. The first and possibly the most important reason is that all the preconditioned solvers benefit. This is evident from Table 5 where we present results for the nonassembly coded results with the optimized assembly code. It is apparent that the use of assembly code has not significantly changed the relative performance of the various preconditioners. In addition, the matrix-vector kernel is a very simple routine that can be easily optimized. And finally, highly optimized versions of standard format sparse matrix-vector multiplies are now being developed for specific machines by vendors [12]. In terms of communication overhead, polynomial methods do not require any type of multicoloring and, therefore, incur fewer costly message startups per iteration than does Gauss–Seidel preconditioning.

Finally, we discuss the local ILU method that lies somewhere between a local and a global preconditioner. The classic ILU scheme is among the most popular preconditioners on serial machines. This scheme is relatively easy to implement and can be designed so that the amount of work per iteration (which is closely tied to the amount of fill-in) can be controlled by a few simple parameters. In this way, it is possible to adjust the ILU scheme according to the difficulty of the problem being solved. It is often the case that on some difficult problems, only the ILU preconditioner will yield satisfactory results. Unfortunately, the classical ILU algorithm (using natural ordering) does not parallelize well, and so we use the local ILU method discussed in §3. Since this LILU scheme ignores coupling between grid points that reside on different processors, one might expect that its convergence rate would be slower than standard ILU. However, in many problems this convergence degradation is quite small. In Table 9 we illustrate this effect using different numbers of subdomains (processors). From the table it is clear that the convergence degradation of the LILU preconditioner on this problem is only slight and is more than offset by the high degree of parallelism. By comparison with the polynomial preconditioners (see Table 10), the resulting LILU speedups are respectable. While the overall run time is competitive with unoptimized polynomial preconditioners, its performance is somewhat mediocre compared to the highly optimized polynomial schemes. However, if one optimizes the LILU it should run quite competitively with the other methods. Unfortunately, this optimization is fairly difficult.

We conclude our evaluation by considering the performance of the nCUBE 2 using these solvers on large problems. In particular, we wish to determine the megaflops and *scaled* speedups that one can expect to obtain using these highly parallel solvers and preconditioners on parallel machines. Specifically, *scaled* speedup is defined as

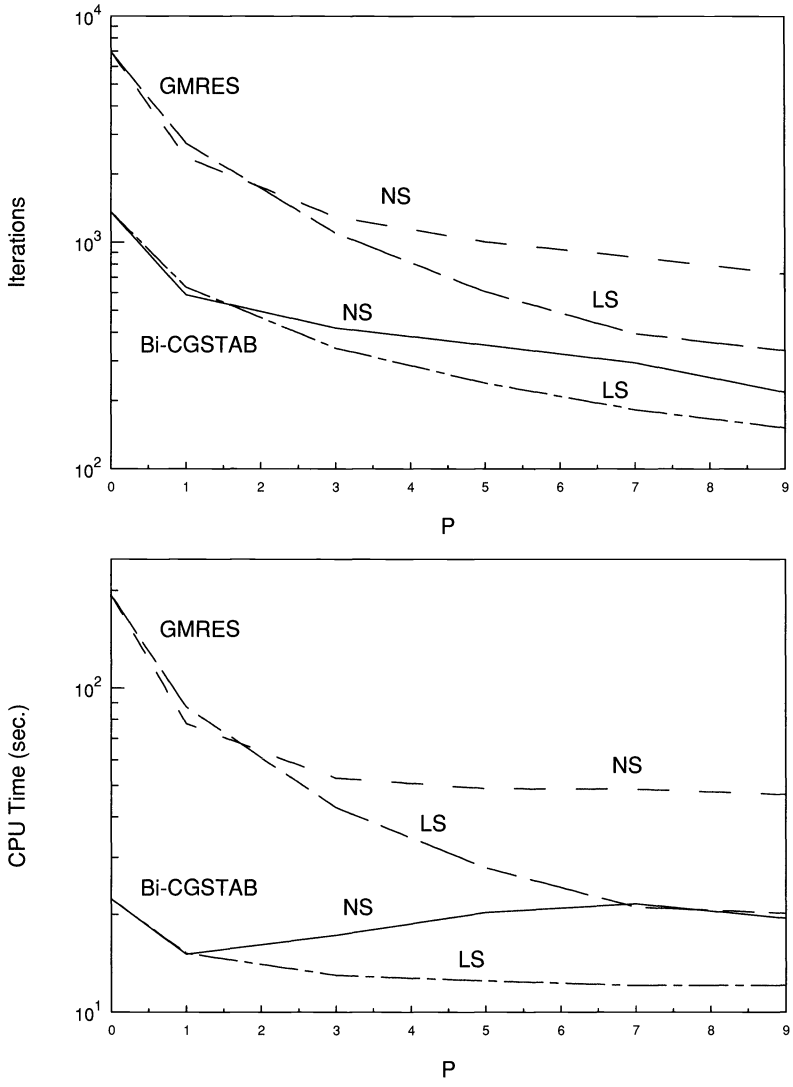


FIG. 6. *Convergence of* GMRES *and* Bi-CGSTAB *for Problem* 1 *on a* 1024 × 1024 *grid with* 1024 *processors varying the degree* (P) *of the polynomial preconditioner.*

TABLE 9
Performance of LILU *preconditioner on an* 80 × 80 *grid for Problem* 1.

Processors	1	2	4	8	16	32	64	128	256	512	1024
Iterations	50	52	55	57	58	59	67	67	73	84	91

$$S_p^* = \frac{pT_1}{T_p},$$

where T_p is the time required to complete one iteration of the problem with $p * n$ unknowns using p processors [11]. From Table 10 we see that extremely high scaled speedups are obtained for all of the solvers. The majority of the methods and preconditioners achieve speedups of

TABLE 10
CPU *time (iterations), scaled speedups, and Mflops for Problem* 1. *The* GMRES *method uses a Krylov subspace size of* 64 *before restarting.*

Algorithm\ Procs		256 (512 x 512 grid)			1024 (1024 x 1024 grid)		
		Time	S_p^*	Mflops	Time	S_p^*	Mflops
CGS:	JAC	41.5 (1265)	231	289	77.0 (2297)	903	1127
	RBGS	55.1 (599)	239	154	118.9 (1286)	949	612
	LS,3	34.5 (353)	245	279	68.4 (698)	979	1113
	LS,9	33.5 (152)	250	280	79.5 (361)	1000	1123
	LILU	64.9 (462)	239	125	127.8 (918)	963	505
	mg(rbgs1)	5.3 (8)	202	62	5.4 (8)	794	244
	mf	7.5 (20)	216	57	8.4 (22)	847	224
CGSTAB:	JAC	35.6 (983)	230	304	73.6 (1978)	856	1189
	RBGS	45.0 (475)	237	160	108.0 (1125)	935	635
	LS,3	28.2 (280)	243	284	53.9 (531)	970	1134
	LS,9	33.1 (148)	248	283	51.5 (230)	991	1132
	LILU	49.2 (339)	234	127	103.2 (720)	946	513
	mg(rbgs1)	6.5 (10)	204	66	6.1 (9)	782	253
	mf	8.3 (22)	192	61	8.9 (23)	750	238
QMRCGS:	JAC	46.9 (1165)	228	248	95.5 (2308)	889	963
	RBGS	58.8 (595)	244	119	138.0 (1383)	965	588
	LS,3	35.3 (335)	247	264	74.0 (698)	981	1048
	LS,9	32.8 (143)	253	272	63.4 (277)	1015	1090
	LILU	64.2 (431)	235	133	135.0 (915)	950	537
	mg(rbgs1)	4.4 (6)	205	57	4.5 (6)	800	223
	mf	6.4 (16)	218	55	6.9 (17)	860	216
GMRES:	JAC	599.6 (6993)	233	442	2194.5 (25273)	920	1743
	RBGS	232.4 (2013)	234	349	780.0 (6685)	928	1384
	LS,3	96.2 (814)	243	394	301.6 (2527)	961	1568
	LS,9	38.6 (215)	252	352	113.0 (619)	997	1414
	LILU	148.6 (1062)	251	308	468.0 (3340)	1002	1230
	LineGS	825.9 (751)	158	62	2601.7 (2237)	601	240
	mg(rbgs2)	4.3 (8)	174	61	5.4 (10)	710	250
	mf	5.4 (23)	225	95	6.1 (25)	890	391

over 900 out of a possible 1024. Even the majority of the multilevel preconditioners that make heavy use of coarse grids (and so are somewhat less efficient on parallel machines) achieve speedups over 700. In fact, only the multilevel methods that use the the line relaxation have poor speedups. Consequently, the majority of the solvers/preconditioners sustain computation rates of over 1 gigaflop and nearly 2 gigaflops for the GMRES method. This compares quite favorably with the vendors maximum computation rating of about 2.5 gigaflops for double precision calculations.

5. Conclusions. We have considered the performance of a variety of preconditioners and Krylov subspace solvers on massively parallel machines. These methods are among the fastest algorithms on serial and vector computers. An implementation of the Krylov solvers and preconditioners was developed on a 1024 processor nCUBE 2. Using this implementation in conjunction with a selection of PDE problems, a number of performance tests were conducted. Based on these experiments, it is apparent that the Krylov solvers are in fact highly parallel. Each method yields scaled speedups over 900 (when used with a highly parallel preconditioner). With respect to numerical convergence, we have found that preconditioning is much more important with these nonsymmetric methods than with the classical CG algo-

rithm. Overall, it appears that multilevel preconditioners offer the largest payoff in terms of convergence rate. In this context, we have shown that algorithmic issues are still important on large-scale parallel machines and that Mflops do not necessarily tell the entire story. Unfortunately, multilevel schemes can be hard to program and often their exceptional performance is somewhat problem dependent.

Among local schemes, polynomial methods (in particular high-order least-squares polynomials) significantly accelerate the convergence of the Krylov solvers enabling fast solution times. These methods combine ease of implementation with the flexibility to alter the balance between local and global data flow requirements. For the most part, all the preconditioners discussed in this paper are highly parallel with the exception of the classical ILU and to some extent the line relaxation. For the line relaxation, it may be possible to improve performance with a better parallel implementation of the block parallel cyclic reduction or by alternative algorithms that perform a similar function but make better use of parallel processors. For the ILU schemes, it is possible to use domain decomposition techniques to maintain parallelism. In this context we feel that some combination of domain decomposition with globally coupled interface equations and LILU should be explored for large-scale MIMD machines.

Acknowledgment. We would like to thank S. R. Wheat and the SUNMOS team for assistance with determining experimental floating-point operation rates for the LILU and lineGS preconditioners.

REFERENCES

[1] L. ADAMS AND J. ORTEGA, *A multi-color* SOR *method for parallel computers*, in Proc. Internat. Conf. Parallel Processing, 1982.

[2] J. BRAMBLE, J. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 1991, to appear.

[3] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[4] T. CHAN, C.-C. J. KUO, AND C. H. TONG, *Parallel elliptic preconditioners: Fourier analysis and performance on the connection machine*, Comput. Phys. Comm., 53 (1989), pp. 237–252.

[5] T. CHAN AND Y. SAAD, *Multigrid algorithms on the hypercube multiprocessor*, IEEE Trans. Comp., C-35(11) (1986), pp. 969–977.

[6] P. CONCUS, G. GOLUB, AND D. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Studies in Numerical Analysis, G. Golub, ed., Vol. 24, MAA Studies in Mathematics, 1984, pp. 178–198.

[7] P. F. DUBOIS, A. GREENBAUM, AND G. H. RODRIQUE, *Approximating the inverse of a matrix for use in iterative algorithms on vector processors*, Computing, 22 (1979), pp. 257–268.

[8] R. W. FREUND, *A Transpose-Free Quasi-Minimal Residual Algorithm for Non-Hermitian Linear Systems*, Tech. Report, RIACS, NASA-Ames Research Center, Moffett Field, CA, Sept. 1991.

[9] R. W. FREUND AND N. M. NACHTIGAL, QMR: *A Quasi-Minimum Resiudal Method for Non-Hermitian Linear Systems*, Tech. Report, RIACS, NASA-Ames Research Center, Moffett Field, CA, 1990.

[10] E. GALLOPOULOS AND Y. SAAD, *Some Fast Elliptic Solvers on Parallel Architectures and their Complexities*, Tech. Report, RIACS, NASA-Ames Research Center, Moffett Field, CA, Apr. 1989.

[11] J. GUSTAFSON, G. MONTRY, AND R. BENNER, *Development of parallel methods for a* 1024*-processor hypercube*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 609–638.

[12] M. HEROUX, P. VU, AND C. YANG, *A parallel preconditioned conjugate gradient package for solving sparse systems on a Cray* Y-MP, Appl. Numer. Math., 8 (1991), pp. 93–115.

[13] L. JOHNSSON, *Solving tridiagonal systems on ensemble architectures*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 354–392.

[14] C.-C. KUO, T. F. CHAN, AND C. H. TONG, *Mulilevel Filtering Elliptic Preconditioners*, Tech. Report CAM 89-23, Dept. of Mathematics, Univ. of California at Los Angeles, 1989.

[15] J. MEIJERINK AND H. 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.

[16] W. MULDER, *Multigrid, alignment, and Euler's equations*, in Proc. 4th Copper Mountain Conf. Multigrid Methods, S. McCormick, ed., Marcel Dekker, New York, 1989, pp. 348–364.

[17] N. NAIK AND J. ROSENDALE, *Robust parallel multigrid using multiple semi-coarse grids*, in Preliminary Proc. 5th Copper Mountain Conf. on Multigrid Methods, S. McCormick, ed., Marcel Dekker, New York, 1991.

[18] Y. SAAD, *Practical use of polynomial preconditionings for the conjugate gradient method*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 865–881.

[19] ———, *Least squares polynomials in the complex plane and their use for solving nonsymmetric linear systems*, SIAM J. Numer. Anal., 24 (1987), pp. 155–169.

[20] ———, SPARSKIT: *A Basic Tool Kit for Sparse Matrix Computations*, Tech. Report, RIACS, NASA-Ames Research Center, Moffett Field, CA, Apr. 1990.

[21] 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.

[22] J. N. SHADID, *Experimental and Computational Study of the Stability of Natural Convection Flow in an Inclined Enclosure*, Ph.D. thesis, Mechanical Engineering Dept., Univ. of Minnesota, Minneapolis, 1989.

[23] J. N. SHADID AND R. S. TUMINARO, *Iterative methods for nonsymmetric systems on* MIMD *machines*, in Proc. 5th SIAM Conf. Parallel Processing for Scientific Applications, J. Dongarra, P. Messina, D. C. Sorensen, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 123–129.

[24] ———, *Sparse iterative algorithm software for large-scale* MIMD *machines: An initial discussion and implementation*, Concurrency: Practice and Experience, 4 (1992), pp. 481–497.

[25] B. SMITH AND O. WIDLUND, *A domain decomposition algorithm using a hierarchical basis*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 1212–1220.

[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. TONG, T. CHAN, AND C. KUO, *A multilevel nodal basis domain-decomposed preconditioner*, in Preliminary Proc. 5th Copper Mountain Conf. on Multigrid Methods, S. McCormick, ed., Marcel Dekker, New York, 1991.

[28] H. VAN DER VORST, *Iterative solution methods for certain sparse linear systems with a nonsymmetric matrix arising from* PDE *problems*, J. Comput. Phys., 44 (1981), pp. 1–19.

[29] 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.

[30] Y. S. WONG AND H. JIANG, *Approximate polynomial preconditionings applied to biharmonic equations*, J. Supercomputing, 3 (1989), pp. 125–144.

[31] D. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 460–473, March 1994

014

MEMORY ASPECTS AND PERFORMANCE OF ITERATIVE SOLVERS*

CLAUDE POMMERELL† AND WOLFGANG FICHTNER‡

Abstract. In many scientific computing problems, the overall execution time is dominated by the time to solve very large linear systems. Quite often, the matrices are unsymmetric and ill conditioned, with an irregular sparsity structure reflecting the irregular refinement in the discretization grid. With increasing problem size and problem dimension, direct solvers cannot be used because of the huge memory requirements. The performance of preconditioned iterative solvers is largely dominated by memory-related aspects like size, bandwidth, and indirect addressing speed.

This article summarizes the experience of the authors on the relationship between memory aspects and performance in real applications in the domain of very large scale integration (VLSI) device simulation. The authors analyze storage requirements of direct and iterative solvers on a statistical data set, and demonstrate performance variations due to memory-related architectural features on a number of computers ranging from workstations to Cray, NEC, and Fujitsu supercomputers for typical and ill-conditioned linear systems, using different iterative methods and preconditioners.

The experiments are done using PILS, a package of iterative linear solvers. PILS implements a large number of iterative methods and preconditioners and allows them to be combined in a flexible way.

Key words. solution of large systems of linear equations, preconditioned iterative methods, software, supercomputer performance, semiconductor device simulation

AMS subject classifications. 65Y20, 65F10

1. Introduction.

The solution of large systems of linear equations is one of the main operations inside many large-scale scientific applications. As problem sizes increase, the linear solver often dominates the total execution time. Sparsity of the system matrix must be exploited to achieve acceptable efficiency. Irregular sparsity patterns and ill-conditioned linear systems induce further complexity on the choice of methods and implementations.

There are two basic approaches used to solve large sparse linear systems. *Direct methods* use (explicitly or implicitly) a factorization of the matrix, where the factors are easily invertible (e.g., triangular matrices). *Iterative methods* construct a sequence of approximations that converges to the solution and stop when a given tolerance is achieved. In §4.1, we analyze the storage requirements of both approaches on a larger set of practical cases. We analyze their dependence on the dimensionality of the problem and point out the infeasibility of direct methods on very large problems. In the rest of this paper, we focus only on iterative solvers.

The performance of scientific codes is usually measured in Mflops (millions of floating-point operations per second). Our benchmarks in §4.2 show the Mflops performance of our preferred preconditioned iterative solver over a range of machines from workstations to supercomputers. Using a set of problems of different sizes, we explain why performance does not necessarily increase with problem size.

The use of Mflops may be reasonable for applications where a given algorithm is known to be the "best" and "fastest." It is questionable for solvers targeted towards ill-conditioned and irregular sparse matrices, as there is no single algorithm that is the fastest for all problems (even not for all problems among a restricted class, such as the one described in §2). Different algorithms use a different number of operations to achieve the same result. Also, the same number of flops requires a different number and different kinds of memory accesses, which

*Received by the editors May 18, 1992; accepted for publication (in revised form) February 26, 1993.

†AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974 (pommy@research.att.com).

‡Integrated Systems Laboratory, Eidgenössische Technische Hochschule (ETH) Zürich, Gloriastrasse 35, CH-8092 Zürich, Switzerland (fw@iis.ee.ethz.ch).

better suit one architecture or another. We illustrate these variations of the performance by using different iterative methods in §4.3 and different preconditioners in §4.4.

Periodically, we stumble over certain very ill-conditioned linear systems that standard preconditioned iterative solvers cannot solve. We have developed several parametrized sets of preconditioners to solve these difficult problems. In §4.5, we illustrate how the parameters of these robust solvers affect their speed and their storage requirements.

The experiments in §4 are preceded by an overview of the methods and our context. Section 2 summarizes the need for linear solvers in our application, semiconductor device simulation. Section 3 lists those features of the methods, preconditioners, and data structures that are essential for this study. It also introduces PILS, our software package used both as a test bed for studies like this and as a solver library for production runs.

2. Context. The present study is part of a large project to analyze and construct new VLSI semiconductor devices (diodes, transistors, memory cells, thyristors, sensors). Numerical semiconductor device simulation predicts the behavior of such structures by solving a set of three transient partial differential equations (PDEs), the so-called drift-diffusion equations [34], over a one-, two- or three-dimensional domain [4]. This system consists of Poisson's equation and two continuity equations for the two carriers in semiconductors, electrons and holes. The drift-diffusion equations relate the values of the electrostatic potential and the two carrier concentrations.

The discretization of these PDEs leads to a system of nonlinear equations that is then solved by damped Newton methods [3], [7] or block variants thereof. Each Newton iteration requires the solution of a system of linear equations. Several tens of linear systems typically must be solved in stationary simulations and several hundreds or even thousands in transient simulations.

Modern device simulators use irregularly refined finite-element grids, where the grid density in the simulated domain may vary over several orders of magnitude. The number of grid points varies between 1,000 and 20,000 for two-dimensional (2-D) simulations and between 15,000 and 100,000 for three-dimensional (3-D) simulations. The number of unknowns in the corresponding linear systems is up to three times as large.

The system of PDEs is known to be extremely nonlinear due to the coupling of the physical variables in these equations. The linear systems are unsymmetric and tend to be very ill conditioned. Because of their huge resource requirements, both in storage and computation time, device simulators must run on high-performance architectures. The linear solver time is dominating; it takes more than 95% of the total execution time [17]. The irregularity of the discretization grid leads to an irregular sparsity structure for the matrices. This makes the task of vectorizing and parallelizing the solver algorithms considerably more difficult than for regular (e.g., finite difference) discretizations.

For further details about device simulation, please consult [18].

3. PILS. PILS is a software package of iterative linear solvers designed to meet the requirements for solving large linear systems as they occur in applications like device simulation. PILS implements a large number of iterative methods, preconditioners, and other variants for iterative solvers, and runs on a large range of platforms from workstations to supercomputers. It concentrates on methods and preconditioners suited for very ill-conditioned unsymmetric systems and on the efficient exploitation of high-performance architectures.

In this section, we will only discuss those aspects of PILS and its algorithms that are relevant for the present study. More about PILS, its requirements, its features, its structure and implementation, and its applications can be found in [27]. The methods, preconditioners, and data structures are covered in detail in [25].

3.1. Iterative methods. The set of iterative methods implemented in PILS currently includes conjugate gradients (CG) [19] (for symmetric positive-definite systems only), conjugate gradients applied to the normal equations (CGNR) [19], (restarted) generalized conjugate residuals (GCR) [11], (truncated) Orthomin [35], (restarted) generalized minimal residuals (GMRES) [30], biconjugate gradients (BiCG) [12], conjugate gradients squared (CGS) [31], Bi-CGSTAB [32], and BiCGStab2 [16].

CG applied to symmetric and positive definite systems converges monotonically, has a constant and low storage overhead, uses the minimum number of matrix-vector products to reach a residual norm within a tolerance, and cannot break down in divisions by zero. The other methods can be used for unsymmetric systems as well, but at the price of losing one or more of these four properties. CGNR is usually much slower than the other methods. GCR(∞), Orthomin(∞), and GMRES(∞) need the minimum number of matrix-vector products to minimize the residual, but must store another vector at every iteration. In practice, the methods in this family are usually restarted (GCR(ℓ), GMRES(ℓ)) or truncated (Orthomin(ℓ)). Unfortunately, the limited storage versions usually require many more iterations. Breakdowns do occur sometimes in GCR and Orthomin, but not in GMRES (which is otherwise algebraically equivalent to GCR). BiCG has constant and low storage requirements, but has an irregular convergence behavior. If BiCG converges, CGS usually converges much faster, but its convergence behavior is also much more erratic. Bi-CGSTAB and BiCGStab2 are usually about as fast as CGS, but have a smoother (yet not monotonic) convergence behavior. The methods in the BiCG family can break down, but breakdowns are quite infrequent and can usually be fixed by restarting the method.

Table 1 lists the storage requirements and the types of operations involved.

TABLE 1

Storage overhead and (average) operation count per iteration for the different iterative methods. n is the problem size, k is the number of iterations needed for convergence, ℓ is the truncation or restarting parameter. Minor terms are ignored (assuming that $n \gg k^2 \gg \ell^2$).

Method	Storage overhead (extra values to store)	Operations per iteration			
		Matrix-vector products	Transposed matrix-vector products	Vector dot products	Linear operations on vectors
CG	$2n$	1	–	2	6
CGNR	$2n$	1	1	2	6
Orthomin(∞) / GCR(∞)	$(2k+1)\,n$	1	–	$\frac{1}{2}k+2.5$	$k+4.5$
GMRES(∞)	kn	1	–	$\frac{1}{2}k+1.5$	$k+3.5$
Orthomin(ℓ)	$(2\ell+1)\,n$	1	–	$\ell+2$	$2\ell+4$
GCR(ℓ)	$(2\ell+1)\,n$	1	–	$\frac{1}{2}\ell+2$	$\ell+4$
GMRES(ℓ)	ℓn	1	–	$\frac{1}{2}\ell+1$	$\ell+3$
BiCG	$4n$	1	1	2	10
CGS	$6n$	2	–	2	12
Bi-CGSTAB	$4n$	2	–	4	12
BiCGStab2	$7n$	2	–	4.5	18.5

3.2. Preconditioners. Preconditioning significantly improves the convergence speed of all the methods above. A *preconditioned* iterative method solves the linear system $[Q_1^{-1}AQ_2^{-1}][Q_2x] = [Q_1^{-1}b]$. The preconditioner $Q = Q_1Q_2$ is chosen such that iterative methods converge in substantially fewer iterations than on the original system $Ax = b$. Generally, a preconditioner is an approximation to the matrix that is more or less inexpensive to construct and to invert.

Most of the preconditioners in PILS can be viewed as incomplete LDU-factorizations. A complete factorization of the matrix A into the product of the unit lower triangular matrix L, the diagonal matrix D, and the unit upper triangular matrix U can be computed by

$$\forall i > j : l_{ij} = \frac{1}{d_{jj}} \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik} d_{kk} u_{kj} \right), \tag{1}$$

$$\forall i < j : u_{ij} = \frac{1}{d_{ii}} \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} u_{kj} \right), \tag{2}$$

$$\forall i : d_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik} d_{kk} u_{ki}. \tag{3}$$

Incomplete factorizations are obtained by simplifying the factorization rules (1)–(3). Jacobi (or diagonal) preconditioning is the most radical simplification, approximating A by its diagonal only. Symmetric successive overrelaxation (SSOR) preconditioning ignores the summations in (1)–(3). D-ILU preconditioning ignores the summations in (1) and (2), but uses the complete summation for the diagonal in (3). ILU preconditioning [21], [11] uses all three formulae (1)–(3), but keeps the same sparsity structure as A; a zero in A imposes a zero in the same position of the ILU factors.[1] Table 2 summarizes the resource cost of these four preconditioners. The low cost in applying SSOR or D-ILU preconditioners results from a clever combination of the operations for a *preconditioned matrix-vector product*, known as the *Eisenstat trick* [10].

TABLE 2

Storage overhead and number of extra flops for the different no-fill incomplete factorization preconditioners. n is the problem size, m is the number of nonzeros in the original matrix. An unpreconditioned matrix-vector multiplication takes m multiplications and $(m - n)$ additions for a total of $(2m - n)$ flops.

Preconditioner	Storage overhead (extra values to store)	Computational overhead (flops)	
		Construction $Q :\approx A$ (once per linear system)	Application $w := Q^{-1}v$ (once or twice per iteration)
Jacobi	0	0	n
SSOR	0	0	$5n$
D-ILU	n	$\frac{3}{2}(m - n)$	$5n$
ILU	m	$O(m^2/n)$	$2m - n$

While the incomplete factorizations discussed above conserve the sparsity structure of the original matrix, other variants in PILS allow some limited fill of nonzero entries in the factors. ILU(k) uses a positional dropping criterion (based on the elimination tree, as suggested in [5]) to select fill entries; higher values of the integer parameter k allow for more fill. Another approximate factorization preconditioner in PILS uses numerical dropping with a drop tolerance τ and is referred to as ND(τ) [2], [13], [25], [27], [28], [37]; nonzero entries whose absolute value is smaller than the tolerance times the row or column maximum are dropped from the factorization. The overhead of these preconditioners depends, in a monotonic but nontrivial way, on their parameter.

[1] D-ILU is equivalent to ILU on sparsity structures resulting from tensor-product grids (or any other graph containing no triangles). Since many authors use only finite difference discretizations on such regular grids for their experiments, these two variants of no-fill incomplete LDU-factorizations are often confounded.

Nested iterative solvers form another class of preconditioners in PILS. Every iterative solver in PILS can be used to precondition another iterative method. This nesting can be taken to any level, and the innermost iterative method can again be preconditioned by an incomplete factorization. Nested iterative solvers using variants of GCR as the outer method and GMRES as the inner method have been introduced and analyzed under the name GMRESR by van der Vorst and Vuik [33]. Other variants for the inner method were investigated in [36].

We use incomplete factorizations mostly as a split preconditioner ($Q_1 = L$, $Q_2 = DU$) and nested solvers as a right preconditioner ($Q_1 = I$, $Q_2 = Q$).

3.3. Sparse matrix data structures. PILS applies a multicolor reordering (using a greedy coloring heuristic [1], [26]) to vectorize and parallelize the no-fill incomplete factorization preconditioners. A jagged diagonal sparse matrix structure [29] helps to vectorize sparse matrix-vector multiplication. To make transposed matrix-vector products (as needed in CGNR and BiCG) as efficient as regular matrix-vector products, the set of edges corresponding to each jagged diagonal forms a *matching* [23] in the graph representing the sparsity structure of the matrix. The data structure in PILS allows efficient and vectorizable preconditioned matrix-vector multiplication with the preconditioned matrix itself and its transpose. It stores the lower and the upper triangle of the matrix by color blocks, each block using jagged diagonals that form a *partitioning by matchings* of the corresponding edges of the associated graph.

Each iteration of the inner loops processing this data structure consists of one multiplication and one addition, together with five memory accesses. One of these memory accesses is indexed and requires gather/scatter operations on vector machines.

PILS uses different, specialized data structures for factorizations with parameter-controlled fill. Because of their higher complexity, operations on these data structures run essentially in sequential mode.

3.4. Application. PILS is fully integrated into several device simulators [6], [17], [18], [20], [22], and other applications, and has been used in many different problems over a period of more than two years. The options for PILS, usually controlled from the client application, can be modified externally even while the application is running. In case a given combination of a method and a preconditioner cannot solve a system within a given number of iterations, PILS can switch automatically to another (usually more resource intensive, but more stable) combination.

All the experiments presented in this paper were conducted using PILS and its monitoring options. The linear systems were extracted from real-life device simulations. They were selected to reflect the *typical* characteristics of linear systems in device simulation, including some very ill-conditioned cases.

4. Experiments. In §4.1, we analyze the effect that problem size and grid dimension have on the storage requirements of direct and iterative linear solvers. We then fix the solution approach (iterative solvers) and list, in §4.2, the Mflops rates of one particular preconditioned iterative method on a set of machines and on a representative set of linear systems. In §4.3, we select one typical linear system and analyze the speed and performance of several iterative methods applied to this problem (using the same preconditioner). We fix the iterative method in §4.4 and apply it with different preconditioners to that same typical system. In §4.5, we examine convergence speed and storage requirements of different preconditioners on a difficult, ill-conditioned linear system.

While §4.1 presents a statistical analysis on an impartially collected data set, the examples in §§4.2–4.5 were selected by hand as typical cases. Among the many thousands of linear systems we solved over the last years, we would be able to select examples for almost any

reasonable or unreasonable claim, such as the superiority of any given method over any other given method, or the particular quality of a machine. The reader must trust that our selection of examples truly reflects the *typical* behavior over the various applications in device simulation. An impartial statistical analysis on a significant data set would be too expensive, and large variances on the results would overshadow any of the observations that we point out in §§4.2–4.5.

Until now, our experience has been that the fastest linear solver for most problems in semiconductor device simulation is the Bi-CGSTAB method [32], split preconditioned with D-ILU [18], [27]. This method and preconditioner will therefore be used in those experiments where other things are varied.

4.1. Varying the approach. The alternatives to iterative solvers are sparse direct solvers. Direct solvers are often preferred over their iterative counterparts because of their reliability and predictability; they are numerically stable (at least if numerical pivoting is used) [8], and they take constant time and storage for a given sparsity structure (at least if numerical pivoting is not used). Solving for more than one right-hand side costs very little.

Most direct solvers are based on variants of Gaussian elimination. They construct a lower triangular matrix L and an upper triangular matrix U such that $LU = A$. The triangular factors L and U are also sparse, but usually more dense than the original matrix. Nonzeros in the factors appearing at zero positions of the original matrix are called *fill*. Reordering of the equations can reduce the amount of fill. The problem of computing the ordering that minimizes the fill is NP-complete. Heuristics like reverse Cuthill–McKee, nested dissection, or minimum degree and variants are used instead [14], [15]. Numerical stability, storage, efficiency, and parallelism are usually traded for one another [9].

The amount of fill increases superlinearly with the problem size and with the dimensionality of the problem. The fill for a matrix arising from a 3-D discretization is higher than for a matrix from a 2-D discretization with the same number of grid points. It is the combination of these two growth factors that makes the storage requirements of sparse direct solvers an issue when passing from 2-D to 3-D models.

Figure 1 compares the memory requirements for direct and iterative linear solvers. The fill was determined without actually performing the factorization on all the examples (no machine available to us has enough memory for the large 3-D cases), but an algorithm with minimal storage [5] was assumed, using nested dissection ordering [14] to reduce the fill. The analysis involved 224 nontrivial examples from device simulation,[2] using the matrices from the coupled Newton solution, and assuming 8-byte floating-point and 4-byte integer numbers.

In the double logarithmic plot of Fig. 1, the sample points for each of the assorted categories lie close to a line. We can thus state an empirical law of fill given by

$$s = \mu n^{\nu}, \tag{4}$$

where s is the amount of storage for a given number of unknowns n. Table 3 gives the values obtained from our empirical study.

The largest 2-D discretizations for semiconductor devices rarely exceed 10k grid points, and thus 30k unknowns for coupled linear systems (cf. §2). Direct solvers for such 2-D problems fit into 100 Mbytes of memory, which are usually available on departmental computers or in medium-size batch queues of supercomputers. On the other hand, iterative solvers for such 2-D problems fit into some 8 Mbytes available on any workstation. Direct solvers for

[2]Our data set used 172 2-D grids and 52 3-D grids for different kinds of semiconductor devices. In fact, these were all of the distinct nontrivial grids that were on disk at the Integrated Systems Laboratory of ETH on October 9, 1991. PILS was running inside the multidimensional device simulator SIMUL [20].

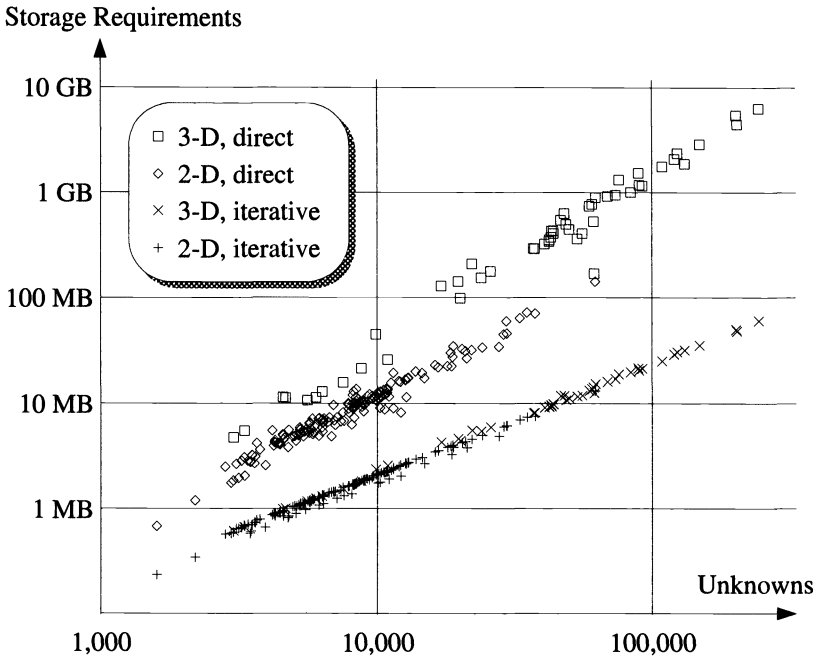


FIG. 1. *Sampled storage requirements for direct and iterative solvers.*

TABLE 3
Empirical values for the storage requirement formula (4).

Grid dimension	Type of solver	Factor μ [bytes/unknown]	Exponent ν
2-D	iterative	167	1.02
2-D	direct	63	1.32
3-D	iterative	156	1.04
3-D	direct	9	1.64

average size 3-D simulations can run only on very large computer memories, and no machine available today could accommodate truly large 3-D problems with several hundreds of thousands or even millions of grid points. Iterative solver storage requirements, even for 3-D problems, still increase linearly with problem size. At least for such large 3-D problems, there is no choice: direct methods are infeasible because of their storage requirements, and iterative solvers must be used.

The number of operations required to perform the factorization grows even faster than the storage requirements. Timing comparisons between the two choices show that for grid sizes above a certain crossover point, iterative methods are more efficient [18]. Such a comparison will always be biased, because it involves many parameters from the problem, the accuracy requirements, the implementation, and the platform, but the critical grid size is certainly much lower for 3-D than for 2-D.

4.2. Varying the machine. Table 4 reports the Mflops rates for one algorithm (split D-ILU preconditioned Bi-CGSTAB) when applied to the same set of problems on different machines. The differences in the performance clearly reflect the classes that these machines belong to: previous generation microprocessors, today's microprocessors, minisupercomputers, and supercomputers.

TABLE 4

Mflops rates for split D-ILU *preconditioned* Bi-CGSTAB *on different machines. Except for the Alliant, only one processor was used on each machine. Missing results indicate that the floating-point range, the virtual memory, or the maximum contiguously allocatable chunk of memory was too small on the machine at hand.*

Problem	lddh	uvdih	bp25e	dr15c	mct70c
Unknowns	2.7k	22k	26k	47k	210k
Nonzeros/row	6.9	6.5	9.1	21	20
Grid dimension	2-D	2-D	3-D	3-D	3-D
Sun-3/280	0.10	0.10			
Sequent S81 (i386 + fpa)	0.20	0.18	0.18	0.16	
DEC system 5500	1.98	1.46	1.33		
Sun SparcStation 1+	1.05	0.93	0.92		
Sun SparcStation 2	1.87	1.48	1.47	1.28	
Sun-4/490	1.85	1.61	1.58	1.44	1.27
Sun Sparc 10	5.05	3.42	3.22	2.90	1.98
IBM RS6000	5.22	3.46	3.57	2.70	
SGI Mips R3000	3.52	2.11	2.05	1.70	1.21
SGI Mips R4000	6.42	4.58	4.39	3.94	
DEC Alpha	9.86	6.61	6.60	5.92	4.17
HP 9000	12.13	7.38	7.30	6.11	
Alliant FX/80 (6 procs.)	5.09	3.65	3.55	3.46	2.53
Convex C-220	7.79	8.00	7.36	6.76	6.85
Convex C-3820	19.25	20.43	19.08	14.42	14.54
Cray-2	37.28	38.29	31.73	26.61	28.73
Cray Y-MP M92	57.77	60.47	52.28	47.78	48.19
Cray Y-MP 4D	105.42	126.83	114.50	103.55	106.18
Cray C98	207.46	256.21	210.48	201.06	210.81
NEC SX-3/22	156.53	197.43	142.03	136.49	
Fujitsu VP2200	109.15	126.32		96.94	101.49

The problem size (i.e., the number of unknowns) increases from left to right in Table 4. The performance does not always improve with increasing problem size; rather, the opposite is more often the case. There are two reasons for this effect and both are related to memory access speed.

First of all, the average number of nonzeros per row differs in the five examples. The two problems on the left of Table 4 come from 2-D simulations, the other three come from 3-D simulations. Also, the first three problems involve one unknown per grid point, the two others involve three unknowns per point. More nonzeros per row lead to a higher percentage of indirect memory accesses (inside the sparse matrix operations), which are generally slower than the direct (and even sequential) accesses in the rest of the operations.

For problems with approximately the same number of nonzeros per row ("lddh" and "uvdih," "dr15c" and "mct70c"), a positive effect of problem size on performance can be observed for the six machines at the bottom of the table. On other machines, the performance still decreases, and this is because the cache hit ratio decreases.

Note also the difference in performance between the two variants of the Cray Y-MP architecture. The two machines have exactly the same processor and clock frequency, and differ only in their memory performance: the Y-MP 4D has 128 banks of fast 17ns ECL memory, while the more economical Y-MP M92 uses 64 banks of slower 60ns DRAM.

When iterative method and preconditioner are fixed, the Mflops rate does give an appreciation of the relative speed of different machines. However, problem size and matrix density have a significant effect on performance. This effect is not explained by the sole computation speed (or peak performance), but by various architectural features like memory bandwidth and cache size.

4.3. Varying the iterative method. Table 5 shows the performance of different iterative methods when used with the same preconditioner (split D-ILU), executed on the same machine (one processor of a Cray Y-MP 4D), and applied to the same problem (a typical, moderately ill-conditioned system with 46k unknowns taken from the 3-D simulation of a trench DRAM cell inside the simulator SECOND [17], [18]).

TABLE 5
Performance of different split D-ILU *preconditioned methods on one processor of a Cray* Y-MP 4D.

Method	Performance [Mflops]	Convergence [iterations]	Time [seconds]
CGNR	98	>400	–
Orthomin(∞) } GCR(∞) }	194	89	6.89
GCR(∞)	123	361	12.16
GMRES(∞)	188	89	5.15
Orthomin(10)	148	>400	–
GMRES(10)	116	388	12.24
BiCG	100	140	7.18
CGS	100	78	4.09
Bi-CGSTAB	103	65	3.43
BiCGStab2	110	65	3.53

The methods from the GMRES-family (including Orthomin, GCR, GMRES) deliver much higher Mflops rates than the biorthogonalization methods (BiCG, CGS, Bi-CGSTAB, BiCGStab2). This difference can be explained with the following reflection. Knowing that there are on average 21 nonzeros in each row of the particular matrix used for Table 5, and looking at the operation breakdown in Table 1, we see that more than half of the number of flops in the GMRES-like methods are spent in well-vectorized linear operations (vector additions, scaling, and inner products), and the rest are spent in operations involving indirect memory addressing (sparse matrix-vector products and preconditioning). The biorthogonalization methods, on the other hand, spend only around 20% of their flops in linear operations, and four times as many of their flops require gather/scatter memory access.

The important column in Table 5, however, is the last one, showing the time in seconds required to solve the system. Truncated or restarted variants of GMRES generally require more iterations than BCG variants on device simulation problems [18], [24]. Note that on a similar machine without hardware gather/scatter support, GMRES(∞) would certainly converge in less time than Bi-CGSTAB, provided that enough memory is available to store the 89 direction vectors.

4.4. Varying the preconditioner. Table 6 shows variations of the performance with different preconditioners for the same combination of an iterative method, a problem, and a machine.

At first glance, the goal of a preconditioner is to reduce the number of iterations that an iterative method needs to solve a linear system to a given accuracy. In practice, however, the goal is to minimize the time to find a solution. Constructing a preconditioner takes time, and applying this preconditioner in an iterative method slows down each single iteration. The choice of one particular preconditioner is a trade-off between this cost of individual operations and the effect of reducing the number of iterations.

Our experience over several years now has been that D-ILU preconditioning achieves the best trade-off for most cases. The linear system selected for Table 6 represents well this "typical" behavior.

TABLE 6

Performance of Bi-CGSTAB *with different preconditioners (in split position), on a Convex* C-220 *and on a Cray* Y-MP 4D *(one processor used). Note that for technical reasons, the Mflops rates are measured only in the iteration phase, and not in the preconditioner setup.*

Preconditioner	Performance [Mflops] C-220	Y-MP 4D	Number of iterations	Total time [seconds] C-220	Y-MP 4D
none	6.66	100	>400	–	–
Jacobi	6.70	102	153	114	7.45
SSOR	6.74	103	69	55	3.62
D-ILU	6.76	103	65	53	3.43
ILU	6.24	94	61	110	11.47
ILU(1)	2.84	15	63	277	53.96
ND(0.01)	2.78	15	29	164	39.35
Nested	6.90	100	4	85	5.87

TABLE 7

Performance of Bi-CGSTAB *split preconditioned with a numerical dropping factorization on different machines.*

Machine	Total solution time [seconds]	Time in factorization [percentage]
Sun-4/490	222	23%
Sun Sparc 10	108	23%
IBM RS/6000	402	16%
SGI Mips R3000	197	22%
SGI Mips R4000	91	26%
DEC Alpha	69	26%
HP 9000	55	21%
Convex C-220	163	45%
Convex C-3820	70	45%
Cray Y-MP M92	54	57%
Cray Y-MP 4D	39	59%
Fujitsu VP2200	31	48%

More effective preconditioners are usually more costly in both time and memory consumption and offer less room for vectorization and parallelization. Accordingly, the relative performance of different preconditioners varies from one machine to another. Since an approximate factorization with numerical dropping is sometimes needed to precondition very difficult linear systems, we have listed its performance on a selection of machines in Table 7. The factorization phase is almost exclusively scalar code with finely grained data-dependent control flow, and the iteration phase is partly vectorized.

Note that the picture is slightly distorted by the fact that, because of our previous experiences [18], we tuned the data structures in PILS to SSOR, D-ILU, and ILU preconditioners. Slightly higher performance would be possible with no preconditioning or with Jacobi preconditioning. Some fine-grained parallelism would be conceivable in the incomplete factorizations with fill, but has not been investigated yet.

4.5. Varying the storage consumption. Occasionally, linear systems arise for which most preconditioners do not achieve convergence in any iterative method. Within a device simulation run involving hundreds of linear solves, a single such ill-conditioned linear system is sufficient to preclude the entire simulation. We have devised robust preconditioners for this purpose. To prevent excessive loss of efficiency, our software uses fast preconditioners for all the other systems and switches automatically to a robust preconditioner when needed.

We will now analyze such a very ill-conditioned linear system. Standard preconditioners and methods can solve it only after a very high number of iterations or not at all. Our system

occurs in the solution of the hole continuity equation in the simulation of a UV diode on a strongly refined 2-D grid with more than 20k points.

Such a resilient case can be solved with an approximate factorization preconditioner using numerical dropping. The lower the drop tolerance τ, the more fill entries are allowed in the approximate factors and the fewer iterations are needed for convergence. On the other hand, more fill in the preconditioner increases the CPU time *per iteration*. Therefore, the plot of the total time to solve the system as a function of the storage requirements for ND(τ) (dotted curves in Figs. 2 and 3) is convex.

Similarly, certain nested iterative solvers can solve this problem in a reasonable amount of time. The higher the maximum number of inner iterations (the numbers inside the circles in Figs. 2 and 3) per outer iteration, the fewer outer iterations are needed. We used split D-ILU preconditioned Bi-CGSTAB as inner method and GCR(∞) as outer method for Figs. 2 and 3. GCR requires the storage of two more vectors for each iteration, so that the number of outer iterations dictates the storage requirements of the entire solver.

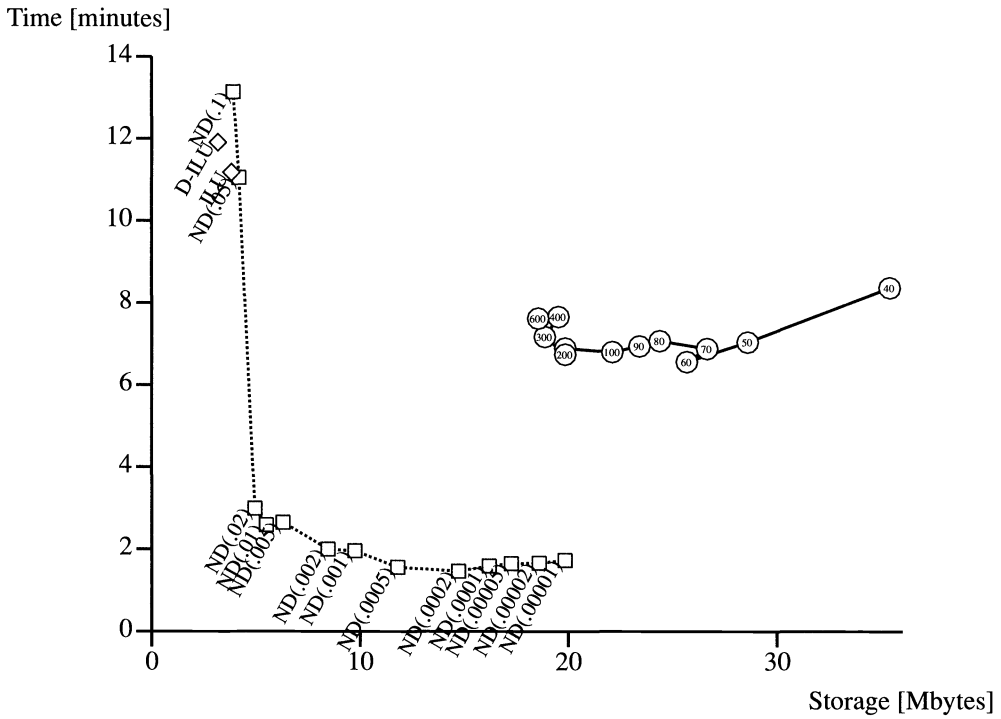


FIG. 2. *Storage size and solution time for an ill-conditioned linear system on a Convex* C-220 *(one processor). The square marks refer to different preconditioners used with* Bi-CGSTAB, *the dotted line following the marks of numerical dropping preconditioning with different drop tolerances. The solid line with the round marks refers to a nested iterative solver, with* GCR(∞) *as the outer method and split* D-ILU *preconditioned* Bi-CGSTAB *as the inner method.*

The experiments for Fig. 2 were done on a Convex C-220 and the experiments for Fig. 3 on a Cray Y-MP 4D (one processor). The set of experiments was exactly the same. The relative locations of the (dotted) curve for numerical dropping preconditioning and (solid) curve for nested iterative solvers are different for two reasons:

1. The ND(τ) preconditioner runs mainly in scalar mode. The performance ratio between vector and scalar code is much higher on the Cray than on the Convex, so that the ND(τ) preconditioned solvers are not as much faster than the others on the Cray.

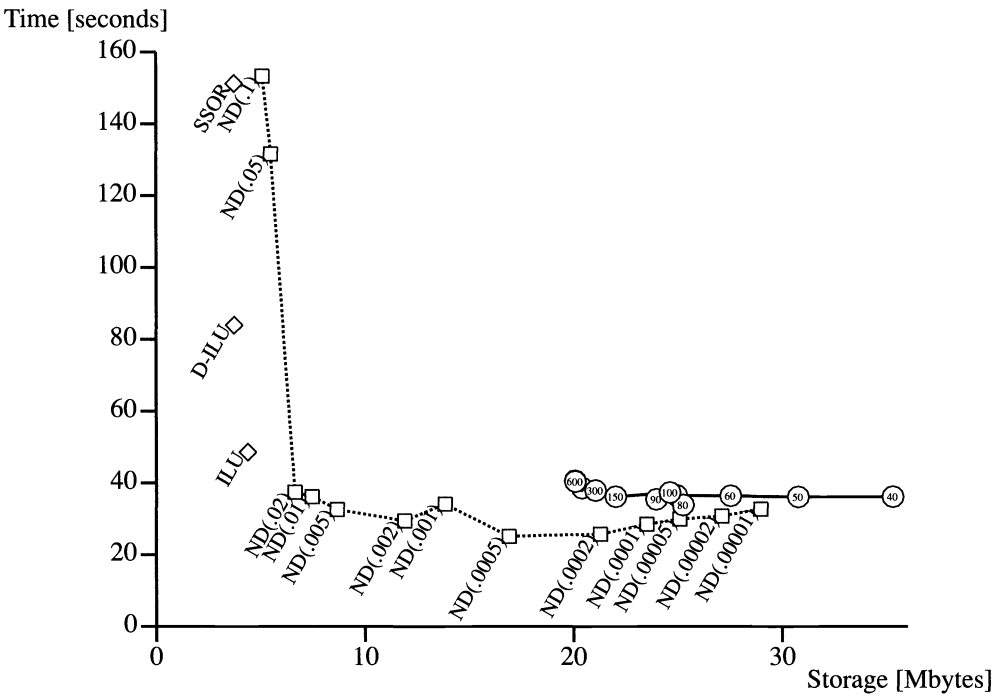


FIG. 3. *Storage size and solution time for an ill-conditioned linear system on a Cray* Y-MP 4D.

2. ND(τ) uses a considerable amount of integer storage. One integer takes 8 bytes on a Cray, but only 4 bytes on a Convex. Therefore, the preconditioner requires almost twice as much memory on the Cray, even if the number of fill entries is the same.

Also, the shape of the curves is not exactly the same. This is due to the differences in the floating-point formats on the two machines that lead to different rounding effects and sometimes ultimately to different iteration numbers.

5. Conclusions. The usefulness and the performance of preconditioned iterative solvers for numerically difficult large-scale applications with irregular structure is, to a great extent, governed by storage-related aspects.

The size of the memory requirements for direct sparse solvers limits their range of application for today's problem sizes and storage capacities. These requirements do not only increase superlinearly with problem size; the exponent of the leading term also increases with the dimensionality of the problem. Iterative solvers are much less restricted by storage limitations.

Mflops rates are of little help in evaluating different iterative methods. The fastest algorithms in terms of computation time do not always correspond to the methods with the highest performance numbers. Mflops do give acceptable figures with which to compare an iterative solver's performance on different machines, as long as the problem and the solution method are held constant.

The performance of iterative solvers does not necessarily increase with problem size. A growth in the problem complexity (dimension of the domain, number of PDEs, irregularity) leads to higher demands on the memory access. Memory aspects (latency, bandwidth, size), rather than pure computation speed, largely dominate the performance of iterative solvers on today's architectures.

Approximate factorizations with numerical dropping and nested iterative solvers are new preconditioners that are able to solve even very ill-conditioned linear systems. In essence, they trade memory for convergence speed. The solution time is always some convex function of the memory requirements, but the exact relations and the optimum value depend strongly on the characteristics of the machine.

Acknowledgments. The initial version of this article was submitted while the first author was still with the Integrated Systems Laboratory of ETH Zürich. The authors would like to express their gratitude to Henk van der Vorst and to Martin Gutknecht for their advice and cooperation concerning their latest iterative methods; to Marco Fillo and to Roland Rühl for their constructive proofreading; to Arno Liegmann for his generous help with the Cray benchmarks in Zürich and at Cray Research Inc. in Eagan MI; to Gernot Heiser and Frank Crawford for porting and benchmarking on the Fujitsu VP-2200 at Australian Supercomputer Technology in Sydney; to the computer centers of ETH in Zürich, EPF in Lausanne; and to CSCS in Manno for providing continued access to their Cray and NEC supercomputers; and especially to Cray Research for supporting our work over several years.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, MA, 1983.

[2] M. A. Ajiz and A. Jennings, *A robust incomplete Cholesky-conjugate gradient algorithm*, Journal for Numer. Methods in Engrg., 20 (1984), pp. 949–966.

[3] R. E. Bank and D. J. Rose, *Global approximate Newton methods*, Numer. Math., 37 (1981), pp. 279–295.

[4] R. E. Bank, D. J. Rose, and W. Fichtner, *Numerical methods for semiconductor device simulation*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 416–435.

[5] R. E. Bank and R. K. Smith, *General sparse elimination requires no permanent integer storage*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 574–584.

[6] J. F. Bürgler, *Discretization and Grid Adaptation in Semiconductor Device Modeling*, Ph.D. thesis, Eidgenössische Technische Hochschule (ETH) Zürich, Switzerland; Hartung-Gorre Verlag, Konstanz, Germany, 1992.

[7] P. Deuflhard, *Global inexact Newton methods for very large scale nonlinear problems*, IMPACT Comput. Sci. Engrg., 3 (1991), pp. 366–393.

[8] J. J. Dongarra, I. S. Duff, D. C. 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.

[9] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, London, 1986.

[10] S. C. Eisenstat, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1–4.

[11] H. C. Elman, *Iterative Methods for Large, Sparse Nonsymmetric Systems of Linear Equations*, Ph.D. thesis, Department of Computer Science, Yale University, New Haven, CT, Apr. 1982.

[12] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Proc. 6th Biennal Dundee Conference on Numerical Analysis, G. A. Watson, ed., United Kingdom, Springer-Verlag, New York, Berlin, 1976.

[13] K. Gallivan, A. Sameh, and Z. Zlatev, *Solving general sparse linear systems using conjugate gradient-type methods*, in 1990 Internat. Conf. on Supercomputing, Amsterdam, June 1990, ACM, pp. 132–139.

[14] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[15] ———, *The evolution of the minimum degree ordering algorithm*, SIAM Rev., 31 (1989), pp. 1–19.

[16] M. H. Gutknecht, *Variants of BiCGStab for Matrices with Complex Spectrum*, SIAM J. Sci. Comput., 14 (1993), pp. 1020–1033.

[17] G. Heiser, *Design and Implementation of a Three-Dimensional, General Purpose Semiconductor Device Simulator*, Ph.D. thesis, Eidgenössische Technische Hochschule (ETH) Zürich, Switzerland; Hartung-Gorre Verlag, Konstanz, Germany, 1992.

[18] 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, 10 (1991), pp. 1218–1230.

[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] K. KELLS, S. MÜLLER, W. FICHTNER, AND G. WACHUTKA, *Simulating temperature effects in multi-dimensional silicon devices with generalized boundary conditions*, in Simulation of Semiconductor Devices and Processes IV, W. Fichtner and A. Aemmer, eds., Hartung-Gorre Verlag, Konstanz, Germany, 1991, pp. 141–148.

[21] 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.

[22] S. MÜLLER, N. HITSCHFELD, C. POMMERELL, K. KELLS, M. WESTERMANN, AND W. FICHTNER, *Technology-CAD algorithms, implementations, and results*, in Proc. 2nd NEC Symposium, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[23] C. H. PAPADIMITRIOU AND K. STEIGLITZ, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[24] M. R. PINTO, *Comprehensive Semiconductor Device Simulation for Silicon ULSI*, Ph.D. thesis, Stanford University, Stanford, CA, 1990.

[25] C. POMMERELL, *Solution of Large Unsymmetric Systems of Linear Equations*, Ph.D. thesis, Eidgenössische Technische Hochschule (ETH) Zürich, Switzerland; Hartung-Gorre Verlag, Konstanz, Germany, 1992.

[26] 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.

[27] C. POMMERELL AND W. FICHTNER, *PILS: An iterative linear solver package for ill-conditioned systems*, in Supercomputing '91, Albuquerque, NM, Nov. 1991, ACM-IEEE, pp. 588–599.

[28] Y. SAAD, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.

[29] ———, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1200–1232.

[30] 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.

[31] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[32] 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.

[33] H. A. VAN DER VORST AND C. VUIK, *GMRESR: a family of nested GMRES methods*, Tech. Rep. 91-80, Dept. of Techn. Math. and Inf., Delft University of Technology, Delft, the Netherlands, 1991.

[34] W. VAN ROOSBROECK, *Theory of flow of electrons and holes in germanium and other semiconductors*, Bell System Tech. J., 29 (1950), pp. 560–607.

[35] P. K. W. VINSOME, *ORTHOMIN – an iterative method for solving sparse sets of simultaneous linear equations*, in Proc. Fourth SPE Symposium on Reservoir Simulation, Paper SPE 5739, Los Angeles, Feb. 1976, Society of Petroleum Engineers, pp. 149–160.

[36] C. VUIK, *Further experiences with GMRESR*, in Copper Mountain Conf. Iterative Methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, Apr. 1992.

[37] D. P. YOUNG, R. G. MELVIN, F. T. JOHNSON, J. E. BUSSOLETTI, L. B. WIGTON, AND S. S. SAMANTH, *Application of sparse matrix solvers as effective preconditioners*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1186–1199.

SIAM J. SCI. COMPUT.
Vol. 15, No. 2, pp. 474–493, March 1994

015

A PARALLEL VERSION OF A MULTIGRID ALGORITHM FOR ISOTROPIC TRANSPORT EQUATIONS*

T. MANTEUFFEL†, S. MCCORMICK†, J. MOREL‡, S. OLIVEIRA§, AND G. YANG†

Abstract. The focus of this paper is on a parallel algorithm for solving the transport equations in a slab geometry using multigrid. The spatial discretization scheme used is a finite element method called the modified linear discontinuous (MLD) scheme. The MLD scheme represents a lumped version of the standard linear discontinuous (LD) scheme. The parallel algorithm was implemented on the Connection Machine 2 (CM2). Convergence rates and timings for this algorithm on the CM2 and Cray-YMP are shown.

Key words. multigrid, parallel algorithms, transport equations

AMS subject classification. 65Y05, 65N22, 65F10, 65Y10, 65N20

1. Introduction.

The description of the neutron transport problem is given in previous papers [1]–[3]. For steady state problems within the same energy group for the isotropic case (by isotropic we mean that the probability of scattering for the particles is the same for all directions), the transport equation in a slab geometry of slab width b becomes

$$\mu \frac{\partial \psi}{\partial x} + \sigma_t \psi = \frac{1}{2}\sigma_s \int_{-1}^{1} \psi(x, \mu') d\mu' + q(x, u), \tag{1.1}$$

for $x \in$ (0,b) and $\mu \in [-1,1]$. Here, $\psi(x, \mu)$ represents the flux of particles at position x traveling at an angle $\theta = \arccos(\mu)$ from the x-axis; $\sigma_t\, dx$ represents the expected number of interactions (absorptive or scattering) that a particle will have in traveling a distance dx; $\sigma_s\, dx$ represents the expected number of scattering interactions; $\sigma_a = \sigma_t - \sigma_s$ represents the expected number of absorptive interactions; and $q(x, \mu)$ represents the particle source. The boundary conditions prescribing particles entering the slab are

$$\psi(0, \mu) = g_0(\mu), \qquad \psi(b, -\mu) = g_1(\mu), \qquad \mu \in (0, 1). \tag{1.2}$$

This problem is difficult for conventional methods to solve in two cases of physical interest:

1. $\gamma = \frac{\sigma_s}{\sigma_t} = 1$ (pure scattering, no absorption);
2. $\frac{1}{\sigma_t} \ll$ b (optically dense).

In fact, as $\sigma_t \to \infty$ and $\gamma \to 1$, the problem becomes singularly perturbed.

Standard discrete approximations to (1.1) and (1.2) will have operators with condition numbers on the order of at least σ_t^2, regardless of the meshsize [1]. This phenomenon presents problems for numerical solution techniques in general and multigrid in particular. In this paper, the discretization used in the spatial dimension is a special finite element method called the modified linear discontinuous (MLD) scheme (described in the next section). To solve the linear system of equations, we use a suitable relaxation process, called two-cell μ-line relaxation, within a multigrid algorithm. The serial version of this algorithm was described

*Received by the editors June 17, 1992; accepted for publication (in revised form) March 22, 1993. This work was supported by Air Force Office of Scientific Research grant AFOSR 86-0126, National Science Foundation grant DMS-8704169, Department of Energy grant DE-FG02-90ER25086, and Los Alamos National Laboratory.

†Program in Applied Mathematics, University of Colorado at Boulder, Boulder, Colorado 80309-0526 (tmanteuf@boulder.colorado.edu), (stevem@boulder.colorado.edu).

‡Computer Research Group (C-3), Los Alamos National Laboratory, Los Alamos, New Mexico 87545 (jim@c3serve.c3.lanl.gov).

§Centre for Mathematics and its Applications, Australian National University, Australia 2601 (suely @thrain.anu.edu.au).

in [3] and the convergence properties were analyzed. It was shown that for $\gamma = 1$ (pure scattering), which is the case we examine in this paper, the algorithm yields a convergence factor on the order of $O((\frac{1}{\sigma_t h})^2)$ when $\sigma_t h \gg 1$ and $O((\sigma_t h)^3)$ when $\sigma_t h \ll 1$. For the two-angle case, it behaves like an exact solver. These convergence rates were proved for special cases in [3] and are substantiated by Fourier analysis in [12]. The Fourier analysis for multigrid based on red-black μ-line relaxation (with numerical results for both Jacobi and red-black Gauss–Seidel relaxation) is shown in [12]. The results show a close agreement between the Fourier analysis and the computational results presented here. The serial algorithm for the case in which $\gamma < 1$ is examined in [4]. The multigrid algorithms were compared to a version of the diffusion synthetic acceleration (DSA) method in [3] and [4]. It was shown to be faster than the DSA in all regimes.

Multigrid algorithms for the transport equations have been examined in [8]–[11]. The algorithm here takes advantage of the rank-1 form of the isotropic scattering operator in the implementation of the two-cell μ-line relaxation (see §3). In this form the two-cell μ-line relaxation is efficient, effective, and parallel. Together with efficient interpolation and restriction operators, this algorithm yields the rates mentioned above (for complete details see [3]). We also mention that a parallel implementation of the DSA algorithm was examined in [13].

In this paper we describe a parallel version of the algorithm. The main benefit of the single instruction multiple data (SIMD) architecture of the CM2 is attained at the relaxation step of the multigrid algorithm, where we take advantage of the structure of the matrix, matching the nonzero entries of very sparse matrices with the relevant processors.

An outline of the remainder of this paper is as follows. In §2 we describe the discretization scheme used. In §3 we show how to accomplish the inversion of the relaxation matrix to take advantage of the SIMD architecture on the CM2. In §4 we describe the multigrid scheme, the interpolation, and the restriction operators used. In §5 we discuss the storage and data structure. In §6 we develop the algorithm for the parallel relaxation and a few implementation details. In §7 we consider a few implementation details of our CM2 multigrid code. In §8 we report on convergence rates and compare these results with the Fourier analysis prediction. In §9 we show the timings obtained with our CM2 parallel codes and compare them with a sequential version of our algorithm on a Cray Y-MP. Finally, in §10 we make a few conclusions.

2. Modified linear discontinuous scheme. The angular discretization is accomplished by expanding the angular dependence in Legendre polynomials and is known as the S_N approximation when the first N Legendre polynomials are used. This results in a semidiscrete set of equations that resemble collocation at N Gauss quadrature points, μ_j, $j = 1, \ldots, N$, with weights w_j, $j = 1, \ldots, N$. Since the quadrature points and weights are symmetric about zero, we reformulate the problem in terms of the positive values, μ_j, $j = 1, \ldots, n$, where $n = \frac{N}{2}$. We define $\psi_j^+(x) = \psi(x, \mu_j)$ and $\psi_j^-(x) = \psi(x, -\mu_j)$ for $j = 1, \ldots, n$. The spatial discretization is accomplished by the MLD scheme, which uses elements that are linear across each cell and discontinuous in the upwind direction. In our grid representation, the variable $\psi_{ij}^{+(-)}$ denotes the flux of particles at position x_i in the direction μ_j ($-\mu_j$). Assuming γ=1, the nodal equations are

$$\frac{\mu_j}{\sigma_t}\frac{\psi^+_{i+\frac{1}{2},j} - \psi^+_{i-\frac{1}{2},j}}{h_i} + \psi^+_{i,j} = \sum_{k=1}^{n} \omega_k(\psi^+_{i,k} + \psi^-_{i,k}) + q^+_{i,j}, \tag{2.1}$$

$$2\frac{\mu_j}{\sigma_t}\frac{\psi^+_{i+\frac{1}{2},j} - \psi^+_{i,j}}{h_i} + \psi^+_{i+\frac{1}{2},j} = \sum_{k=1}^{n} \omega_k\left(\psi^+_{i+\frac{1}{2},k} + 2\psi^-_{i,k} - \psi^-_{i-\frac{1}{2},k}\right) + q^+_{i,j}, \tag{2.2}$$

$$\frac{\mu_j}{\sigma_t}\frac{\psi^-_{i-\frac{1}{2},j}-\psi^-_{i+\frac{1}{2},j}}{h_i}+\psi^-_{i,j}=\sum_{k=1}^{n}\omega_k(\psi^+_{i,k}+\psi^-_{i,k})+q^-_{i,j}, \tag{2.3}$$

and

$$2\frac{\mu_j}{\sigma_t}\frac{\psi^-_{i-\frac{1}{2},j}-\psi^-_{i,j}}{h_i}+\psi^-_{i-\frac{1}{2},j}=\sum_{k=1}^{n}\omega_k\left(\psi^-_{i-\frac{1}{2},k}+2\psi^+_{i,k}-\psi^+_{i+\frac{1}{2},k}\right)+q^-_{i,j}, \tag{2.4}$$

$j=1,\ldots,n$, $i=1,\ldots,m$, with boundary conditions

$$\psi^+_{\frac{1}{2},j}=g^+_{0,j},\quad \psi^-_{m+\frac{1}{2},j}=g^-_{1,j}, \tag{2.5}$$

$j=1,\ldots,n$.

In our model, $x_{i+1/2}$ and $x_{i-1/2}$ are cell edges, $x_i=\frac{1}{2}(x_{i+1/2}+x_{i-1/2})$ is the cell center, and $h_i=x_{i+1/2}-x_{i-1/2}$ is the cell width, $1\le i\le m$. Equations (2.1) and (2.3) are called balance equations and (2.2) and (2.4) are called edge equations. In block matrix form, (2.1)–(2.5) can be written, respectively, as

$$B_i\left(\underline{\psi}^+_{i+\frac{1}{2}}-\underline{\psi}^+_{i-\frac{1}{2}}\right)+\underline{\psi}^+_i=R(\underline{\psi}^+_i+\underline{\psi}^-_i)+\underline{q}^+_i, \tag{2.6}$$

$$2B_i\left(\underline{\psi}^+_{i+\frac{1}{2}}-\underline{\psi}^+_i\right)+\underline{\psi}^+_{i+\frac{1}{2}}=R\left(\underline{\psi}^+_{i+\frac{1}{2}}+2\underline{\psi}^-_i-\underline{\psi}^-_{i-\frac{1}{2}}\right)+\underline{q}^+_{i+\frac{1}{2}}, \tag{2.7}$$

$$B_i\left(\underline{\psi}^-_{i-\frac{1}{2}}-\underline{\psi}^-_{i+\frac{1}{2}}\right)+\underline{\psi}^-_i=R(\underline{\psi}^+_i+\underline{\psi}^-_i)+\underline{q}^-_i, \tag{2.8}$$

$$2B_i\left(\underline{\psi}^-_{i-\frac{1}{2}}-\underline{\psi}^-_i\right)+\underline{\psi}^-_{i-\frac{1}{2}}=R\left(\underline{\psi}^-_{i-\frac{1}{2}}+2\underline{\psi}^+_i-\underline{\psi}^+_{i+\frac{1}{2}}\right)+\underline{q}^-_{i-\frac{1}{2}}, \tag{2.9}$$

$$\underline{\psi}^+_{\frac{1}{2}}=\underline{g}^+_0,\ \underline{\psi}^-_{m+\frac{1}{2}}=\underline{g}^-_1, \tag{2.10}$$

$i=1,\ldots,m$. Here,

$$B_i=\begin{bmatrix}\frac{\mu_1}{\sigma_t h_i} & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \frac{\mu_n}{\sigma_t h_i}\end{bmatrix}\text{ and }R=\begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix}\begin{bmatrix}\omega_1 & \cdots & \omega_n\end{bmatrix}, \tag{2.11}$$

where $\mu_1,\mu_2,\ldots,\mu_n$ are the positive Gauss quadrature points, $w_1,w_2,\ldots,w_n$ are the Gauss quadrature weights, and $\underline{\psi}^{+(-)}_i$ is an n-vector: $\underline{\psi}^{+(-)}_i=(\psi^{+(-)}_{i1},\ldots,\psi^{+(-)}_{in})^T$.

In the computational grid, the inflow for positive angles is on the left of each cell and for the negative angles it is on the right. For a μ-line relaxation, the inflows of each cell are assumed to be known. This is the same as using the values of these variables from the previous iteration. Figure 1 shows the computational domain with $2m+1$ spatial points and n angles.

Consider cell i. In μ-line relaxation cell centers, $\underline{\psi}^+_i$ and $\underline{\psi}^-_i$, together with the outflow variables, $\underline{\psi}^-_{i-1/2}$ and $\underline{\psi}^+_{i+1/2}$, will be updated using the following matrix equation:

$$\begin{bmatrix} I+2B_i-R & -2R & -2B_i & R\\ 0 & I-R & -R & B_i\\ B_i & -R & I-R & 0\\ R & -2B_i & -2R & I+2B_i-R\end{bmatrix}\begin{bmatrix}\underline{\psi}^-_{i-\frac{1}{2}}\\ \underline{\psi}^+_i\\ \underline{\psi}^-_i\\ \underline{\psi}^+_{i+\frac{1}{2}}\end{bmatrix}=\begin{bmatrix}0\\ B_i\underline{\psi}^+_{i-\frac{1}{2}}\\ B_i\underline{\psi}^-_{i+\frac{1}{2}}\\ 0\end{bmatrix}+\begin{bmatrix}\underline{q}^-_{i-\frac{1}{2}}\\ \underline{q}^+_i\\ \underline{q}^-_i\\ \underline{q}^+_{i+\frac{1}{2}}\end{bmatrix}. \tag{2.12}$$

Solving this matrix equation corresponds to performing a μ-line relaxation. In our implementations we perform two-cell relaxations for the whole domain. In this kind of relaxation we consider pair of cells coupled together. This relaxation yields an error that is linear, independent of angle, and continuous across two cells up to $O(\frac{1}{\sigma_t h})$ accuracy when $\sigma_t h \gg 1$ [3]. Instead of updating four variables with a relaxation scheme, we update eight variables. For example, in Fig. 1, variables $\underline{\psi}^-_{i-1/2}$, $\underline{\psi}^-_i$, $\underline{\psi}^+_i$, $\underline{\psi}^-_{i+1/2}$, $\underline{\psi}^+_{i+1/2}$, $\underline{\psi}^-_{i+1}$, $\underline{\psi}^+_{i+1}$, and $\underline{\psi}^+_{i+3/2}$, will be updated. Notice that the inflow variables $\underline{\psi}^+_{i-1/2}$ and $\underline{\psi}^-_{i+3/2}$ will be used from the previous iteration.

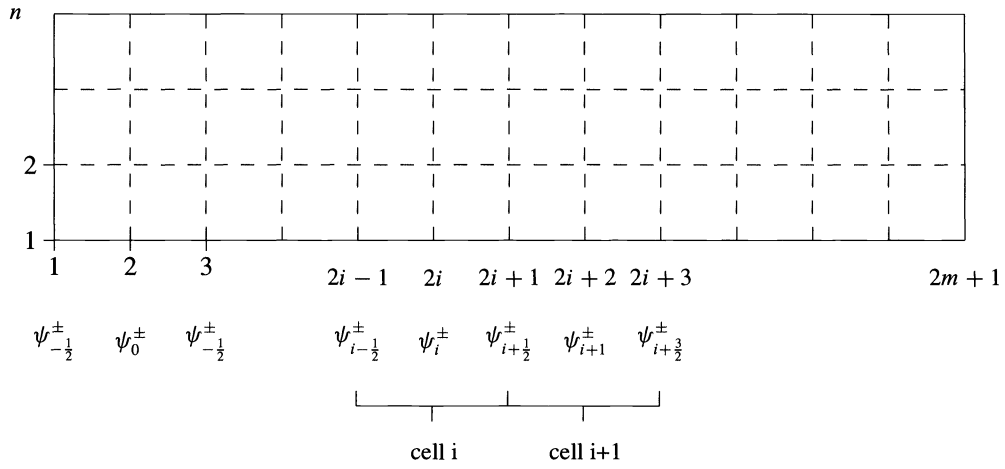


FIG. 1. *Computational grid.*

The two-cell μ-line relaxation involves inversion of the $8n \times 8n$ matrix

(2.13)
$$\begin{bmatrix}
I+2B_i-R & -2R & -2B_i & R & 0 & 0 & 0 & 0 \\
0 & I-R & -R & B_i & 0 & 0 & 0 & 0 \\
B_i & -R & I-R & 0 & -B_i & 0 & 0 & 0 \\
R & -2B_i & -2R & I+2B_i-R & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & I+2B_{i+1}-R & -2R & -2B_{i+1} & R \\
0 & 0 & 0 & -B_{i+1} & 0 & I-R & -R & B_{i+1} \\
0 & 0 & 0 & 0 & B_{i+1} & -R & I-R & 0 \\
0 & 0 & 0 & 0 & R & -2B_{i+1} & -2R & I+2B_{i+1}-R
\end{bmatrix}.$$

The order of variables for this matrix are

$$\underline{\psi} = \left(\underline{\psi}^-_{i-\frac{1}{2}}, \underline{\psi}^+_i, \underline{\psi}^-_i, \underline{\psi}^+_{i+\frac{1}{2}}, \underline{\psi}^-_{i+\frac{1}{2}}, \underline{\psi}^+_{i+1}, \underline{\psi}^-_{i+1}, \underline{\psi}^+_{i+\frac{3}{2}}\right)^T. \tag{2.14}$$

The right-hand side for the two-cell μ-line relaxation is given by

$$\left(0, B_i\underline{\psi}^+_{i-\frac{1}{2}}, 0, 0, 0, 0, B_{i+1}\underline{\psi}^-_{i+\frac{3}{2}}, 0\right)^T. \tag{2.15}$$

In our implementations we developed a multigrid scheme that uses either a block Jacobi relaxation or a block red-black Gauss–Seidel relaxation in parallel. For the Jacobi relaxation, all of the two-cell relaxations will be performed simultaneously for the whole domain. For red-black relaxation, we update half of the total number of cell-pairs during the first (red) stage of the algorithm and the other half during second (black) stage of the algorithm, each time skipping neighboring cell-pairs. To illustrate, Fig. 2 shows four cells. The even numbers represent the cell centers, the odd numbers represent the cell edges. For simplicity, we omitted the vertical lines (passing through each spatial grid point) that contain the Gauss quadrature points (angles). For a Jacobi type of relaxation process, variables at x points 5–9 will be updated at the same time as points 1–5. Each pair of cells will use the relaxation matrix just described. On a parallel machine like the CM2, red-black relaxation takes approximately twice as much CPU time as a Jacobi relaxation. This is partially offset by a better convergence rate (refer to the Fourier analysis in [12] and numerical results in §8).

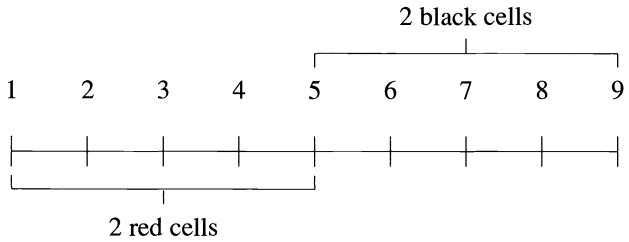


FIG. 2. *Red-black relaxation for four cells.*

3. Two-cell inversion. The inversion of the two-cell relaxation matrix can be done in a very suitable way for SIMD computers like the CM2. First, notice that the matrix can be written as the difference of an easily inverted matrix, which we call A, and a rank-four matrix, which we write as VW^T. The matrix A has a special structure that allows its inverse to be calculated basically in place. It is given by

(3.1)

$$A = \begin{bmatrix}
I+2B_i & 0 & -2B_i & 0 & & & & \\
0 & I & 0 & B_i & & & & \\
B_i & 0 & I & 0 & -B_i & & & \\
0 & -2B_i & 0 & I+2B_i & & & & \\
 & & & & I+2B_{i+1} & 0 & -2B_{i+1} & 0 \\
 & & & -B_{i+1} & 0 & I & 0 & B_{i+1} \\
 & & & & B_{i+1} & 0 & I & 0 \\
 & & & & 0 & -2B_{i+1} & 0 & I+2B_{i+1}
\end{bmatrix}_{(8n\times 8n)}$$

This is a block matrix with $n \times n$ blocks. Each nonzero block is diagonal. The rank-four matrix VW^T is defined by

$$V = \begin{bmatrix} \underline{2} & \underline{0} & \underline{0} & \underline{0} \\ \underline{1} & \underline{1} & \underline{0} & \underline{0} \\ \underline{1} & \underline{1} & \underline{0} & \underline{0} \\ \underline{0} & \underline{2} & \underline{0} & \underline{0} \\ \underline{0} & \underline{0} & \underline{0} & \underline{2} \\ \underline{0} & \underline{0} & \underline{1} & \underline{1} \\ \underline{0} & \underline{0} & \underline{1} & \underline{1} \\ \underline{0} & \underline{0} & \underline{2} & \underline{0} \end{bmatrix}_{(8n\times 4)} \tag{3.2}$$

and

$$W^T = \begin{bmatrix} -\frac{1}{2}\underline{w}^T & \underline{w}^T & 0 & \frac{1}{2}\underline{w}^T & 0 & 0 & 0 & 0 \\ \frac{1}{2}\underline{w}^T & 0 & \underline{w}^T & -\frac{1}{2}\underline{w}^T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2}\underline{w}^T & \underline{w}^T & 0 & \frac{1}{2}\underline{w}^T \\ 0 & 0 & 0 & 0 & \frac{1}{2}\underline{w}^T & 0 & \underline{w}^T & -\frac{1}{2}\underline{w}^T \end{bmatrix}_{(4\times 8n)}, \tag{3.3}$$

where $\underline{1} = (1, 1, \ldots, 1)^T, \underline{0} = (0, 0, \ldots, 0)^T, \underline{2} = (2, 2, \ldots, 2)^T$, and $\underline{w}^T = (\omega_1, \omega_2, \ldots, \omega_n)$.

We can use the Sherman–Morrison formula to calculate the inverse of the two-cell μ-line relaxation matrix $M = A - VW^T$:

$$M^{-1} = A^{-1} + A^{-1}VE^{-1}W^TA^{-1}, \tag{3.4}$$

where $E = I - W^TA^{-1}V$. If $\underline{y}$ is an n-vector, we have $M^{-1}\underline{y} = (I + A^{-1}VE^{-1}W^T)A^{-1}\underline{y}$.

The connectivity of A is two interleaved 4×4 block matrices. Thus, inverting A amounts to solving $2n$ tridiagonal matrices, each of size 4×4, which can be done in parallel. Since all matrices are diagonal, and thus commute, we make use of the notation $\frac{M_1}{M_2} = M_1M_2^{-1}$. We have

$$A^{-1} = \begin{bmatrix} \frac{I}{D_i} & 0 & \frac{2B_i}{D_i} & 0 & \frac{2B_i^2}{D_iD_{i+1}} & 0 & \frac{4B_i^2B_{i+1}}{D_iD_{i+1}} & 0 \\ 0 & \frac{I+2B_i}{D_i} & 0 & \frac{-B_i}{D_i} & 0 & 0 & 0 & 0 \\ \frac{-B_i}{D_i} & 0 & \frac{I+2B_i}{D_i} & 0 & \frac{B_i+2B_i^2}{D_iD_{i+1}} & 0 & \frac{(B_i+2B_i^2)(2B_{i+1})}{D_iD_{i+1}} & \\ 0 & \frac{2B_i}{D_i} & 0 & \frac{I}{D_i} & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \frac{I}{D_{i+1}} & 0 & \frac{2B_{i+1}}{D_{i+1}} & 0 \\ 0 & \frac{(B_{i+1}+2B_{i+1}^2)2B_i}{D_iD_{i+1}} & 0 & \frac{B_{i+1}+2B_{i+1}^2}{D_iD_{i+1}} & 0 & \frac{I+2B_{i+1}}{D_{i+1}} & 0 & \frac{-B_{i+1}}{D_{i+1}} \\ 0 & 0 & 0 & 0 & \frac{-B_{i+1}}{D_{i+1}} & 0 & \frac{I+2B_{i+1}}{D_{i+1}} & 0 \\ 0 & \frac{4B_iB_{i+1}^2}{D_iD_{i+1}} & 0 & \frac{2B_{i+1}^2}{D_iD_{i+1}} & 0 & \frac{2Bi+1}{D_{i+1}} & 0 & \frac{I}{D_{i+1}} \end{bmatrix}, \tag{3.5}$$

where $B_i = \operatorname{diag}(\ldots, b_{ij}, \ldots)$, $D_i = \operatorname{diag}(\ldots, 1 + 2b_{ij} + 2b_{ij}^2, \ldots)$, and $b_{ij} = \frac{\mu_j}{\sigma_i h_i}$. The

matrix $E = I - W^T A^{-1} V$ is a 4×4 matrix of the form

$$I - W^T A_0^{-1} V = \begin{bmatrix} d_1 & 0 & a_{11} & b_1 \\ 0 & d_1 & c_1 & a_1 \\ a_{22} & b_2 & d_2 & 0 \\ c_2 & a_2 & 0 & d_2 \end{bmatrix}, \tag{3.6}$$

where

$$d_1 = 1 - 2\sum_{j=1}^{N} \frac{\omega_j + \omega_j b_{ij}}{d_{ij}}, \tag{3.7}$$

$$d_2 = 1 - 2\sum_{j=1}^{N} \frac{\omega_j + \omega_j b_{i+1j}}{d_{i+1j}}, \tag{3.8}$$

$$a_{11} = 2\sum_{j=1}^{N} \frac{\omega_j b_{ij} b_{i+1j} + \omega_j b_{ij}^2 b_{i+1j}}{d_{ij} d_{i+1j}}, \tag{3.9}$$

$$b_1 = -2\sum_{j=1}^{N} \frac{\omega_j b_{ij} + \omega_j b_{ij} b_{i+1j} + \omega_j b_{ij}^2 + \omega_j b_{ij}^2 b_{i+1j}}{d_{ij} d_{i+1j}}, \tag{3.10}$$

$$c_1 = -2\sum_{j=1}^{N} \frac{\omega_j b_{ij}^2 b_{i+1j}}{d_{ij} d_{i+1j}}, \tag{3.11}$$

$$a_1 = -2\sum_{j=1}^{N} \frac{\omega_j b_{ij}^2 + \omega_j b_{ij}^2 b_{i+1j}}{d_{ij} d_{i+1j}}, \tag{3.12}$$

$$a_{22} = -2\sum_{j=1}^{N} \frac{\omega_j b_{i+1j}^2 + \omega_j b_{ij} b_{i+1j}^2}{d_{ij} d_{i+1j}}, \tag{3.13}$$

$$b_2 = -2\sum_{j=1}^{N} \frac{\omega_j b_{ij} b_{i+1j}^2}{d_{ij} d_{i+1j}}, \tag{3.14}$$

$$c_2 = -2\sum_{j=1}^{N} \frac{\omega_j b_{i+1j} + \omega_j b_{ij} b_{i+1j} + \omega_j b_{i+1j}^2 + \omega_j b_{ij} b_{i+1j}^2}{d_{ij} d_{i+1j}}, \tag{3.15}$$

$$a_2 = -2\sum_{j=1}^{N} \frac{\omega_j b_{ij} b_{i+1j} + \omega_j b_{ij} b_{i+1j}^2}{d_{ij} d_{i+1j}}, \tag{3.16}$$

and E^{-1} is also a 4×4 matrix and can be calculated analytically. Its entries are shown in Appendix A.

4. Multigrid. To illustrate the multigrid scheme, we consider it in a two-grid form. Let L^h denote the fine grid operator, L^{2h} the coarse grid operator, and I_{2h}^h and I_h^{2h} the interpolation and restriction operators, respectively. Let ν_1 and ν_2 be small integers (e.g., $\nu_1 = \nu_2 = 1$), which determine the number of relaxation sweeps performed before and after the coarse grid correction. Then one multigrid $V(\nu_1, \nu_2)$ cycle is represented (in two-grid form) by the following:

1. Relax ν_1 times on $L^h u^h = f^h$.
2. Calculate the residual $r^h = f^h - L^h u^h$.
3. Solve approximately $L^{2h} u^{2h} = I_h^{2h} r^h$.
4. Replace $u^h \leftarrow u^h + I_{2h}^h u^{2h}$.
5. Relax ν_2 times on $L^h u^h = f^h$.

Our multigrid scheme is applied with regard to the spatial variable only. For a multilevel scheme in angle, see [7]. Figure 3 illustrates grid points on the fine grid and on the coarse grid after the a restriction operator is applied to the residual. The interpolation and restriction operators used here are based on the same finite element principle as in the derivation of the MLD scheme. They are defined as follows:

$$I_h^{2h} = \begin{bmatrix} S_{1,1} & S_{1,2} & & & & \\ & & S_{3,1} & S_{3,2} & & \\ & & & & \ddots & \ddots \\ & & & & S_{m-1,1} & S_{m-1,2} \end{bmatrix}, \tag{4.1}$$

where

$$S_{i,1} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & \frac{h_i}{h_{i+1}+h_i} I & 0 & 0 \\ 0 & 0 & \frac{h_i}{h_{i+1}+h_i} I & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } S_{i,2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{h_{i+1}}{h_{i+1}+h_i} I & 0 & 0 \\ 0 & 0 & \frac{h_{i+1}}{h_{i+1}+h_i} I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}$$

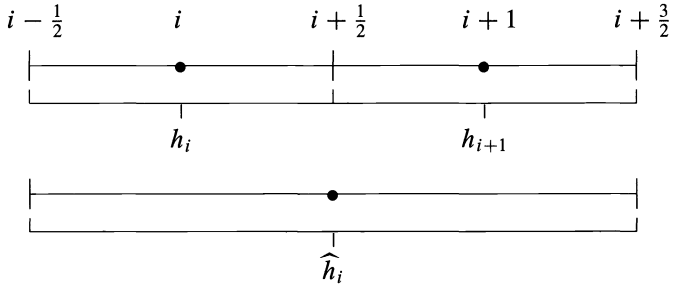


FIG. 3. *Fine and coarse grid.*

and

$$I_{2h}^{h} = \begin{bmatrix} T_{1,1} & & & & \\ T_{2,1} & & & & \\ & T_{1,3} & & & \\ & T_{2,3} & & & \\ & & \ddots & & \\ & & \ddots & & \\ & & & T_{1,m-1} & \\ & & & T_{2,m-1} & \end{bmatrix}, \tag{4.2}$$

where

$$T_{1,i} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & \frac{2h_{i+1}+h_i}{h_{i+1}+h_i} I & 0 & \frac{-h_{i+1}}{h_{i+1}+h_i} I \\ \frac{h_{i+1}}{h_{i+1}+h_i} I & 0 & \frac{h_i}{h_{i+1}+h_i} I & 0 \\ 0 & \frac{2h_{i+1}}{h_{i+1}+h_i} I & 0 & \frac{h_i-h_{i+1}}{h_{i+1}+h_i} I \end{bmatrix}$$

and

$$T_{2,i} = \begin{bmatrix} \frac{h_{i+1}-h_i}{h_{i+1}+h_i} I & 0 & \frac{2h_i}{h_{i+1}+h_i} I & 0 \\ 0 & \frac{h_{i+1}}{h_{i+1}+h_i} I & 0 & \frac{h_i}{h_{i+1}+h_i} I \\ \frac{-h_i}{h_{i+1}+h_i} I & 0 & \frac{2h_i+h_{i+1}}{h_{i+1}+h_i} I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}.$$

The order of variables for these operators is the same as in (2.14). The coarse grid operator, L^{2h}, is defined as

$$L^{2h} = I_{2h}^{h} L^{h} I_{h}^{2h},$$

where L^h is given by (2.6)–(2.9). The coarse grid operator L^{2h} has the same form as L^h, but on a new grid.

We use the notation $2h$ to indicate a coarse grid, although our grids are not really assumed to be uniform. In the general nonuniform case, the meshsize at cell i on the coarse grid is given by $\widehat{h_i} = h_{2i-1} + h_{2i}$.

5. Storage and data structure. Implementation of this scheme on the CM2 involves the following considerations.

- We assign one processor to each point (i.e., for each (x_i, μ_j) pair) in the computational mesh, assuming there are at least $(2m + 1)n$ processors. If we are using less than $(2m + 1)n$ physical processors, the CM2 will reuse the physical processors, creating virtual processors to store the additional grid points.
- Variables that are defined at different grid levels have an index that represents the grid level. We store these variables such that if they represent the same spatial and angular coordinate point, they will be assigned to the same processor even if their grid levels are distinct.

This avoids communication when data is accessed from different grid levels for a given spatial and angle coordinate variable. For example, variable $\underline{\psi}$ appears as ψ(:serial,:news,:news) in the code. Here, the first index represents the grid level, and the remaining indices represent spatial and angular coordinates.

• In the relaxation for the nonuniform grid (variable h), some processors will require products of variables that belong to different cells, as can be seen, for example, in any row of (3.5). In particular, look at the first row of this matrix. Even though this row will have its elements stored in a processor in the first cell of the two-cell pair, it uses data from both cells (i.e., $B_i^2 B_{i+1}$). To avoid extra complication in the parallel algorithm, we store these variables in the proper processors at the outset. For example, we have an array b that stores the values of matrix B for the first cell and an array $bp1$ that stores the values of matrix B for the second cell of the pair. In this way all the processors representing grid points of a two-cell pair will contain data from both cells of the pair.

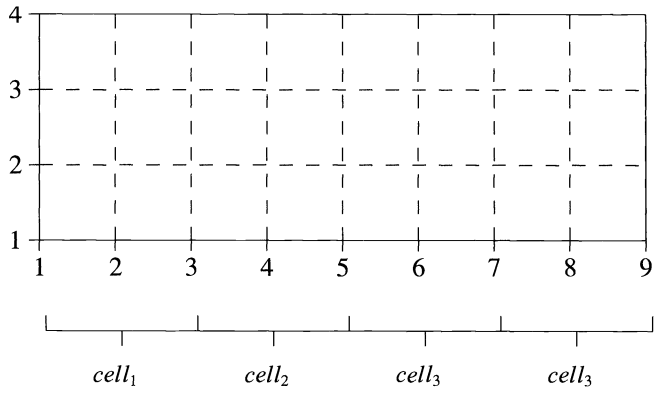


FIG. 4. *Computational mesh for four cells.*

Figure 4 illustrates the computational mesh for four cells (horizontal axis) and four angles (vertical axis). Vertical lines 1–5 represent the variables associated with $cell_1$ and $cell_2$, i.e.,

$$\left(\underline{\psi}^{+(-)}_{-\frac{1}{2}}, \underline{\psi}^{+(-)}_{1}, \underline{\psi}^{+(-)}_{\frac{3}{2}}, \underline{\psi}^{+(-)}_{2}, \underline{\psi}^{+(-)}_{\frac{5}{2}}\right),$$

while vertical lines 5–9 represent the variables associated with $cell_3$ and $cell_4$,

$$\left(\underline{\psi}^{+(-)}_{\frac{5}{2}}, \underline{\psi}^{+(-)}_{3}, \underline{\psi}^{+(-)}_{\frac{7}{2}}, \underline{\psi}^{+(-)}_{4}, \underline{\psi}^{+(-)}_{\frac{9}{2}}\right).$$

Processors represented by vertical lines 1–4 will contain all the information associated with $cell_1$ and $cell_2$, while processors represented by vertical lines 5–8 will contain all the information associated with $cell_3$ and $cell_4$. Consider $cell_1$ and $cell_2$. Array b contains the values of $b(i, j) = \mu(j)/(\sigma_t(cell_1)h(cell_1))$. Thus $bp(1,3) = bp(2,3) = bp(3,3) = bp(4,3)$. Array $bp1$ contains the values of $bp1(i, j) = \mu(j)/(\sigma_t(cell_2)h(cell_2))$. Thus $bp1(1,3) = bp1(2,3) = bp1(3,3) = bp1(4,3)$. However, $b(1, 1) \neq b(1, 2)$, $bp1(1, 1) \neq bp1(1, 2)$, and so on. The arrays b and $bp1$ are constant on the same subsets of the computational mesh. We also use some replication of data in a slightly different way for the mesh size variable h. The elements of array h contain the cellsize h_i, the array $hm1$ contains h_{i-1}, and the array $hp1$ contains h_{i+1}. These three arrays are constant on vertical lines in Fig. 4. This data structure is the same for all the grid levels, but with a different number of cells for each grid level.

• Some communication, of course, is necessary between processors. In fact, between the boundaries of every pair of cells we will have some shifting of data. For example, consider the variables (2.14) updated with the appropriate two-cell relaxation matrix ($cell_i$ and $cell_{i+1}$). Note that $\underline{\psi}^+_{i+3/2}$ will be updated by this matrix, but the variables $\underline{\psi}^-_{i+3/2}$ will be updated by the next two-cell pair relaxation matrix ($cell_{i+2}$ and $cell_{i+3}$). Again, consider Fig. 4. Vertical line 5 represents the variables $\underline{\psi}^{(+)(-)}_{5/2}$. The two-cell relaxation matrix for $cell_1$ and $cell_2$ will be stored on the processors associated with vertical lines 1–4. So, to update variables $\underline{\psi}^+_{5/2}$, the processors on line 5 will need to access data from line 4.

• In the CM2, conformable arrays (arrays with the same shape and size) are always stored in the same set of processors in the same order. To compute expressions like the elements of the matrix E in (3.6) without the need for communication, we store the variables w_j (a one-dimension array) as a two-dimension array that is conformable with array b (Fig. 4). Consequently, w is constant over horizontal lines in Fig. 4. Note that the elements of E^{-1} in (A.1) will also be conformable with variables w and b.

6. Relaxation in parallel. A two-cell relaxation step consists of applying M^{-1} (M is given by (2.13)) to a right-hand side vector that we call $\underline{y}$. What follows is the process to perform a two-cell relaxation step. If the current grid has k cells, then $\frac{k}{2}$ two-cell relaxations will be performed simultaneously for Jacobi, while $\frac{k}{4}$ two-cell relaxations steps will be performed in each of two stages for red-black Gauss–Seidel. Each two-cell relaxation can be performed through the following steps.

Step 1. Form $\underline{z} = A^{-1}\underline{y}$. This step is done at the same time for all the cells, using the matrix A^{-1} as shown in (3.5). To update any variable, the associated processor needs only one row of this matrix. For example, with the order of variables used and shown in (2.14), to update the variable $\underline{\psi}^+_i$, we need only the second row of A^{-1}; so, the processor that updates this variable needs only to store this row's entries. Note, though, that this row's entries will multiply data that are allocated in other processors. Through the use of the CM2 intrinsic function *eoshift*, we can perform a move of data in the x direction (index i), and then multiply by the appropriate entry of the matrix. To update all the variables, we thus have the following combination of shifts:

$$\psi^+_i = A_2\psi^+_i + A_1\psi^+_{i-\frac{1}{2}} + A_3\psi^+_{i+\frac{1}{2}} + A_4\psi^+_{i+1} + A_5\psi^+_{i+\frac{3}{2}}$$

for j =1, ..., n (at the same time). Here, A_1, A_2, A_3, A_4, and A_5 are the elements of the appropriate row. Note that processor (i, j) does not have variables other than the ones with indexes i, j. It will therefore use the CM2 intrinsic function *eoshift*[1] in the following way:

$$temp = \psi_{i+\frac{1}{2}} = eoshift(\psi_i, 1, 1),$$
$$\psi_{i+1} = eoshift(temp, 1, 1),$$

and so on. Each new shift will take advantage of the previous one. The big advantage of this step is that this updating occurs at the same time for all the processors, making this matrix multiplication a job that requires only eight communications. Note that for each cell the vector $\underline{z}$ has length $8n$.

Step 2. Form $VE^{-1}W^T\underline{z}$. In this step we take advantage of the fact that the $8n \times 8n$ matrix $VE^{-1}W^T$ can be written as the tensor product $F \otimes R = F \otimes \underline{1}\underline{w}^t$. Remember that $\underline{1}$ and $\underline{w}^t$ are n-vectors. The 8×8 matrix F is derived from a combination of rows and columns of the matrix E^{-1} and is given by

[1]The first argument of the *eoshift* function is the array name, the second is the array index that is going to be shifted, and the last is the stride of the shift.

$$(6.1)\quad F = \begin{bmatrix} 2 & 0 & & & & & & \\ 1 & 1 & & & & & & \\ 1 & 1 & & & & & & \\ 0 & 2 & & & & & & \\ & & 2 & 0 & & & & \\ & & 1 & 1 & & & & \\ & & 1 & 1 & & & & \\ & & 0 & 2 & & & & \end{bmatrix} E^{-1} \begin{bmatrix} \frac{1}{2} & 1 & 0 & -\frac{1}{2} & & & & \\ -\frac{1}{2} & 0 & 1 & \frac{1}{2} & & & & \\ & & & & \frac{1}{2} & 1 & 0 & -\frac{1}{2} \\ & & & & -\frac{1}{2} & 0 & 1 & \frac{1}{2} \end{bmatrix}.$$

Note that if the grids are nonuniform, the matrices E^{-1} and F will have their elements as arrays and will be different for each distinct pair of cells. Note that we can write $F \otimes \underline{1}\underline{w}^t = (I_8 \otimes \underline{1})F(I_8 \otimes \underline{w}^t)$, where I_8 is the 8×8 identity matrix. Thus, multiplication by the matrix $F \otimes \underline{1}\underline{w}^t$ can thus be done in three steps.

Step 2.1. Form $z^* = (I_8 \otimes \underline{w}^T)\underline{z}$. This step consists of a dot product of $\underline{w}^t$ and each n-vector that makes up the $8 \times n$-vector $\underline{z}$. Remember, vector $\underline{z}$ was obtained in Step 1 and $\underline{w}^t$ is an n-vector. So, first we multiply each entry of $\underline{w}^t$ by each corresponding entry of $\underline{z}$: $w_{ij} \times z_{ij}$ which is performed for all processors at the same time because $\underline{w}$ is conformable to $\underline{z}$. The i index for variable z_{ij} goes from 1 to 8 for each cell. The second stage of the dot product is to perform a summation in the angle index direction. This is done with the use of the CM2 intrinsic function *sum*. The vector $\underline{z}^*$ has length 8.

Step 2.2. Form $\underline{\eta} = \mathrm{F}\,\underline{z}^*$. This step is a multiplication of a vector by a matrix, and can be performed in the CM2 similar to the way Step 1 of this relaxation was performed. Depending on which point the processor is representing, it will have stored different rows of the matrix F. Its variables will be updated through the use of the CM2 intrinsic function *eoshift* only (in this case, 16 of them).

Step 2.3. Form $\Theta = (I_8 \otimes \underline{1})\underline{\eta}$. This spreads the result of Step 2.1 across all the angles (for all spatial points at the same time). This is done with the use of the CM2 intrinsic function *spread*.

Step 3. Form $A^{-1}\underline{\Theta}$ as in Step 1 and add the result to $\underline{z}$.

A special remark should be made here. It is not hard to see that Steps 2.2 and 2.3 can have their order interchanged. In fact, since all the CM2 processors will execute the same instruction simultaneously, unless instructed otherwise by the use of masks, we make use of this property of the algorithm. We spread vector z^* throughout all the angles represented by different processors. Since all the processors had the appropriate row elements of matrix F from the outset, we perform Step 2.2 last. This will make the use of masks necessary only for distinction between the processors regarding the kind of spatial point it represents, not the kind of angle.

It is easy to show that, if we take advantage of the structure of the various matrices and perform the matrix multiplication steps as described, the total operation count and the number of communications per processor for one relaxation sweep are both $O(n)$, more precisely $2(n-1)$. Details are given in [12].

7. Multigrid in parallel. The parallel relaxation process of the multigrid algorithm was explained in §6. For calculation of the residual on the finer grid, again we use the CM2 intrinsic function *eoshift*. For example, to calculate the residual at an edge grid point for a

negative angle, we use (2.4). Note that the processor that contains the variable $\psi^-_{i,j}$ will not contain other variables necessary for the calculation of the residual at this point. Therefore, we perform a move of data:

$$\psi^-_{i+\frac{1}{2}} = eoshift(\psi^-_i, 1, 1),$$
$$\psi^-_{i-\frac{1}{2}} = eoshift(\psi^-_i, 1, -1).$$

For calculation of the right-hand side of this equation, we perform a one-step multiplication for the conformable arrays w and ψ^+ (the same for ψ^-), and then, using the CM2 intrinsic function *sum*, we perform a summation in the angle direction for the resultant products.

After the residual is calculated, to solve for the error on the coarser grid, we have to estimate the residual on the coarser grid. This is done by multiplying r^h, residual on the fine grid by the restriction operator I^{2h}_h shown in (4.1). In some processors there will be no operation or communication at all, because as in §3 the grid level index is serial, and some processors will contain all the information they need. For example, for a negative angle at the left-hand side of a two-cell pair (negative angle outflow), the restriction consists of $r^{2h}_{i-1/2} = r^h_{i-1/2}$ (see the first row of (4.1)). This requires no communication.

For processors that represent cell centers on the coarse grid (second and third row of (4.1)), we use the CM2 intrinsic function *eoshift* in a way that is similar to what was done in Step 1 of relaxation.

$$temp\ 1 = r^h_{i+1} = eoshift(r^h_i, 1, 1),$$
$$temp2 = r^h_{i-1} = eoshift(r^h_i, 1, -1).$$

We then calculate the residual on the coarse grid

$$r^{2h}_i = \frac{hp1}{(hp1 + h)} temp1 + \frac{h}{(h + hp1)} temp2.$$

After forming r^{2h}, we perform parallel relaxation again, this time to estimate the error on the coarser grid (u^{2h}).

To use the coarser grid quantities to correct the solution on a finer grid, we multiply u^{2h} by the matrix I^h_{2h} shown in (4.2). This multiplication is performed similarly to the multiplication of r^h by I^{2h}_h.

8. Numerical results. The CM2 codes were run for different values of $\sigma_t h$ on the fine grid. In Tables 1 and 2 we chose these values to be the worst cases predicted by the Fourier analysis shown in [12]. All of the convergence factors (i.e., the ratio of Euclidean norms of the residuals before and after a multigrid V-cycle) shown here are for the case with no absorption ($\gamma = 1$). The results are shown for the two kinds of relaxation (Jacobi and red-black Gauss–Seidel) used in our multigrid scheme. The convergence factors shown for the CM2 codes were obtained for one V-cycle after five V-cycles. The convergence factors were close to the ones predicted by the Fourier analysis as Tables 1 and 2 show. Tables 1 and 2 show results for uniform grids for different number of angles, e.g., S_4 means four scattering angles (n=2) and so on. Table 3 shows convergence factors obtained for nonuniform grids; these grids were generated randomly with h varying between 1 and 100. Table 4 shows a comparison of three relaxation methods, discussed in the next section, and the behavior of the convergence factor for varying $\sigma_t h$ for a two-cell Jacobi ($V(1, 1)$ and $V(2, 2)$) or two-cell red-black Gauss–Seidel ($V(1, 1)$) relaxation. Note that the convergence factor appears to be $O((\frac{1}{\sigma_t h})^2)$ when $\sigma_t h \gg 1$

and $O((\sigma_t h)^3)$ when $\sigma_t h \ll 1$ for all three relaxation methods. To make a fair comparison between the different relaxations, we have to consider the difference in time spent to attain the above convergence factors. This will be analyzed in the next section.

TABLE 1

Worst-case convergence factor for multigrid with Jacobi V(1, 1) *relaxation, Fourier analysis, and* CM2 *results (for* $m = 512$).

Angles	$S4$	$S8$	$S16$	$S32$
$\sigma_t h$	.230	.100	.150	.400
Fa	.012	.012	.013	.13
CM2	.016	.015	.016	.017

TABLE 2

Worst-case convergence factor for multigrid with red-black Gauss–Seidel relaxation, Fourier analysis, and CM2 *results* ($m = 512$).

Angles	$S4$	$S8$	$S16$	$S32$
$\sigma_t h$	.150	.240	.100	.200
Fa	.0042	.0045	.0043	.0045
CM2	.0063	.0065	.0086	.0045

TABLE 3

Convergence factors for nonuniform cells (Jacobi and red-black Gauss–Seidel relaxation on the CM2), $1 \leq h \leq 100$ *and* $\sigma_t = 1$.

Angles	$S4$	$S8$	$S16$
Jacobi	1.6×10^{-4}	1.3×10^{-4}	5.6×10^{-4}
Red-black	1.3×10^{-4}	1.0×10^{-4}	9.7×10^{-5}

TABLE 4

Convergence factors for the Jacobi ($V(1, 1)$ *and* $V(2, 2)$) *and red-black Gauss–Seidel* ($V(1, 1)$), *for various values of* $\sigma_t h$, S_4 *case with* $m = 512$.

$\sigma_t h$	Jacobi $V(1, 1)$	Red-black $V(1, 1)$	Jacobi $V(2, 2)$
10^{-5}	9.4×10^{-10}	3.3×10^{-10}	1.47×10^{-10}
10^{-4}	8.7×10^{-7}	2.8×10^{-7}	1.38×10^{-7}
10^{-3}	4.1×10^{-4}	7.2×10^{-5}	7.40×10^{-5}
10^{-2}	1.0×10^{-2}	4.3×10^{-3}	2.41×10^{-3}
10^{-1}	1.5×10^{-2}	8.0×10^{-3}	2.71×10^{-3}
1.	4.2×10^{-3}	1.1×10^{-3}	1.03×10^{-3}
10^1	3.9×10^{-5}	3.2×10^{-5}	1.86×10^{-5}
10^2	3.4×10^{-7}	2.5×10^{-7}	4.16×10^{-7}

In Table 5 we make $\sigma_t h$ equal to .1 and vary the number of cells (m). For a fixed grid meshsize the convergence factor is basically constant. The proof of convergence through

Fourier analysis for uniform meshs and additional results are shown in [12]. An analytical proof of convergence is also shown in [3].

TABLE 5

Convergence factors for Jacobi ($V(1,1)$ and $V(2,2)$ cycle) and red-black Gauss–Seidel ($V(1,1)$) on the CM2. *S4 case and* $\sigma_t h = .1$.

m	Jacobi $V(1,1)$	Red-black $V(1,1)$	Jacobi $V(2,2)$
512	1.5×10^{-2}	9.5×10^{-3}	2.7×10^{-3}
1024	1.5×10^{-2}	9.7×10^{-3}	2.8×10^{-3}
2048	1.5×10^{-2}	9.8×10^{-3}	2.9×10^{-3}
4096	1.5×10^{-2}	9.9×10^{-3}	3.0×10^{-3}
32768	1.6×10^{-2}	9.9×10^{-3}	3.2×10^{-3}
65536	1.6×10^{-2}	9.9×10^{-3}	3.1×10^{-3}

9. Timings. First, we show the cost behavior when we keep n constant (equal to one) and vary the spatial dimension m (Figs. 5 and 6 and Table 6). Second, we analyze the cost when n varies and the spatial dimension m is kept constant (Fig. 7). All the results shown in this section were measured for a $V(1,1)$ or $V(2,2)$ cycle. There are two codes, one for uniform grids and the other for nonuniform grids. The uniform grid code was used for the timings in this paper. However, similar results have been obtained with the nonuniform grid code, but with all of the timings roughly doubled. The sequential timings were measured using one processor of a Cray Y-MP, which has a vector architecture.

When we increase the number of grid points of the computational grid, we should have no increase on the time spent for our relaxation step on computers like the CM2 until all the processors are being used. However, in our multigrid relaxation scheme, the number of levels should be defined such that the coarsest grid consists of two cells. Hence the number of grid levels will always be $\log_2 \frac{m}{2}$; so, every time we double the number of cells (m) in our finest grid, we have new computations at the new added grid level since the grid level index is serial. This increase in time corresponds only to the new calculations. This can be observed in Fig. 5 for m between 2^6 and 2^9, specifically, where we notice that the increase in time when we double m is linearly proportional to $\log_2 m$, while after $m = 2^9$ there is a bigger increase in the slope of the graph every time we double m because the processors are saturated.

For our example, when m reaches the value of 2^{10}, we have begun to saturate the CM2. Here the number of grid points $(2m + 1)$ is 2049. When these timings were measured we used one sequencer, which consists of 2^{12} processors with 512 floating point units (FPU). Note that 2048 is four times the number of FPUs, where four is the vector-length at each FPU. This explains the beginning of increase for the slopes of the piecewise linear graph in Fig. 5, since we will be reusing these FPUs. Also, when we vary m between 2^{10} and 2^{11}, we are changing the number of grid points by 2048, which causes an even bigger increase on the slopes of Fig. 5. In fact, the timings will increase (disregarding the overhead) every time we further increase the number of cells $(2m + 1)$ by multiples of 2048 (the number of FPU $\times$ 4).

If we look at the timings and dimensions for larger m, the parallel code starts increasing its time proportionally to the increase of m, similarly to the way a sequential code does. Consider the points $m = 2^{15}(32768)$ and $m = 2^{16}(65536)$ in Fig. 6. This is explained by the fact that, for this range of m, every time we increase m we will be reusing FPUs.

Figure 7 shows the variation of cost when n changes. The multigrid in this paper was applied to the spatial variable m, so none of the increase in cost is due to the multilevel scheme used. The increase of cost here is due to the fact of using the CM2 intrinsic functions *sum* and

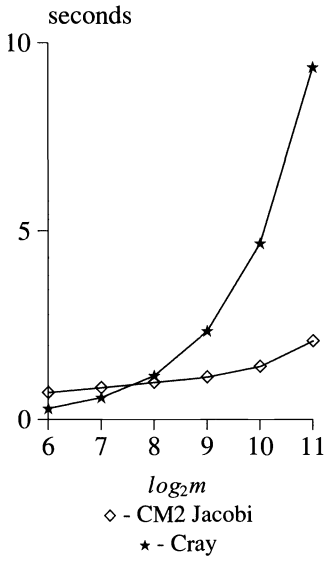


FIG. 5. *Timings for a V* (1, 1) *cycle for small m and n* = 1.

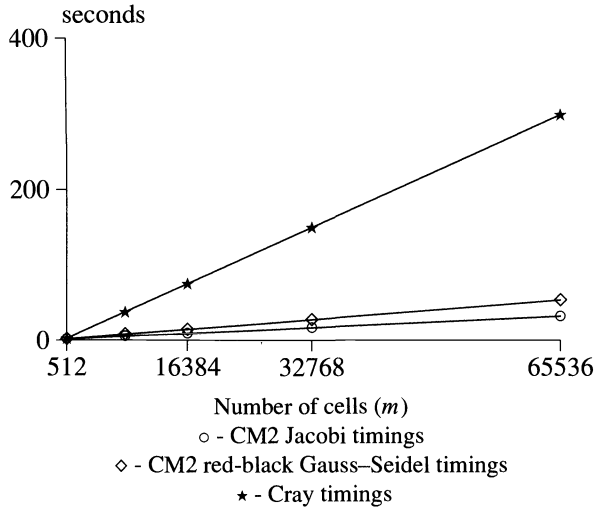


FIG. 6. *Timings for a V* (1, 1) *for large m and n* =*1.*

spread in the angular variable direction. This happens, for example, in Step 2 of the parallel relaxation. Of course, when we increase n to the point of reusing the FPUs, the time cost will increase similarly to the way it did when m increased.

Figure 6 also shows the additional time spent if we use a red-black Gauss–Seidel relaxation instead of a Jacobi. The timings for red-black Gauss–Seidel (V(1,1)) and Jacobi (V(1,1) and V(2,2)) two-cell relaxation are compared to the Cray timings on Table 6. The Cray timings were measured for a V(1,1) cycle. Note that since the code on the Cray is sequential, it is irrelevant for timing purposes whether we use the Jacobi or red-black Gauss–Seidel relaxation since both relaxations will take approximately the same time, $\frac{m}{2}$ sequential steps.

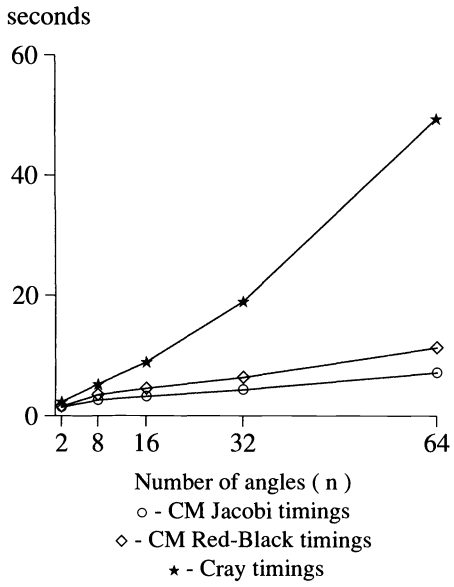


FIG. 7. *Timings for a* $V(1, 1)$ *cycle varying* (n) *and* $m = 512$.

TABLE 6

Timings for Jacobi ($V(1, 1)$ *and* $V(2, 2)$) *and red-black* ($V(1, 1)$) *on the* CM2 *and Cray* Y-MP. *S*4 *case*, $\sigma_t h = .1$.

m	Jacobi $V(1, 1)$	Red-black $V(1, 1)$	Jacobi $V(2, 2)$	Cray
512	1.19	1.78	1.69	2.9
1024	1.93	3.00	2.82	5.85
2048	2.78	4.54	4.35	11.71
4096	4.60	7.57	7.41	23.43
32768	31.00	53.53	52.69	187.5
65536	63.14	109.54	108.6	376.0

In the CM2 for the same number of V-cycles, the Jacobi relaxation in a $V(1, 1)$ cycle is much faster than the red-black Gauss–Seidel relaxation in a $V(1, 1)$ cycle, but its convergence factor is generally worse than the red-black $V(1, 1)$ convergence rate (Table 4). If we consider the Jacobi relaxation for the $V(2, 2)$ cycle, the timings will be slightly better than the red-black Gauss–Seidel $V(1, 1)$ cycle and the convergence factors in this case will be comparable. We show here that if the additional time used in doing a V-cycle with red-black Gauss–Seidel relaxation was used to perform more of the Jacobi V-cycles, the convergence factor (both $V(1, 1)$ and $V(2, 2)$) would be generally better.

For a fixed number of V-cycles, Table 7 compares Jacobi $V(1, 1)$ and $V(2, 2)$ with red-black Gauss–Seidel $V(1, 1)$. In this table we compare the convergence factors for each relaxation using the time for a red-black $V(1, 1)$ as the standard unit. For example, if red-black Gauss–Seidel $V(1, 1)$ took twice as long as Jacobi $V(1, 1)$, then two Jacobi $V(1, 1)$ cycles could be performed in the same time as one red-black $V(1, 1)$. Using the time for red-black $V(1, 1)$ as the standard unit of time, the proper convergence factor for Jacobi $V(1, 1)$ would be $(\rho_{j1})^2$. We define

t_{rb} = time for 1 $V(1, 1)$ cycle with red-black Gauss–Seidel relaxation,

t_{j1} = time for 1 $V(1, 1)$ cycle with Jacobi relaxation,

t_{j2} = time for 1 $V(2, 2)$ cycle with Jacobi relaxation.

TABLE 7

Comparison between Jacobi V(1, 1), *Jacobi* V(2, 2), *and red-black* V(1, 1) *convergence factors* (ρ_{j1}, ρ_{j2} *and* ρ_{rb1}) *for various values of* $\sigma_t h$ (S4 *case*).

$\sigma_t h$	$(\rho_{j1})^{\frac{t_{rb1}}{t_{j1}}}$	ρ_{rb1}	$(\rho_{j2})^{\frac{t_{rb1}}{t_{j2}}}$
10^{-5}	2.8×10^{-14}	3.3×10^{-10}	4.4×10^{-11}
10^{-4}	8.1×10^{-10}	2.8×10^{-7}	5.9×10^{-10}
10^{-3}	8.6×10^{-6}	7.2×10^{-5}	4.5×10^{-5}
10^{-2}	1.0×10^{-3}	4.3×10^{-3}	1.7×10^{-3}
10^{-1}	1.8×10^{-3}	8.0×10^{-3}	2.0×10^{-3}
1.	2.7×10^{-4}	1.1×10^{-3}	7.1×10^{-4}
10^{1}	2.4×10^{-7}	3.2×10^{-5}	1.0×10^{-5}
10^{2}	2.0×10^{-10}	2.5×10^{-7}	1.9×10^{-7}

Likewise we define the convergence factors ρ_{rb}, ρ_{j1}, and ρ_{j2}. The correct comparisons are

$$\rho_{rb} \quad \text{vs.} \quad (\rho_{j1})^{\frac{t_{rb}}{t_{j1}}} \quad \text{vs.} \quad (\rho_{j2})^{\frac{t_{rb}}{t_{j2}}}.$$

These comparisons are shown in Table 7. It is obvious that both Jacobi V(1,1) and V(2,2) are superior to red-black Gauss–Seidel. If we compare the Jacobi V(1,1) and V(2,2) relaxations, we conclude that Jacobi V(1,1) gives, in general, a better convergence factor in this parallel environment for the same CPU time.

10. Conclusions. In this paper we have shown a parallel algorithm for a multigrid scheme for solving the transport equations in slab geometry. This algorithm is suitable for SIMD computers and takes advantage of this kind of architecture during the different stages of the multigrid scheme. It was implemented on the CM2 and was much faster than a sequential version of the same algorithm on the Cray Y-MP (using only one processor), especially when comparing large gridsizes. For the relaxation step of the multigrid algorithm, we used the Sherman–Morisson formula and developed an efficient relaxation with the use of a few intrinsic functions on the CM2. The interpolation and restriction operations, for some grid points, were able to be performed without communication. The increase in the CPU time spent in a V-cycle when either one of the grid dimensions was increased was small before saturation. For the spatial dimension this increase is due to the addition of a new grid level, and for the angular dimension it is due to the use of the CM2 intrinsic functions. After saturation of the CM2 is reached, we have shown that the sharper increases in timings are caused by reuse of FPUs. In fact, when the CM2 becomes saturated, the parallel timings start doubling when one of the grid dimensions is doubled. This, of course, always happens in the sequential timings for any doubling of m or n.

The convergence rates attained per V-cycle were extremely good and matched theoretical results. We implemented three different kind of relaxations: Jacobi ($V(1, 1)$ and $V(2, 2)$) and red-black Gauss–Seidel $V(1,1)$ in two kinds of implementation each, uniform and nonuniform meshes. We have shown that in this context, the Jacobi $V(1,1)$-cycle was superior to Jacobi $V(2, 2)$-cycle and red-black Gauss–Seidel $V(1, 1)$-cycle. The timings for the nonuniform grid code do not appear in this paper, since they were obtained under a timesharing environment that was not reliably comparable to the uniform grid code executed under the dedicated environment. However, similar results and conclusions seem to hold, with the exception that all of the times are roughly doubled.

The results in this paper show the good performance of our parallel algorithms for solving the isotropic form of the transport equation on SIMD architectures. For the anisotropic scattering, we are developing an algorithm suitable for SIMD computers [12] using an angular multigrid developed in [7].

Appendix A. The inverse of the matrix E in (3.6) is given by

$$E^{-1} = \begin{bmatrix} e_{11} & e_{12} & e_{13} & e_{14} \\ e_{21} & e_{22} & e_{23} & e_{24} \\ e_{31} & e_{32} & e_{33} & e_{34} \\ e_{41} & e_{42} & e_{43} & e_{44} \end{bmatrix}, \tag{A.1}$$

where

$$e_{11} = \frac{(a_{11}b_2 + b_1a_2)d_2}{\det_1}, \tag{A.2}$$

$$e_{12} = \frac{(a_{11}b_2 + b_1a_2)d_2}{\det_1}, \tag{A.3}$$

$$e_{13} = \frac{[d_1d_2 - (c_1b_2 + a_1a_2)](-a_1) + (a_{11}b_2 + b_1a_2)(-c_1)}{\det_1}, \tag{A.4}$$

$$e_{14} = \frac{[d_1d_2 - (c_1b_2 + a_1a_2)](-b_1) + (a_1 1b_2 + b_1a_2)(-a_1)}{\det_1}, \tag{A.5}$$

$$e_{21} = \frac{(c_1a_{22} + a_1c_2)d_2}{\det_1}, \tag{A.6}$$

$$e_{22} = \frac{[d_1d_2 - (a_{11}a_{22} + b_1c_2)]d_2}{\det_1}, \tag{A.7}$$

$$e_{23} = \frac{[c_1a_{22} + a_1c_2](-a_{11}) + [d_1d_2 - (a_{11}a_{22} + b_1c_2)](-c_1)}{\det_1}, \tag{A.8}$$

$$e_{24} = \frac{[c_1a_{22} + a_1c_2](-b_1) + [d_1d_2 - (a_{11}a_{22} + b_1c_2)](-a_1)}{\det_1}, \tag{A.9}$$

$$e_{31} = \frac{[d_1d_2 - (c_2b_1 + a_1a_2)](-a_{22}) + (a_{22}b_1 + b_2a_1)(-c_2)}{\det_2}, \tag{A.10}$$

$$e_{32} = \frac{[d_1d_2 - (c_2b_1 + a_1a_2)](-b_2) + (a_{22}b_1 + b_2a_1)(-a_2)}{\det_2}, \tag{A.11}$$

$$e_{33} = \frac{[d_1d_2 - (c_2b_1 + a_1a_2)](d_1)}{\det_2}, \tag{A.12}$$

$$e_{34} = \frac{(a_{22}b_1 + a_1b_2)](d_1)}{\det_2}, \tag{A.13}$$

$$e_{41} = \frac{(c_2a_{11} + a_2c_1)(-a_{22}) + [d_1d_2 - (a_{22}a_{11} + b_2c_1)](-c_2)}{\det_2}, \tag{A.14}$$

$$e_{42} = \frac{(c_2a_{11} + a_2c_1)(-b_2) + [d_1d_2 - (a_{22}a_{11} + b_2c_1)](-a_2)}{\det_2}, \tag{A.15}$$

$$e_{43} = \frac{(c_2a_{11} + a_2c_1)(d_1)}{\det_2}, \tag{A.16}$$

$$e_{44} = \frac{d_1d_2 - (a_{11}a_{22} + b_2c_1)(d_1)}{\det_2}, \tag{A.17}$$

$$\det_1 = [d_1d_2 - (c_1b_2 + a_1a_2)][d_1d_2 - (a_{11}a_{22} + b_1c_2)] - (c_1a_{22} + a_1c_2)(a_{11}b_2 + b_1a_2), \tag{A.18}$$

and

$$\det_2 = [d_1d_2 - (c_2b_1 + a_1a_2)][d_1d_2 - (a_{11}a_{22} + b_2c_1)] - (c_2a_{11} + a_2c_1)(a_{22}b_1 + b_2a_1). \tag{A.19}$$

REFERENCES

[1] V. Faber and T. A. Manteuffel, *Neutron transport from the viewpoint of linear algebra*, Transport Theory, Invarient Embedding, and Integral Equations, Lecture Notes in Pure and Appl. Math., Vol. 115, Marcel-Decker, New York, April 1989.

[2] E. E. Lewis and W. F. Miller, *Computational Methods of Neutron Transport*, John Wiley and Sons, New York, 1984.

[3] T. Manteuffel, S. McCormick, J. Morel, S. Oliveira, and G. Yang, *A fast multigrid algorithm for isotropic transport problems*, SIAM J. Sci. Comput., submitted.

[4] T. Manteuffel, S. McCormick, J. Morel, and G. Yang, *A fast multigrid solver for isotropic transport problems with absorption*, SIAM J. Sci. Comput., submitted.

[5] T. Manteuffel, S. McCormick, J. Morel, and S. Oliveira, *A Parallel Multilevel Algorithm for Anisotropic Transport Equations*, manuscript.

[6] J. E. Morel and E. W. Larsen, *A new class of S_N spatial differencing schemes*, Nucl. Sci. Engrg., 1988.

[7] J. E. Morel and T. A. Manteuffel, *An angular multigrid acceleration technique for the S_n equations with highly forwarded-peaked scattering*, Nucl. Sci. Engrg., 107 (1991), pp. 330–342.

[8] A. Nowak, J. E. Morel, and D. R. Harris, *A multigrid acceleration method for one dimensional S_N equations with anisotropic scattering*, Nucl. Sci. Engrg., 102 (1989), pp. 1–23.

[9] P. F. Nowak, *A Coupled Synthetic and Multigrid Acceleration Method for Two-Dimensional Transport Calculations*, Ph.D. thesis, Dept. of Nuclear Engineering, University of Michigan, Ann Arbor, 1988.

[10] P. F. Nowak and E. W. Larsen, *Multigrid problems for S_N problems*, Trans. American Nuclear Society, 57 (1987), pp. 355–356.

[11] P. F. Nowak, *A multigrid method for S_N calculations in x-y geometry*, Trans. of the American Nuclear Society, 56 (1988), pp. 291–292.

[12] S. Oliveira, *Parallel Multilevel Methods For Transport Equations*, Ph.D. thesis, Mathematics Dept., University of Colorado at Denver, May 1993.

[13] M. Yavuz and E. Larsen, *S_N methods on parallel computers*, Nucl. Sci. Engrg., 112 (1992), pp. 32–42.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 495–510, May 1994

ROTATING WAVES FROM HOPF BIFURCATIONS IN EQUATIONS WITH $O(2)$-SYMMETRY*

W. WU†, P. J. ASTON‡, AND A. SPENCE§

Abstract. This paper considers the problem of Hopf bifurcations that occur in equations with $O(2)$-symmetry on a solution that has the full $O(2)$-symmetry. It is well known that at such points, a path of standing waves and a path of rotating waves bifurcates. The standing waves are easily dealt with by restricting attention to a certain fixed point space and so the rotating wave solutions that arise are considered in detail. Efficient numerical methods for the detection and computation of these bifurcation points based on a block diagonalisation of the Jacobian are considered. Also described is a simple means of choosing the basis that gives rise to this block structure. Numerical methods for following the rotating wave branch away from the $O(2)$-symmetric solution are then described. The extension to infinite dimensions is considered briefly before a numerical example is presented.

Key words. rotating waves, $O(2)$-symmetry, Hopf bifurcations

AMS subject classifications. 34A34, 35B32, 65L99

1. Introduction.

For problems with $O(2)$-symmetry, it is well known that Hopf bifurcations on a path of solutions that have the full $O(2)$-symmetry give rise to paths of both standing waves and rotating (or travelling) waves. The form of the bifurcation equations at such a point has been studied in detail by Golubitsky, Stewart, and Schaeffer [7, Chap. XVII]. We consider this problem from a numerical viewpoint, concentrating on efficient methods for detecting such Hopf bifurcation points and then following the paths of rotating waves that occur. We do not consider the standing waves in detail because these solutions arise from a standard Hopf bifurcation in the subspace of functions that are symmetric with respect to the reflection in $O(2)$. Thus, standard methods for dealing with Hopf bifurcations can be employed to compute the branch of standing wave solutions (see, for example, [9]). However, the rotating wave solutions do not have this reflectional symmetry property; as a result, they cannot be isolated by restricting to a particular symmetric subspace of functions. Thus, special methods have to be developed to find these solutions and our approach is to reduce the problem to a steady-state equation to be solved for the rotating wave profile. This equation involves an additional variable, namely, the velocity of the wave. A phase condition for isolating a particular wave profile is an additional equation, which is introduced to match the extra velocity variable.

Rotating waves can also arise from bifurcations on steady-state branches of solutions with dihedral symmetry when the reflectional symmetry is broken. This situation has been analysed from an abstract point of view in [10] and from a numerical aspect in [2]. In this situation, however, the bifurcation is a nonstandard steady state bifurcation associated with a zero eigenvalue of the Jacobian, in contrast to the Hopf bifurcation in the present framework. Another difference between the two cases is that the bifurcation from an $O(2)$-symmetric solution has nonzero velocity on the rotating wave branch at the bifurcation, whereas, for the bifurcation from a branch with dihedral symmetry, the velocity is zero on the rotating wave branch at the bifurcation. This is due to the fact that an $O(2)$ symmetric solution can be considered as a rotating wave solution with arbitrary velocity; this is not true for a solution that does not have the full $O(2)$-symmetry.

There are three main aspects of this paper. Firstly, we construct a basis for the space, which gives rise to a block diagonalisation of the Jacobian matrix. This decomposition then

*Received by the editors November 27, 1991; accepted for publication (in revised form) March 12, 1993.

†Department of Mathematics, University of Jilin, Changchun, China.

‡Department of Mathematical and Computing Sciences, University of Surrey, Guildford GU2 5XH, United Kingdom (p.aston@mcs.surrey.ac.uk).

§School of Mathematical Sciences, University of Bath, Bath BA2 7AY, United Kingdom (as@maths.bath.ac.uk).

gives rise to efficient methods for the detection and computation of Hopf bifurcations. Finally, we consider the computation of the branch of rotating waves arising from the bifurcation.

The block diagonalisation of the Jacobian we consider is associated with the isotypic decomposition of the underlying Hilbert space [12], [1]. However, with this decomposition, there is always a problem in finding an appropriate basis for the block diagonalisation to occur in its simplest and most useful form. Healey [8] and Chang and Healey [3] essentially employ the known projection operators P_k onto the isotypic components and solve the equation

$$(P_k - I)\psi = 0$$

for the eigenfunctions ψ associated with the zero eigenvalue of the linear operator $P_k - I$. Murota and Ikeda [11] derive a transformation matrix that can be applied to any basis to give a basis that results in the block diagonal form of the Jacobian. However, in our case, these relatively complicated methods are not required since we are able to choose an appropriate basis as the eigenfunctions of a linear operator that arises naturally from the action of $O(2)$ on the Hilbert space.

The continuation package AUTO [5] will find rotating wave solutions of a restricted class of reaction-diffusion equations that arise from a Hopf bifurcation. However, in detecting these bifurcations, it deals with only the rather artificial case of fixing the velocity of the wave and allowing the period of the wave to vary.

The plan of the paper is as follows. In §2 we describe the block diagonalisation of the Jacobian and the simple way of finding a basis for the space that gives rise to this structure. In §3 we consider the Hopf bifurcation in more detail before describing methods for computing the bifurcating rotating waves in §4. In §5, we briefly consider the extension to the infinite-dimensional case and finally, we present a numerical example in §6.

2. Block diagonalisation. We consider time-dependent nonlinear problems of the form

$$\frac{du}{dt} + g(u, \lambda) = 0, \quad u \in X, \quad \lambda \in \mathbf{R}, \tag{2.1}$$

where X is an n-dimensional real Hilbert space equipped with an inner product $\langle \cdot, \cdot \rangle$ and $g : X \times \mathbf{R} \to X$ is a smooth nonlinear operator. We discuss the extension to the infinite-dimensional case in §6. We assume that g is equivariant with respect to an action of the group $O(2)$ on X, that is,

$$\gamma g(u, \lambda) = g(\gamma u, \lambda), \quad \forall \gamma \in O(2), \quad u \in X, \tag{2.2}$$

where $O(2)$ is the Lie group generated by rotations r_θ, $\theta \in \mathbf{R}$, and a reflection s, both of which satisfy the following relations for any α, $\beta \in \mathbf{R}$ (where $\mathbf{1}$ is the group identity element):

$$\begin{aligned} &r_{\alpha+2\pi} = r_\alpha, \quad r_{\beta+\alpha} = r_\beta r_\alpha = r_\alpha r_\beta, \\ &s^2 = r_0 = r_{2\pi} = \mathbf{1}, \quad s r_\alpha = r_{-\alpha} s. \end{aligned} \tag{2.3}$$

We also assume, without loss of generality [7, p. 31], that the inner product is $O(2)$-invariant, i.e.,

$$\langle \gamma x, \gamma y \rangle = \langle x, y \rangle \quad \forall x, y \in X, \ \gamma \in O(2). \tag{2.4}$$

For the sake of convenience, we assume the existence of a trivial steady-state solution path $(0, \lambda)$ so that

$$g(0, \lambda) = 0, \quad \lambda \in \mathbf{R}.$$

We are concerned with analysing the situation when a Hopf bifurcation occurs on this $O(2)$-symmetric trivial solution, with particular interest in the branch of rotating wave solutions that arises from such a bifurcation. In practice, Hopf bifurcations can often be found analytically in such situations. However, the theory also applies to problems with nontrivial $O(2)$-symmetric solutions such as are found in partial differential equation problems on a circular domain. Hopf bifurcations cannot be found analytically in such problems and so the numerical techniques we describe must be employed.

Rotating wave solutions of (2.1) are defined by

$$u(t) = r_{ct}x,$$

where $x \in X$ is independent of time and $c \in \mathbf{R}$ is the velocity of the wave. Substituting this form of solution into the original equation (2.1) leads to the "steady-state" equation

$$g(x, \lambda) + cAx = 0, \tag{2.5}$$

where A is the linear operator on X defined by

$$Ax := \frac{d}{d\theta} r_\theta x \Big|_{\theta=0}. \tag{2.6}$$

Because this operator A plays a central role in our theory, we now explore some of its properties.

LEMMA 2.1. (i) *For all* $\theta \in \mathbf{R}$,

$$\frac{dr_\theta}{d\theta} = r_\theta A, \quad r_\theta A = A r_\theta, \quad sA = -As. \tag{2.7}$$

(ii) *The linear operator* A^2 *is a symmetric linear operator on* X *with nonpositive eigenvalues.*

Proof. (i) These results are easily proved using the group relations (2.3).

(ii) Using the invariance property (2.4) of the inner product, we have that

$$\langle Ax, y\rangle = \frac{d}{d\theta}\langle r_\theta x, y\rangle \,|_{\theta=0} = \frac{d}{d\theta}\langle x, r_{-\theta} y\rangle \,|_{\theta=0} = -\langle x, Ay\rangle \tag{2.8}$$

giving $A^T = -A$, i.e., A is skew-symmetric. The stated properties of A^2 follow immediately from this fact. □

Let $\{\alpha_k, \ k = 0, 1, 2, \ldots, N\}$ be the distinct eigenvalues of A^2 and let the corresponding eigenspaces be

$$E_{\alpha_k} := \{\phi \in X : \ A^2\phi = \alpha_k \phi\}.$$

Elementary linear algebra gives

$$X = \sum_{k=0}^{N} \oplus E_{\alpha_k}.$$

We shall show later that this decomposition is very closely related to the isotypic decomposition of the Hilbert space and is the appropriate basis for the block diagonalisation of the Jacobian matrix.

The following results will frequently be used.

LEMMA 2.2. *The eigenspaces* E_{α_k}, $k = 0, 1, 2, \ldots, N$ *are invariant with respect to* A, $g_x(0, \lambda)$ *and* $O(2)$, *that is,*

$$A : E_{\alpha_k} \longrightarrow E_{\alpha_k}, \tag{2.9a}$$

$$g_x(0, \lambda) : E_{\alpha_k} \longrightarrow E_{\alpha_k}, \quad \lambda \in \mathbf{R}, \tag{2.9b}$$

$$\gamma : E_{\alpha_k} \longrightarrow E_{\alpha_k}, \quad \forall \gamma \in O(2). \tag{2.9c}$$

Proof. Property (2.9a) is easily verified. Differentiating the equivariance condition (2.2) for $\gamma = r_\theta$ with respect to θ and setting $\theta = 0$ gives

$$g_x(0, \lambda)A = Ag_x(0, \lambda), \quad \lambda \in \mathbf{R}. \tag{2.10}$$

Let $\phi \in E_{\alpha_k}$. Then

$$A^2 g_x(0, \lambda)\phi = g_x(0, \lambda)A^2\phi = \alpha_k g_x(0, \lambda)\phi,$$

which proves (2.9b). The relation

$$\gamma A^2 = A^2\gamma, \quad \gamma \in O(2),$$

which is derived from (2.7), is used to prove (2.9c) in a similar way. □

It is well known that the reflectional symmetry s can be used to decompose the space X into symmetric and antisymmetric subspaces as

$$X = X_s \oplus X_a,$$

where

$$X_s := \{x \in X : sx = x\}, \quad X_a := \{x \in X : sx = -x\}.$$

It follows from the last relation of (2.7) that $A : X_s \to X_a$ and $A : X_a \to X_s$, so both X_s and X_a are invariant under A^2. This enables the eigenspaces E_{α_k} to be decomposed as

$$E_{\alpha_k} = E^s_{\alpha_k} \oplus E^a_{\alpha_k}, \tag{2.11}$$

where

$$E^s_{\alpha_k} := E_{\alpha_k} \cap X_s, \quad E^a_{\alpha_k} := E_{\alpha_k} \cap X_a.$$

Note that $E^s_{\alpha_k}$ and $E^a_{\alpha_k}$ are also eigenspaces of A^2. It is easily verified that if $\alpha_k \neq 0$, then

$$E^a_{\alpha_k} = AE^s_{\alpha_k} \quad \text{and} \quad E^s_{\alpha_k} = AE^a_{\alpha_k} \tag{2.12}$$

so that if

$$E^s_{\alpha_k} = \text{span}\{\phi_1, \ldots, \phi_m\}, \tag{2.13a}$$

then

$$E^a_{\alpha_k} = \text{span}\{A\phi_1, \ldots, A\phi_m\}. \tag{2.13b}$$

The next result uses these eigenspaces to give the block diagonalisation of the Jacobian matrix $g_x(0, \lambda)$.

THEOREM 2.3. *If the basis of X is chosen to be the eigenvectors of A^2, then $g_x(0, \lambda)$ has the block diagonal structure*

$$g_x(0,\lambda) = \begin{bmatrix} B_0 & & & \\ & \cdot & & \\ & & \cdot & \\ & & & \cdot \\ & & & B_N \end{bmatrix}. \tag{2.14a}$$

In addition, for $\alpha_k \neq 0$, choosing (2.13) as bases of $E^s_{\alpha_k}$ and $E^a_{\alpha_k}$, leads to a further decomposition of the B_k block corresponding to E_{α_k} as

$$B_k = \begin{pmatrix} J_k & 0 \\ 0 & J_k \end{pmatrix}. \tag{2.14b}$$

Proof. The decomposition (2.14a) follows directly from (2.9b). The further two block decomposition (2.14b) also follows immediately from the splitting (2.13) and the fact that X_s and X_a are invariant with respect to $g_x(0, \lambda)$. It remains to prove that the two blocks are identical.

Let the two blocks of the Jacobian on the symmetric and antisymmetric subspaces of E_{α_k} be J^s and J^a, respectively. Then, relative to the basis (2.13a), which we assume is an orthogonal basis,

$$J^s_{i,j} = \frac{\langle \phi_i, g_x(0,\lambda)\phi_j \rangle}{\langle \phi_i, \phi_i \rangle}.$$

Similarly, relative to the basis (2.13b),

$$\begin{aligned} J^a_{i,j} &= \frac{\langle A\phi_i, g_x(0,\lambda)A\phi_j \rangle}{\langle A\phi_i, A\phi_i \rangle} \\ &= \frac{\langle A\phi_i, Ag_x(0,\lambda)\phi_j \rangle}{\langle A\phi_i, A\phi_i \rangle} \quad \text{using (2.10)} \\ &= \frac{\langle A^2\phi_i, g_x(0,\lambda)\phi_j \rangle}{\langle A^2\phi_i, \phi_i \rangle} \quad \text{using (2.8)} \\ &= \frac{\alpha_k \langle \phi_i, g_x(0,\lambda)\phi_j \rangle}{\alpha_k \langle \phi_i, \phi_i \rangle} \quad \text{since } \phi_i \in E_{\alpha_k} \\ &= J^s_{i,j}. \end{aligned}$$

This completes the proof. □

Clearly, the structure exhibited in the block decomposition of Theorem 2.3 is important for the numerical solution of the problem. However, before we consider the numerical implications, we establish the link between the eigenspaces E_{α_k} and the isotypic components of the Hilbert space.

A Hilbert space has an isotypic component associated with every (nonequivalent) irreducible representation of the group on the Hilbert space. In particular, the projection onto the isotypic component of a real Hilbert space associated with the irreducible representation τ_k of the compact Lie group Γ is

$$P_k = \frac{n_k}{d_k} \int_\Gamma \chi_k(\gamma)\gamma \, d\gamma,$$

where n_k is the dimension of τ_k, d_k is 1 for absolutely irreducible representations and $\chi_k(\gamma)$ is the trace of $\tau_k(\gamma)$. The irreducible representations of the group $O(2)$ are given by

(i) $\tau_0^+ : r_\theta = I,\ s = I,$

(ii) $\tau_0^- : r_\theta = I,\ s = -I,$

(iii) $\tau_k : r_\theta = \begin{bmatrix} \cos k\theta & \sin k\theta \\ -\sin k\theta & \cos k\theta \end{bmatrix}, \quad s = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad k \in \mathbf{Z}^+.$

We note that all of these irreducible representations are absolutely irreducible. The first step towards establishing the link is the following result.

THEOREM 2.4. *For all* $x \in X$,

(i) $A^2 P_k x = P_k A^2 x,\ k \in \mathbf{Z}^+,\ A^2 P_0^\pm x = P_0^\pm A^2 x,$

(ii) $A^2 P_0^+ x = 0,$

(iii) $A^2 P_0^- x = 0,$

(iv) $A^2 P_k x = -k^2 P_k x,\ k \in \mathbf{Z}^+.$

Proof. Result (i) follows immediately from the fact that r_θ and s commute with A^2, which follows from (2.7). We prove parts (ii) and (iii) together. Now

$$P_0^\pm = \frac{1}{4\pi}\int_0^{2\pi} r_\theta d\theta \pm \frac{1}{4\pi}\int_0^{2\pi} s r_\theta d\theta,$$

since $n_0^\pm = d_0^\pm = 1$, $\chi_0^\pm(r_\theta) = 1$, and $\chi_0^\pm(sr_\theta) = \pm 1$. Thus,

$$A^2 P_0^\pm = \frac{1}{4\pi}\int_0^{2\pi} A^2 r_\theta \pm A^2 s r_\theta d\theta$$

$$= \frac{1}{4\pi}\int_0^{2\pi} A^2 r_\theta \pm s A^2 r_\theta d\theta,$$

since A^2 commutes with s. From (2.7), we have

$$\frac{dr_\theta}{d\theta} = A r_\theta$$

and so

$$\frac{d^2 r_\theta}{d\theta^2} = A\frac{dr_\theta}{d\theta} = A^2 r_\theta.$$

Therefore,

$$A^2 P_0^\pm = \frac{1}{4\pi}\int_0^{2\pi} \left(\frac{d^2 r_\theta}{d\theta^2} \pm s\frac{d^2 r_\theta}{d\theta^2} \right) d\theta$$

$$= \frac{1}{4\pi}\left[\frac{dr_\theta}{d\theta} \pm s\frac{dr_\theta}{d\theta} \right]_0^{2\pi}$$

$$= 0$$

as required, since $\frac{dr_\theta}{d\theta}\,|_{\theta=2\pi} = \frac{dr_\theta}{d\theta}\,|_{\theta=0}$ $(= A)$.

Similarly, for part (iv) we have for $k \in \mathbf{Z}^+$ that

$$P_k = \frac{2}{4\pi}\left[\int_0^{2\pi} 2\cos k\theta\, r_\theta d\theta + \int_0^{2\pi} 0.s r_\theta d\theta \right]$$

$$= \frac{1}{\pi}\int_0^{2\pi} \cos k\theta\, r_\theta d\theta$$

and so

$$A^2 P_k = \frac{1}{\pi}\int_0^{2\pi} \cos k\theta \; A^2 r_\theta d\theta$$

$$= \frac{1}{\pi}\int_0^{2\pi} \cos k\theta \; \frac{d^2 r_\theta}{d\theta^2} d\theta$$

$$= -\frac{k^2}{\pi}\int_0^{2\pi} \cos k\theta \; r_\theta d\theta$$

$$= -k^2 P_k$$

using integration by parts twice. □

COROLLARY 2.5. *The only possible eigenvalues of* A^2 *are* 0 *and* $-k^2$ *for some* $k \in \mathbf{Z}^+$.

We can now establish the link between the eigenspaces of A^2 and the isotypic components of X in the following result.

THEOREM 2.6. *Let the isotypic components of* X *be* $V_0^{\pm} = P_0^{\pm} X$ *and* $V_k = P_k X$, $k \in \mathbf{Z}^+$. *Then*

(i) $E_0 = V_0^+ \oplus V_0^-$,

(ii) $E_{-k^2} = V_k$, $k \in \mathbf{Z}^+$.

Proof. (i) If $x \in V_0^{\pm}$, then $P_0^{\pm} x = x$ and so $A^2 x = 0$ by Theorem 2.4 (ii) and (iii), thus giving $V_0^+ \oplus V_0^- \subseteq E_0$. Conversely, suppose that $x \in E_0$, i.e., $A^2 x = 0$. Then for $k \in \mathbf{Z}^+$,

$$0 = P_k A^2 x = A^2 P_k x = -k^2 P_k x$$

by Theorem 2.4 (i), (iv), giving $P_k x = 0$ since $k \neq 0$. Thus, $x \in V_0^+ \oplus V_0^-$ since it has no component in any of the isotypic components V_k for $k \in \mathbf{Z}^+$. Therefore, $E_0 \subseteq V_0^+ \oplus V_0^-$. We thus conclude that $E_0 = V_0^+ \oplus V_0^-$.

(ii) As for part (i), if $x \in V_k$, then $P_k x = x$ and so $A^2 x = -k^2 x$ by Theorem 2.4 (iv), thus giving $V_k \subseteq E_{-k^2}$. Conversely, suppose that $x \in E_{-k^2}$. Then $A^2 x = -k^2 x$, so

$$-k^2 P_0^{\pm} x = P_0^{\pm} A^2 x = A^2 P_0^{\pm} x = 0$$

by Theorem 2.4 (i), (ii), and (iii), giving $P_0^{\pm} x = 0$ since $k \neq 0$. Similarly, for some $j \in \mathbf{Z}^+$,

$$-k^2 P_j x = P_j A^2 x = A^2 P_j x = -j^2 P_j x.$$

Thus, $P_j x = 0$ if $j \neq k$. Combining these results, we conclude that $x \in V_k$ and so $E_{-k^2} \subseteq V_k$. Thus, $V_k = E_{-k^2}$ as required.

Finally, by Theorem 3.3 of [1], the isotypic components V_k, $k \in \mathbf{Z}^+$ can be decomposed into symmetric and antisymmetric parts as

$$V_k = V_k^s \oplus V_k^a$$

with the property that $g_x(x, \lambda) : V_k^s \to V_k^s$ and $g_x(x, \lambda) : V_k^a \longrightarrow V_k^a$. This agrees with the decompositions of the B_k block into two blocks that act on the symmetric and antisymmetric subspaces. □

The numerical consequences of this block diagonalisation of the Jacobian are considered in the next section.

3. The Hopf bifurcation. Suppose that $(0, \lambda_0)$ is a Hopf bifurcation point at which

$$X_{\omega_0} := \text{Null}\{(g_x^0)^2 + \omega_0^2 I\} \neq 0, \quad \omega_0 \neq 0, \tag{3.1}$$

where $g_x^0 := g_x(0, \lambda_0)$. By [7, Chap. XVI, Prop. 1.4], generically X_{ω_0} is $O(2)$-simple. Since all the irreducible representations of $O(2)$ are absolutely irreducible, this implies that, generically

$$X_{\omega_0} = W_1 \oplus W_2, \tag{3.2}$$

where W_1 and W_2 are absolutely irreducible subspaces of X that are $O(2)$-isomorphic. We assume henceforth that (3.2) holds. Note that since W_1 and W_2 are isomorphic, their representations are equivalent. Hence, they are contained in the same isotypic component of the Hilbert space. As a result they are, by Theorem 2.6, also contained in the same eigenspace of A^2. However, we also assume that

$$X_{\omega_0} \cap E_0 = \{0\},$$

since the converse does not lead to (nontrivial) rotating wave solutions. This follows because $g : E_0 \times \mathbf{R} \to E_0$; thus any bifurcating periodic orbit remains in E_0 so that if it is a rotating wave solution, it must be a trivial one in the sense that $r_{ct}x = x$ for all time t. By Theorem 2.6, this excludes consideration of the isotypic components associated with the two one-dimensional irreducible representations and so (3.2) implies that dim $X_{\omega_0} = 4$, since W_1 and W_2 must have dimension 2, as they are irreducible.

When (3.1) and (3.2) hold, this clearly corresponds to a matrix block J_k in the decomposition (2.14b) having simple eigenvalues $\pm i\omega$ for some $k \in \mathbf{Z}^+$. It then follows from the choice of basis (2.13), which gives rise to the decomposition (2.14b), that

$$X_{\omega_0}^s := X_{\omega_0} \cap X_s = \text{span}\{\phi_1, \phi_2\},$$

where ϕ_1 and ϕ_2 satisfy

$$g_x^0 \phi_1 + \omega_0 \phi_2 = 0, \tag{3.3a}$$

$$g_x^0 \phi_2 - \omega_0 \phi_1 = 0. \tag{3.3b}$$

These equations are the real and imaginary parts of the complex eigenvalue equation. Since g_x^0 commutes with A (see (2.10)) and $A : X_s \to X_a$, it follows immediately from these equations that

$$X_{\omega_0}^a := X_{\omega_0} \cap X_a = \text{span}\{A\phi_1, A\phi_2\}.$$

This is not suprising in view of the choice of basis (2.13b) of $E_{\alpha_k}^a$. Clearly, $X_{\omega_0} = X_{\omega_0}^s \oplus X_{\omega_0}^a$.

In the next lemma, we highlight the relationship between the subspaces

$$X_{c_0}^{\pm} := \text{Null}\{g_x^0 \pm c_0 A\} \tag{3.4}$$

and the eigenspace X_{ω_0}.

LEMMA 3.1. *Suppose that $X_{\omega_0} \subset E_{-k_0^2}$ for some $k_0 \in \mathbf{Z}^+$. Then*

$$X_{\omega_0} = X_{c_0}^+ \oplus X_{c_0}^-,$$

where $c_0 = \omega_0/k_0$. *Moreover, there exists a* $\phi_0 \in X_{\omega_0}\backslash\{0\}$ *such that*

$$X^+_{c_0} = \text{span}\{\phi_0,\ A\phi_0\}, \tag{3.5a}$$

$$X^-_{c_0} = \text{span}\{S\phi_0,\ SA\phi_0\}. \tag{3.5b}$$

Proof. This result is based on the careful choice of ϕ_0. The appropriate function is

$$\phi_0 = k_0\phi_1 - A\phi_2, \tag{3.6}$$

where ϕ_1 and ϕ_2 are the eigenfunctions in $X^s_{\omega_0}$. Note that $\phi_1 \in X_s$ and $A\phi_2 \in X_a$ and so ϕ_0 is neither symmetric nor antisymmetric. Then,

$$\begin{aligned}(g^0_x + c_0A)\phi_0 &= (g^0_x + c_0A)(k_0\phi_1 - A\phi_2)\\ &= k_0g^0_x\phi_1 - g^0_xA\phi_2 + c_0k_0A\phi_1 - c_0A^2\phi_2\\ &= k_0g^0_x\phi_1 - Ag^0_x\phi_2 + c_0k_0A\phi_1 + c_0k_0^2\phi_2\\ &= -k_0\omega_0\phi_2 - \omega_0A\phi_1 + c_0k_0A\phi_1 + c_0k_0^2\phi_2\\ &= (c_0k_0 - \omega_0)(A\phi_1 + k_0\phi_2)\\ &= k_0^{-1}(c_0k_0 - \omega_0)A\phi_0,\end{aligned}$$

using the fact that g^0_x commutes with A (see (2.10)), the relation $A^2\phi_2 = -k_0^2\phi_2$, since $\phi_2 \in E_{-k_0^2}$ and the eigenvalue equations (3.3). Thus,

$$\phi_0 \in X^+_{c_0} \iff c_0 = \omega_0/k_0.$$

Given $\phi_0 \in X^+_{c_0}$, it follows immediately that $A\phi_0 \in X^+_{c_0}$ since A commutes with g^0_x and it is easily verified that ϕ_0 and $A\phi_0$ are linearly independent.

Finally, for any $\phi \in X^+_{c_0}$, $S\phi \in X^-_{c_0}$, since S commutes with g^0_x but anticommutes with A (see (2.7)). □

The block structure described in the previous section clearly has important numerical implications with regard to the detection and computation of the Hopf bifurcation points we are considering. Generally, in a discretised system of dimension N, the detection of a Hopf bifurcation requires the computation of *all* the eigenvalues of the nonsymmetric $N \times N$ Jacobian matrix, which requires $O(N^3)$ operations. For large N, this is often prohibitively expensive. However, in the situation we are considering, it is possible for any of the blocks J_k in the decomposition (2.14) to have imaginary eigenvalues. Hence, we need only compute the eigenvalues of the blocks J_k that are much smaller than the whole of the Jacobian matrix, thus reducing the computational expense considerably. The most efficient procedure for detecting a Hopf bifurcation is thus to construct each block J_k on the space $V_k \cap X_s$, where V_k is one of the isotypic components of the space X and apply standard methods for detecting Hopf bifurcations to this block. Also, each block depends only on the $O(2)$-symmetric solution and is independent of the other blocks. The computation of the eigenvalues of each block can therefore be performed in parallel, thus reducing the computational expense still further.

Once the Hopf bifurcation has been detected, standard methods can again be applied using the block J_k to compute the bifurcation point accurately [9].

4. The rotating waves. It follows immediately from the equivariance condition (2.2), that if (x, c, λ) is a rotating wave solution of (2.5), then so is $(r_\alpha x, c, \lambda)$ for all $\alpha \in [0, 2\pi)$,

giving rise to an *orbit* of solutions. To isolate a point on this orbit, a phase condition can be imposed to give the system

$$F(x, c, \lambda) := \begin{bmatrix} g(x, \lambda) + cAx \\ \langle \ell, x \rangle \end{bmatrix} = 0 \tag{4.1}$$

for some appropriate choice of $\ell \in X$ (see [2]). However, this extended system has a singular Jacobian at the Hopf bifurcation point and so is not the best choice when computing the bifurcating branch of rotating wave solutions. Adopting an idea in [4] we introduce the following extended system for the computation of rotating waves near the Hopf bifurcation point with $x = \epsilon\tilde{x}$

$$G(y, \epsilon) = 0, \quad G : Y \times \mathbf{R} \to Y,$$

$$G(y, \epsilon) := \begin{bmatrix} \epsilon^{-1} g(\epsilon\tilde{x}, \lambda) + cA\tilde{x} \\ \langle \ell_a, \tilde{x} \rangle \\ \langle \ell_s, \tilde{x} \rangle - 1 \end{bmatrix}, \quad \epsilon \neq 0, \tag{4.2a}$$

$$G(y, 0) := \begin{bmatrix} g_x(0, \lambda)\tilde{x} + cA\tilde{x} \\ \langle \ell_a, \tilde{x} \rangle \\ \langle \ell_s, \tilde{x} \rangle - 1 \end{bmatrix}, \quad \epsilon = 0, \tag{4.2b}$$

$$y = (\tilde{x}, c, \lambda) \in Y := X \times \mathbf{R}^2,$$

where $\ell_s \in X_s$ and $\ell_a \in X_a$. This system necessarily involves the velocity c. However, once the Hopf bifurcation has been detected, the initial velocity c_0 is known from Lemma 3.1.

For the next result, we require the left eigenvectors of $g_x^0 + c_0 A$, so we define

$$\text{Range}(g_x^0 + c_0 A) := \{x \in X, \ \psi_1 x = \psi_2 x = 0\}.$$

THEOREM 4.1. *Let $(x, \lambda) = (0, \lambda_0)$ be a Hopf bifurcation point at which $X_{\omega_0} \subset E_{-k_0^2}$ for some $k_0 \in \mathbf{Z}^+$ and let $y_0 = (\phi_0, c_0, \lambda_0)$, where ϕ_0 and c_0 are defined in Lemma* 3.1. *Choose $\ell_s \in X_s$ and $\ell_a \in X_a$ such that $G(y_0, 0) = 0$ and*

$$\langle \ell_a, A\phi_1 \rangle \neq 0, \tag{4.3}$$

where ϕ_1 is defined by (3.3). *If the nondegeneracy condition*

$$\begin{vmatrix} \langle \psi_1, A\phi_0 \rangle & \langle \psi_1, g_{x\lambda}^0 \phi_0 \rangle \\ \langle \psi_2, A\phi_0 \rangle & \langle \psi_2, g_{x\lambda}^0 \phi_0 \rangle \end{vmatrix} \neq 0 \tag{4.4}$$

holds, then $G_y^0 := G_y(y_0, 0)$ is nonsingular and hence, there exists a unique solution path $(y(\epsilon), \epsilon)$ of (4.2) *in $Y \times \mathbf{R}$.*

Proof. The Jacobian of the system (4.2) is given by

$$G_y(y, \epsilon) = \begin{bmatrix} g_x(\epsilon\tilde{x}, \lambda) + cA & A\tilde{x} & \frac{1}{\epsilon} g_\lambda(\epsilon\tilde{x}, \lambda) \\ \langle \ell_a, \cdot \rangle & 0 & 0 \\ \langle \ell_s, \cdot \rangle & 0 & 0 \end{bmatrix}.$$

Let $\Phi := (\alpha, \beta, \delta) \in Y$ satisfy

$$G_y^0 \Phi = 0. \tag{4.5}$$

Then the first equation of this system is

$$(g_x^0 + c_0 A)\alpha + \beta A\phi_0 + \delta g_{x\lambda}^0 \phi_0 = 0, \tag{4.6}$$

where $g_x^0 := g_x(0, \lambda_0)$, etc., since $\lim_{\epsilon \to 0} \frac{1}{\epsilon} g_\lambda(\epsilon\tilde{x}, \lambda) = g_{x\lambda}(0, \lambda)\tilde{x}$. Taking the inner product of this expression with ψ_1 and ψ_2 gives the two equations

$$\begin{bmatrix} \langle \psi_1, A\phi_0 \rangle & \langle \psi_1, g_{x\lambda}^0 \phi_0 \rangle \\ \langle \psi_2, A\phi_0 \rangle & \langle \psi_2, g_{x\lambda}^0 \phi_0 \rangle \end{bmatrix} \begin{bmatrix} \beta \\ \delta \end{bmatrix} = 0.$$

The nondegeneracy condition (4.4) then implies that $\beta = \delta = 0$. Thus, from (4.6) and (3.5a),

$$\alpha = \mu\phi_0 + \eta A\phi_0,$$

where $\mu, \eta \in \mathbf{R}$. Using the form of ϕ_0 given in (3.6), it is easily verified that $A\phi_0 = k_0(A\phi_1 + k_0\phi_2)$. Thus,

$$\alpha = k_0(\mu\phi_1 + \eta k_0 \phi_2) + (\eta k_0 A\phi_1 - \mu A\phi_2). \tag{4.7}$$

The first term of (4.7) is in X_s and the second term is in X_a, since $\phi_1, \phi_2 \in X_s$ and $A : X_s \to X_a$. The second and third equations of (4.5) are, therefore,

$$\begin{bmatrix} -\langle \ell_a, A\phi_2 \rangle & \langle \ell_a, k_0 A\phi_1 \rangle \\ \langle \ell_s, k_0\phi_1 \rangle & \langle \ell_s, k_0^2 \phi_2 \rangle \end{bmatrix} \begin{bmatrix} \mu \\ \eta \end{bmatrix} = 0.$$

Now $G(y_0, 0) = 0$ and this implies that $\langle \ell_a, \phi_0 \rangle = 0$ and $\langle \ell_s, \phi_0 \rangle = 1$, which in turn imply that $\langle \ell_a, A\phi_2 \rangle = 0$ and $\langle \ell_s, \phi_1 \rangle = 1$. Thus, the above matrix is nonsingular if and only if the nondegeneracy condition (4.3) holds, giving $\mu = \eta = 0$ and hence $\alpha = 0$ as well. Thus, $\Phi = 0$ and G_y^0 is nonsingular. The Implicit Function Theorem can then be applied to give the existence and uniqueness of the path of solutions $(y(\epsilon), \epsilon)$ of (4.2). □

Differentiating $G(y(\epsilon), \epsilon) = 0$ with respect to ϵ and evaluating at $\epsilon = 0$ gives

$$G_y^0 \dot{y}_0 + G_\epsilon^0 = 0, \tag{4.8}$$

where $G_\epsilon^0 := G_\epsilon(y_0, 0) = (\frac{1}{2} g_{xx}^0 \phi_0 \phi_0, 0, 0)^T$ and $\dot{y}_0 := \frac{dy(\epsilon)}{d\epsilon} |_{\epsilon=0}$. This equation can be solved for the tangent vector $\dot{y}_0$ to the path of rotating wave solutions, which can be used in continuation codes for starting out along the branch of solutions. Once the initial part of the branch has been followed, the rotating wave solutions could then be continued using the slightly smaller system (4.1) with $\ell = \ell_a$ if required.

It is possible to define a reflectional symmetry on the system (4.2). To see this, let

$$S \begin{bmatrix} \tilde{x} \\ c \\ \lambda \end{bmatrix} = \begin{bmatrix} s\tilde{x} \\ -c \\ \lambda \end{bmatrix}.$$

Then it is easily verified that

$$SG(y, \epsilon) = G(Sy, \epsilon)$$

using the invariance property (2.4) of the inner product. An immediate consequence of this property is that, for any value of ϵ, if $(\tilde{x}, c, \lambda)$ is a solution of (4.2), then $(s\tilde{x}, -c, \lambda)$ is also a solution. This new solution simply consists of a reflected wave rotating in the opposite direction.

Now we consider the symmetry properties of the rotating wave solutions. They clearly do not have the reflectional symmetry s. However, the isotypic component V_k is contained in the fixed point space X^{Z_k}, where Z_k is the cyclic group generated by $r_{2\pi/k}$. Thus, if $X_{\omega_0} \subset V_k$ for some k, then Theorem 4.1 holds restricting attention to the space $Y_k := X^{Z_k} \times \mathbf{R}^2$, resulting in a branch of solutions that are invariant under $r_{2\pi/k}$.

Finally, we show that the nondegeneracy condition (4.4) is essentially an eigenvalue crossing condition. For this, we assume that

$$X_{c_0}^{+} \cap \text{Range}(g_x^0 + c_0 A) = \{0\}. \tag{4.9}$$

Since ϕ_0 and $A\phi_0$ are basis vectors for $X_{c_0}^{+}$, this condition is equivalent to

$$\begin{vmatrix} \langle \psi_1, \phi_0 \rangle & \langle \psi_1, A\phi_0 \rangle \\ \langle \psi_2, \phi_0 \rangle & \langle \psi_2, A\phi_0 \rangle \end{vmatrix} \neq 0. \tag{4.10}$$

THEOREM 4.2. *Let $\sigma(\lambda) \pm i\omega(\lambda)$ be eigenvalues of $g_x(0, \lambda)$ satisfying $\sigma_0 := \sigma(\lambda_0) = 0$. If (4.9) holds, then the nondegeneracy condition (4.4) is satisfied if and only if $\dot{\sigma}_0 := \frac{d\sigma(\lambda)}{d\lambda}|_{\lambda=0} \neq 0$.*

Proof. Equating the real and imaginary parts of the eigenvalue equation gives

$$g_x(0, \lambda)\phi_1(\lambda) = \sigma(\lambda)\phi_1(\lambda) - \omega(\lambda)\phi_2(\lambda),$$

$$g_x(0, \lambda)\phi_2(\lambda) = \sigma(\lambda)\phi_2(\lambda) + \omega(\lambda)\phi_1(\lambda).$$

Differentiating these relations with respect to λ and setting $\lambda = 0$ gives

$$g_{x\lambda}^0\phi_1 + g_x^0\dot{\phi}_1 = \dot{\sigma}_0\phi_1 - \dot{\omega}_0\phi_2 - \omega_0\dot{\phi}_2, \tag{4.11a}$$

$$g_{x\lambda}^0\phi_2 + g_x^0\dot{\phi}_2 = \dot{\sigma}_0\phi_2 + \dot{\omega}_0\phi_1 + \omega_0\dot{\phi}_1. \tag{4.11b}$$

In order to introduce $\phi_0 = k_0\phi_1 - A\phi_2$, we multiply the first equation by k_0, operate on the second with A, and then add the two giving

$$g_{x\lambda}^0\phi_0 + g_x^0\dot{\phi}_0 = \dot{\sigma}_0\phi_0 - \frac{\dot{\omega}_0}{k_0}A\phi_0 - \frac{\omega_0}{k_0}A\dot{\phi}_0,$$

since $g_{x\lambda}^0$ commutes with A and $A\phi_0 = k_0(A\phi_1 + k_0\phi_2)$. Now $\omega_0 = c_0k_0$ by Lemma 3.1, so

$$g_{x\lambda}^0 + (g_x^0 + c_0A)\dot{\phi}_0 = \dot{\sigma}_0\phi_0 - \frac{\dot{\omega}_0}{k_0}A\phi_0.$$

Taking the inner product of this expression with ψ_j, $j = 1, 2$ gives

$$\langle \psi_j, \phi_0 \rangle \dot{\sigma}_0 - \frac{\dot{\omega}_0}{k}\langle \psi_j, A\phi_0 \rangle = \langle \psi_j, g_{x\lambda}^0\phi_0 \rangle,$$

which is two equations to be solved for $\dot{\sigma}_0$ and $\dot{\omega}_0$ giving

$$\dot{\sigma}_0 = \begin{vmatrix} \langle \psi_1, A\phi_0 \rangle & \langle \psi_1, g_{x\lambda}^0\phi_0 \rangle \\ \langle \psi_2, A\phi_0 \rangle & \langle \psi_2, g_{x\lambda}^0\phi_0 \rangle \end{vmatrix} \Big/ \begin{vmatrix} \langle \psi_1, \phi_0 \rangle & \langle \psi_1, A\phi_0 \rangle \\ \langle \psi_2, \phi_0 \rangle & \langle \psi_2, A\phi_0 \rangle \end{vmatrix}.$$

Because (4.9) holds, (4.10) also holds so that the denominator of this expression is nonzero. Thus, $\dot{\sigma}_0 \neq 0$ if and only if the numerator is nonzero, which is precisely condition (4.4). □

The condition $\dot{\sigma}_0 \neq 0$ in Theorem 4.2 is, of course, the familiar condition that the complex conjugate eigenvalues must actually *cross* the imaginary axis as λ passes through the Hopf bifurcation point at λ_0 so that a stable trivial solution loses stability at a Hopf bifurcation point.

5. Extension to infinite dimensions. In this section, we generalize our results to an infinite-dimensional case (cf. §6 of [2]). This enables us to deal with partial differential equations directly.

We assume that g is a C^2 mapping from $\tilde{X} \times \mathbf{R}$ into X, where $\tilde{X}$ and X are both real Hilbert spaces with $\tilde{X} \subset X$. We also assume that g_x is a Fredholm operator of index zero. We suppose that there is an action of $O(2)$ on X that is strongly continuous and that $\tilde{X}$ is an $O(2)$-invariant subspace of X, i.e., $\gamma x \in \tilde{X}, \forall \gamma \in O(2),\ x \in \tilde{X}$. The equivariance condition (2.2) then holds. We also assume that if $x \in \tilde{X}$, then $Ax \in X$, so that (2.5) defines a mapping from $\tilde{X} \times \mathbf{R}$ into X. Finally, we assume that the decomposition of X into eigenspaces of A^2 takes the form of an infinite sum of the finite-dimensional subspaces E_{-k^2}, $k = 0, 1, 2, \ldots$. This decomposition then gives a natural decomposition of $\tilde{X}$ also. All the results of the previous sections then hold for the infinite-dimensional case with minor changes.

6. Numerical results. We now apply the theory to a practical example. We consider the coupled pair of reaction-diffusion equations known as the Sel'kov model [6]:

$$u_t = d_1 u_{xx} + 1 - uv^2,$$
$$v_t = d_2 v_{xx} + p(uv^2 - v),$$

which we write in more general form as

$$U_t = G(U, d_1) = DU_{xx} + F(U),$$

where $U = \binom{u}{v}$, D is the diagonal matrix of diffusion coefficients, and $F(U)$ is the nonlinear reaction terms. We choose the diffusion coefficient d_1 as our bifurcation parameter. Let H^m be the Hilbert space of 2π-periodic functions whose derivatives up to and including the mth are square integrable, with inner product

$$\langle u, v \rangle_{H^m} := \frac{1}{2\pi} \int_0^{2\pi} u^{(m)}(x) v^{(m)}(x) + u(x)v(x)\, dx.$$

Then $G : H^2 \times H^2 \times \mathbf{R} \longrightarrow H^0 \times H^0$. We define an action of $O(2)$ on H^0 by

$$r_\alpha u(x) = u(x + \alpha), \quad \alpha \in [0, 2\pi), \quad su(x) = u(-x). \tag{6.1}$$

For this action, the inner product on H^m is $O(2)$-invariant and H^2 is an invariant subspace of H^0. Also G is equivariant with repect to the diagonal action on $H^0 \times H^0$ defined by

$$\gamma \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \gamma u \\ \gamma v \end{pmatrix} \quad \forall \gamma \in O(2).$$

In this example, the linear operator A has the form

$$A \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_x \\ v_x \end{pmatrix} \tag{6.2}$$

and the eigenspaces of A^2 are

$$E_{-k^2} = (\text{span}\{\cos kx,\ \sin kx\})^2,\quad k \neq 0,$$

$$E_0 = (\text{span}\{1\})^2.$$

Clearly, the eigenspaces E_{-k^2} are four-dimensional for $k \neq 0$, giving rise to 4×4 blocks B_k in the decomposition of the Jacobian given by (2.14a). These blocks then decompose into 2×2 blocks (as in (2.14b)) on the symmetric and antisymmetric parts of the eigenspace E_{-k^2}. Using the definition of the reflectional symmetry s in (6.1), the symmetric subspace involves only the cosine functions, whereas the antisymmetric subspace involves only the sine functions. Thus, we know precisely the spaces that give rise to the block decomposition of the Jacobian.

As an example, we choose $d_2 = 4$, $p = 6$ and there is then a Hopf bifurcation from the trivial solution $u = v = 1$ at $d_1 = 1$ associated with the E_{-1} eigenspace for which $\omega_0 = c_0 = \sqrt{8}$. The symmetric eigenfunctions ϕ_1 and ϕ_2 at this point are given by

$$\phi_1 = \begin{pmatrix} (1+\sqrt{2})\cos x \\ 3\cos x \end{pmatrix}, \quad \phi_2 = \begin{pmatrix} (\sqrt{2}-1)\cos x \\ 3\cos x \end{pmatrix}.$$

With the linear operator A defined by (6.2), the corresponding antisymmetric eigenfunctions $A\phi_1$ and $A\phi_2$ are obtained from ϕ_1 and ϕ_2 by replacing $\cos x$ with $-\sin x$. The standing waves which bifurcate at this point can be computed by restricting attention to the subspace of symmetric functions (involving only the cosine functions) on which the Hopf bifurcation is standard. The usual methods can then be employed for switching onto the branch of standing waves. For computing the travelling waves, however, we require the function ϕ_0 defined by (3.6), which, for this example, is given by

$$\phi_0 = \begin{pmatrix} (1+\sqrt{2})\cos x + (\sqrt{2}-1)\sin x \\ 3(\cos x + \sin x) \end{pmatrix}.$$

By Theorem 4.1, we must choose the fixed vectors ℓ_s and ℓ_a so that they satisfy certain conditions. All these conditions are satisfied with

$$\ell_s = \begin{pmatrix} 0 \\ \frac{1}{3}\cos x \end{pmatrix}, \quad \ell_a = \begin{pmatrix} \sin x \\ \frac{1-\sqrt{2}}{3}\sin x \end{pmatrix}.$$

The system (4.2) can then be used to compute the branch of travelling waves.

The standing waves and travelling waves that bifurcate from the Hopf bifurcation point are shown in Fig. 1. The results are shown in terms of $\|U\|_2$, where U is defined by

$$U = \begin{pmatrix} u-1 \\ v-1 \end{pmatrix}$$

so that the trivial solution is $U = 0$. Figure 2 shows the variation in the velocity of the travelling waves along the branch together with the frequency of the standing waves.

Finally, we remark that the type of Hopf bifurcation we are considering will not occur generically in a single equation because the eigenspaces E_{-k^2} are then only two-dimensional. This is not sufficient to satisfy the generic condition (3.2), which requires a four-dimensional eigenspace of g_x^0 contained in E_{-k^2}. However, bifurcations to rotating wave solutions from nontrivial steady states are possible in this case, as in the Kuramoto–Sivashinsky equation (see [2]).

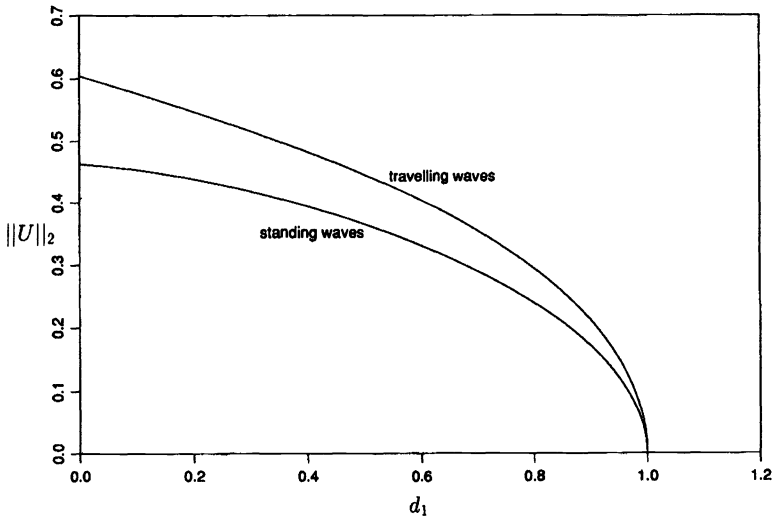


FIG. 1.

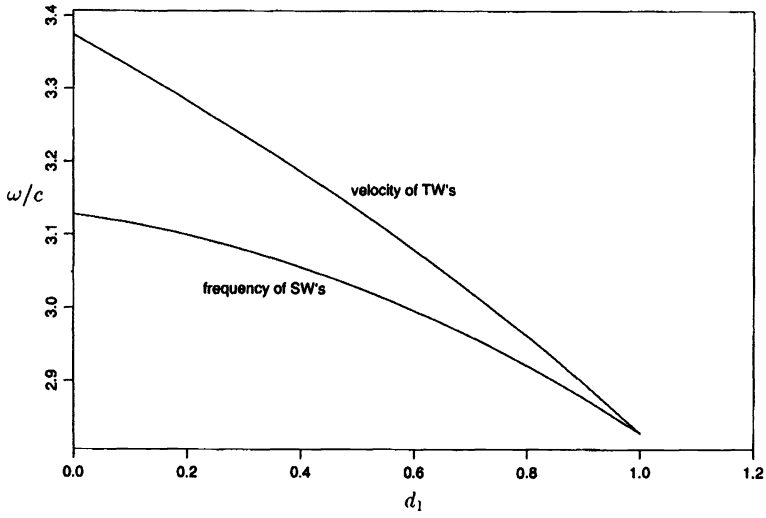


FIG. 2.

REFERENCES

[1] P. ASTON, *Analysis and computation of symmetry-breaking bifurcation and scaling laws using group theoretic methods*, SIAM J. Math. Anal., 22 (1991), pp. 181–212.

[2] P. ASTON, A. SPENCE, AND W. WU, *Bifurcation to rotating waves in equations with $O(2)$-symmetry*, SIAM J. Appl. Math., 52 (1992), pp. 792–809.

[3] P. CHANG AND T. HEALEY, *Computation of symmetry modes and exact reduction in nonlinear structural analysis*, Comput. & Structures, 28 (1988), pp. 135–142.

[4] D. DECKER AND H. KELLER, *Multiple limit point bifurcation*, J. Math. Anal. Appl., 75 (1980), pp. 417–430.

[5] E. DOEDEL, *Auto: A program for the automatic bifurcation analysis of autonomous systems*, Congr. Numer., 30 (1981), pp. 265–284.

[6] J. EILBECK AND J. FURTER, *Understanding steady-state bifurcation diagrams for a model reaction-diffusion system*, in Continuation and Bifurcations: Numerical Techniques and Applications, D. Roose, B. D. Dier, and A. Spence, eds., Kluwer, the Netherlands, 1990, pp. 25–41.

[7] M. GOLUBITSKY, I. STEWART, AND D. SCHAEFFER, *Singularities and Groups in Bifurcation Theory* II, Appl. Math. Sci. Series, vol. 69, Springer-Verlag, New York, 1988.

[8] T. HEALEY, *A group theoretic approach to computational bifurcation problems with symmetry*, Comput. Methods Appl. Mech. Engrg., 67 (1988), pp. 257–295.

[9] H. KELLER AND A. JEPSON, *Steady state and periodic solution paths: Their bifurcations and computations*, in Numerical Methods for Bifurcation Problems, T. Küpper, H. Mittelmann, and H.Weber, eds., Birkhäuser, Basel, 1984, pp. 219–246.

[10] M. KRUPA, *Bifurcations of relative equilibria*, SIAM J. Math. Anal., 21 (1990), pp. 1453–1486.

[11] K. MUROTA AND K. IKEDA, *Computational use of group theory in bifurcation analysis of symmetric structures*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 273–297.

[12] B. WERNER, *Eigenvalue problems with the symmetry of a group and bifurcations*, in Continuation and Bifurcations: Numerical Techniques and Applications, D. Roose, B. D. Dier, and A. Spence, eds., Kluwer, the Netherlands, 1990, pp. 71–88.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 511–527, May 1994

ERROR-MINIMIZING KRYLOV SUBSPACE METHODS*

RÜDIGER WEISS†

Abstract. Iterative methods for the solution of linear systems are usually controlled by the observation of the norm of the residual. In reality, the error should be controlled, but the error is not available. The residuals and the errors are connected by the condition number of the system matrix. If the system is well conditioned, the decrease of the errors is closely connected to the decrease of the residuals. For these cases, Krylov subspace methods that minimize the residuals in the Euclidean norm or in the energy norm are powerful solution techniques. If the system is ill conditioned, the residuals can decrease while the errors increase. For these systems, arising even from very simple and commonly used differential equations, iterative methods that minimize the residuals may require a large number of iterations to reduce the errors. The user may be misled to stop the iteration too early by small residuals. Two families of error-minimizing Krylov subspace methods are proposed to overcome these difficulties. Each of them is suited for different problem types.

Principles for the design of generalized cg methods that minimize the error are derived from the geometric convergence behavior of generalized cg methods. These methods use the transposed system matrix multiplied by the system matrix as the iteration matrix. By this technique a fast convergence is achieved for matrices with clustered singular values and scattered eigenvalues.

A class of Krylov subspace methods minimizing the error by using the simple transposed matrix as the iteration matrix is proposed. Various realization possibilities are inherent in these generalized minimum error methods. The methods are analyzed theoretically. Common and related properties with generalized conjugate gradient methods are presented. These techniques should be preferred if the eigenvalues are more clustered than the singular values. The first promising tests for one distinct method are presented.

Key words. conjugate gradients, convergence, linear systems, error-minimizing methods, Krylov methods

AMS subject classifications. 65F10, 65F50, 40A05

1. Background.

The purpose of this paper is to present error-minimizing Krylov subspace methods for the solution of the linear system

$$Ax = b. \tag{1}$$

The matrix A is a real, square matrix of dimension n, i.e., $A \in \mathbb{R}^{n\times n}$, and $x, b \in \mathbb{R}^n$. In general, the matrix A is nonsymmetric and nonpositive definite. Let us assume A to be nonsingular.

We use the following notation for norms: Let Z be a symmetric, positive definite matrix; then the norm $||y||_Z$ of any vector $y \in \mathbb{R}^n$ is defined by $||y||_Z = \sqrt{y^T Z y}$. If Z is nonsymmetric and nonpositive definite, then $||y||_Z^2$ is mnemonic abbreviation for $y^T Z y$. $||y||_I$ is the Euclidean norm $||y||$.

Let $K_k(B, y) = \text{span}(y, By, \ldots, B^k y)$ be the Krylov space spanned by the matrix $B \in \mathbb{R}^{n\times n}$ and the vector $y \in \mathbb{R}^n$.

For any iterative method for the solution of (1) the residuals $r_k = Ax_k - b$ and the errors $e_k = x_k - x$ are connected by

$$r_k = Ae_k. \tag{2}$$

As a result of $||r_k|| = ||Ae_k|| \le ||A|| \cdot ||e_k||$ and $||e_k|| = ||A^{-1}r_k|| \le ||A^{-1}|| \cdot ||r_k||$, the following inequalities are valid:

$$\frac{1}{\kappa}\frac{||r_k||}{||r_0||} \le \frac{||e_k||}{||e_0||} \le \kappa\frac{||r_k||}{||r_0||} \le \kappa^2\frac{||e_k||}{||e_0||}, \tag{3}$$

*Received by the editors September 24, 1992; accepted for publication (in revised form) March 26, 1993.

†Numerikforschung für Supercomputer, Rechenzentrum der Universität Karlsruhe, Postfach 6980, 76128 Karlsruhe, Germany (weiss@rz.uni-karlsruhe.de).

where $\kappa = ||A|| \cdot ||A^{-1}||$ is the condition number of A. If κ is near to one, the norm of the relative residuals is strongly connected to the norm of the relative errors. In other words, if the residuals decrease sufficiently the errors will be reduced as well. For a large condition number the residuals can decrease while the norm of the errors either remains the same or even increases.

The next example shows that a separation of the norm of the errors from the norm of the residuals happens even for very simple and commonly used systems.

Example 1. Solve the following linear system resulting from the discretization of a one-dimensional Laplace equation:

$$\begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \cdot x = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}.$$

The matrix has the eigenvalues $\lambda_k = 2 \cdot (1 - \cos \frac{k\pi}{n+1})$, where n is the dimension of the system and $1 \leq k \leq n$. Because the matrix is symmetric, the condition number is

$$\kappa = \frac{|\lambda_{\max}|}{|\lambda_{\min}|} = \frac{1 + \cos \frac{\pi}{n+1}}{1 - \cos \frac{\pi}{n+1}}.$$

The dimension of the system is 1000, resulting in a large condition number of approximately $4 \cdot 10^5$. The system was solved by the classical conjugate gradient (cg) method [8] and the GMRES method [11]. The starting guess is $x_0 = 0$. Figure 1 shows that the residuals decrease while the errors do not decrease for either method until 500 matrix-vector multiplications have been performed.

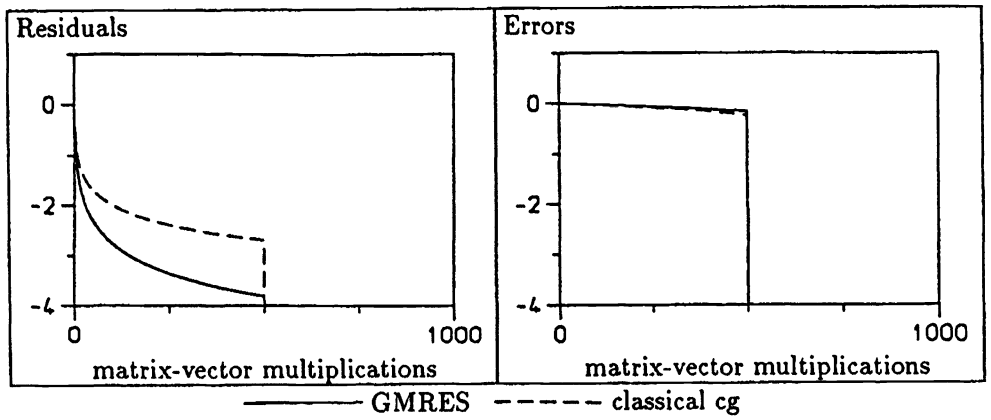


FIG. 1. *Norm of the relative residuals and errors of* GMRES *and classical cg (logarithmic scale).*

In the following sections we propose two remedies in order to obtain error-minimizing methods. The first technique is based on the iteration matrix $A^T A$ and generalized cg methods. The second technique uses the iteration matrix A^T and is a generalized Krylov subspace method.

2. Generalized conjugate gradient methods.

DEFINITION 1. *Let x_0 be any initial guess for the solution of the system $Ax = b$, $r_0 = Ax_0 - b$ the starting residual. The following recurrence is called a generalized cg method. Choose a preconditioning matrix P and calculate for $k \geq 1$ the residuals r_k and approximations x_k so that*

$$x_k \in x_0 + K_{k-1}(PA, Pr_0), \text{ with} \tag{4}$$

$$r_k^T Z r_{k-i} = 0 \tag{5}$$

for $i = 1, \ldots, \sigma_k$, where Z is an auxiliary, nonsingular matrix.

The method is called exact *if $\sigma_k = k$,* restarted *if $\sigma_k = (k-1) \bmod \sigma_{\text{res}} + 1$ with σ_{res} fixed,* truncated *if $\sigma_k = \min(k, \sigma_{\max})$ with $\sigma_{\max}$ fixed, and* combined *if the truncated method is restarted.*

From Definition 1 it follows directly for the residuals r_k and the errors $e_k = x_k - x$ that

$$r_k = \tilde{\Pi}_k(AP)r_0 = \sum_{i=1}^{k} \nu_{i,k}(AP)^i r_0 + r_0, \tag{6}$$

$$e_k = \tilde{\Pi}_k(PA)e_0 = \sum_{i=1}^{k} \nu_{i,k}(PA)^i e_0 + e_0, \tag{7}$$

where $\tilde{\Pi}_k$ is a polynomial of degree k with constant coefficient 1, i.e., $\tilde{\Pi}_k(0) = 1$.

If $Z = \bar{Z}AP$ and $\bar{Z}$ is symmetric, positive definite, then for exact and restarted methods (5) is equivalent to

$$\|r_k\|_{\bar{Z}} = \min_{\mu_{1,k},\ldots,\mu_{\sigma_k,k}} \left\| \sum_{i=1}^{\sigma_k} \mu_{i,k}(AP)^i r_{k-\sigma_k} + r_{k-\sigma_k} \right\|_{\bar{Z}}. \tag{8}$$

In this case the methods are called conjugate residual type methods or minimum residual methods.

If Z is symmetric, positive definite, then for exact and restarted methods (5) is equivalent to

$$\|\bar{r}_k\|_Z = \min_{\alpha_{1,k},\ldots,\alpha_{\sigma_k,k}} \left\| (AP)^{\sigma_k} r_{k-\sigma_k} + \sum_{i=1}^{\sigma_k} \alpha_{i,k}(AP)^{i-1} r_{k-\sigma_k} \right\|_Z, \tag{9}$$

where

$$\bar{r}_k = \frac{1}{\nu_{k,k}} r_k \tag{10}$$

is the pseudoresidual; see [12] and [14]. In this case the methods are called pseudoresidual methods.

The methods break down if division by zero occurs for the calculation of r_k. A breakdown may be curable, by modifying the algorithm [1], or incurable. We assume in the following that the algorithm does not break down.

LEMMA 2. *For any exact, generalized cg method*

$$r_k^T Z P^{-1} A^{-1} r_k = r_k^T Z P^{-1} A^{-1} \Pi_k(AP) r_0 \tag{11}$$

is satisfied for all matrix polynomials $\Pi_k(AP) = \sum_{i=1}^{k} \theta_i (AP)^i + I$ *(i.e.,* $\theta_1, \ldots, \theta_k$ *are arbitrary). This is especially true of*

$$r_k^T Z P^{-1} A^{-1} r_k = r_k^T Z P^{-1} A^{-1} r_j \,, \tag{12}$$

$$e_k^T A^T Z P^{-1} e_k = e_k^T A^T Z P^{-1} e_j \tag{13}$$

for $j = 0, \ldots, k$.

Proof. See [14]. □

The next investigations show some interesting facts concerning the geometric location of the residuals and the errors.

THEOREM 3. *The residuals* r_k *and the errors* e_k *of exact, generalized cg methods satisfy the following equations*:

$$\left\| r_k - \frac{\tilde{r}_j}{2} \right\|^2_{ZP^{-1}A^{-1}} = \frac{\|\tilde{r}_j\|^2_{ZP^{-1}A^{-1}}}{4}, \tag{14}$$

and

$$\left\| e_k - \frac{\tilde{e}_j}{2} \right\|^2_{A^T Z P^1} = \frac{\|\tilde{e}_j\|^2_{A^T Z P^{-1}}}{4} \tag{15}$$

for $j = 0, \ldots, k$ *with*

$$\tilde{r}_j = 2 \left(Z P^{-1} A^{-1} + (Z P^{-1} A^{-1})^T \right)^{-1} Z P^{-1} A^{-1} r_j, \tag{16}$$

$$\tilde{e}_j = 2 \left(Z P^{-1} + (Z P^{-1} A^{-1})^T A \right)^{-1} Z P^{-1} e_j. \tag{17}$$

In particular if $A^T Z P^{-1}$ *is symmetric, then*

$$\tilde{r}_j = r_j, \tag{18}$$

$$\tilde{e}_j = e_j. \tag{19}$$

Proof. By (12),

$$r_k^T Z P^{-1} A^{-1} (r_k - r_j) = 0.$$

Therefore,

$$r_k^T Z P^{-1} A^{-1} r_k - r_k^T Z P^{-1} A^{-1} r_j = 0.$$

From the definition (16) for $\tilde{r}_j$ follows

$$r_k^T Z P^{-1} A^{-1} r_k - \frac{1}{2} r_k^T Z P^{-1} A^{-1} A P Z^{-1} \left(Z P^{-1} A^{-1} + (Z P^{-1} A^{-1})^T \right) \tilde{r}_j = 0$$

and

$$r_k^T Z P^{-1} A^{-1} r_k \; - \; \frac{1}{2} r_k^T \left(Z P^{-1} A^{-1} + (Z P^{-1} A^{-1})^T \right) \tilde{r}_j + \frac{\|\tilde{r}_j\|^2_{ZP^{-1}A^{-1}}}{4} = \frac{\|\tilde{r}_j\|^2_{ZP^{-1}A^{-1}}}{4},$$

which is equivalent to

$$\left\| r_k - \frac{\tilde{r}_j}{2} \right\|^2_{ZP^{-1}A^{-1}} = \frac{\|\tilde{r}_j\|^2_{ZP^{-1}A^{-1}}}{4}.$$

Equation (15) follows by $r_j = Ae_j$. □

Equations (14) and (15) are quadratic forms and describe geometrical figures.

If $A^T ZP^{-1}$ is definite, then the residual r_k and the error e_k lie on hyperellipsoids. The lengths of the semiaxes of the hyperellipsoid for the residual are precisely the singular values of the matrix APZ^{-1} multiplied by $\|\tilde{r}_j\|_{ZP^{-1}A^{-1}}$. The lengths of the semiaxes of the hyperellipsoid for the error are precisely the singular values of the matrix $PZ^{-1}A^{-T}$ multiplied by $\|\tilde{e}_j\|_{A^T ZP^{-1}}$; see Fig. 2 for $j = 0$.

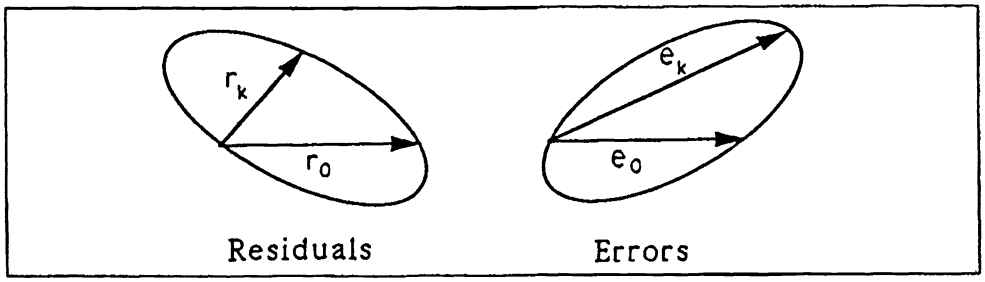


FIG. 2. *Residuals and errors if $A^T ZP^{-1}$ is definite.*

If $Z = AP$, then the residual r_k lies on an n-dimensional sphere; see Fig. 3 for $j = 0$. The norm of the residual is monotonically decreasing.

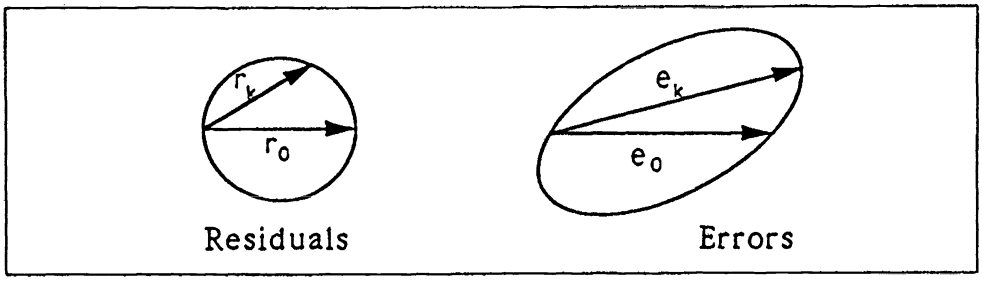


FIG. 3. *Residuals and errors if $Z = AP$.*

If $P = A^T Z$, then the error e_k lies on an n-dimensional sphere; see Fig. 4 for $j = 0$. The norm of the error is monotonically decreasing.

If $A^T ZP^{-1}$ is indefinite, then the geometric figures are not closed and in general the method does not converge; see Fig. 5 for $j = 0$.

The qualitative convergence behavior follows from Theorem 3 and the geometric interpretation. The speed of convergence depends on the eigenvalue distribution of the matrices AP and PA, respectively, which follows from the next lemma. Recall that the symmetric part of a matrix B is $\frac{1}{2}(B + B^T)$ and the skew-symmetric part is $\frac{1}{2}(B - B^T)$.

LEMMA 4. *If APZ^{-1} is positive real, i.e., the symmetric part of APZ^{-1} is positive definite,*

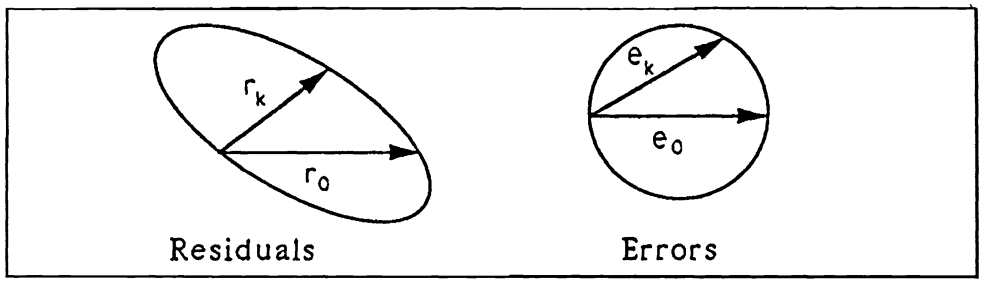


FIG. 4. *Residuals and errors if* $P = A^T Z$.

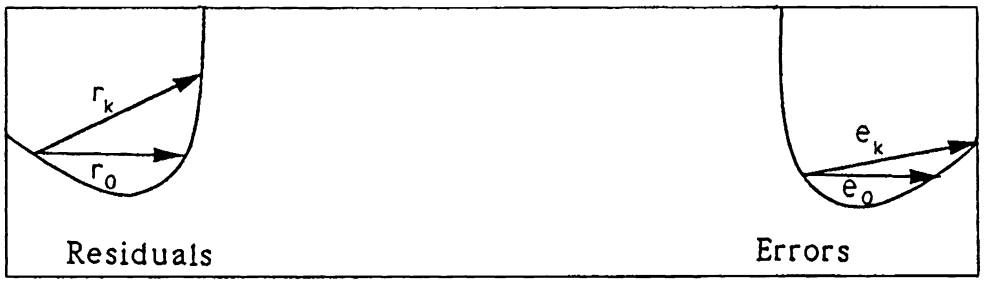


FIG. 5. *Residuals and errors if* $A^T Z P^{-1}$ *is indefinite.*

then

$$\|r_k\|_{ZP^{-1}A^{-1}} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2}} \min_{\Pi_k} \|\Pi_k(AP)r_0\|_{ZP^{-1}A^{-1}}, \tag{20}$$

$$\|e_k\|_{A^T ZP^{-1}} \leq \sqrt{1 + \frac{\rho^2(R)}{\mu_m^2}} \min_{\Pi_k} \|\Pi_k(PA)e_0\|_{A^T ZP^{-1}} \tag{21}$$

holds for exact generalized cg methods. Π_k *is a polynomial of degree* k *with* $\Pi_k(0) = 1$. $\rho(R)$ *is the spectral radius of the skew-symmetric part* R *of* $ZP^{-1}A^{-1}$. μ_m *is the minimum eigenvalue of* M, *the symmetric part of* $ZP^{-1}A^{-1}$. *For residual-minimizing methods* ($Z = AP$)

$$\|r_k\| = \min_{\beta_1,\ldots,\beta_k} \left\| \sum_{i=1}^{k} \beta_i (AP)^i r_0 + r_0 \right\|. \tag{22}$$

For error-minimizing methods ($P = A^T Z$)

$$\begin{aligned} \|e_k\| &= \min_{\beta_1,\ldots,\beta_k} \left\| \sum_{i=1}^{k} \beta_i (PA)^i e_0 + e_0 \right\| \\ &= \min_{\beta_1,\ldots,\beta_k} \left\| \sum_{i=1}^{k} \beta_i (A^T ZA)^i e_0 + e_0 \right\| \end{aligned} \tag{23}$$

is valid.

Proof. See [14]. □

As

$$x^T APZ^{-1}x = x^T Z^{-T} P^T A^T x = y^T A^T ZP^{-1} y$$

with $y = PZ^{-1}x$ for all x, APZ^{-1} is positive real if and only if $A^T ZP^{-1}$ is positive real. Therefore Lemma 4, in fact, covers all converging cases following from Theorem 3 and the geometric interpretation. As a result of Theorem 3 and Lemma 4 the methods can be classified. In Table 1 the different conditions for the minimized items are collected.

TABLE 1
Various generalized cg methods.

Condition	Minimized Item	Method
$ZP^{-1} = I$ and		
A sym., pos. def.	$\|\|r_k\|\|_{A^{-1}}$	minimum energy norm
$ZP^{-1}A^{-1}$ sym., pos. def.	$\|\|r_k\|\|_{ZP^{-1}A^{-1}}$	minimum $ZP^{-1}A^{-1}$-residual
Z sym., pos. def.	$\|\|\tilde{r}_k\|\|_Z$	pseudoresidual
$Z = \tilde{Z}AP$ and		
$\tilde{Z}$ sym., pos. def.	$\|\|r_k\|\|_{\tilde{Z}}$	minimum $\tilde{Z}$-residual
$Z = AP$	$\|\|r_k\|\|$	minimum residual
$Z = A^{-T}\tilde{Z}P$ and		
$\tilde{Z}$ sym., pos. def.	$\|\|e_k\|\|_{\tilde{Z}}$	minimum $\tilde{Z}$-error
$P = A^T Z$	$\|\|e_k\|\|$	minimum error

For systems with clustered eigenvalues, i.e., a small condition number, the relative norm of the errors is closely connected to the relative norm of the residuals, and the speed of convergence is fast. For systems with a large condition number error-minimizing generalized cg methods ($P = A^T Z$) guarantee that the errors do not increase. These methods are techniques used to prevent the separation of the errors from the residuals.

The speed of convergence is dependent on the eigenvalue distribution of the iteration matrices AP for the residuals, PA for the errors, respectively. For error-minimizing generalized cg methods these matrices are $AA^T Z$ and $A^T ZA$. If the eigenvalues of $A^T ZA$ are clustered, then error-minimizing generalized cg methods guarantee a fast convergence. An investigation for the choice of Z in order to obtain a fast convergence on supercomputers is given in [15].

For Example 1 the eigenvalues of the matrix $A^T A$ are more scattered than the eigenvalues of A because the matrix is symmetric. Therefore an error-minimizing generalized cg method with $Z = I$ (Craig's method CGNE [2]) would prevent the separation of the residuals from the errors, but the convergence would be very slow. As regards the CPU time, this would become even worse because one iteration step needs two matrix-vector multiplications. For this example and many others the simple iteration matrix A or A^T would generate a faster convergence. In the next section we try to construct such methods that minimize the error.

3. Generalized minimum error methods. In this section we introduce our second technique: generalized minimum error methods that use $A^T P$ as the iteration matrix instead of using $A^T ZA$, as in the previous section. The technique is a generalization of a method proposed by Fridman [6] for symmetric, positive definite matrices.

The construction of the methods is based on the following lemma.

LEMMA 5. *Let* $y_i \in \mathbb{R}^n$, $q_i = A^T y_i$ *for* $i = k - \sigma_k, \ldots, k-1$ *and*

$$x_k = \sum_{i=k-\sigma_k}^{k-1} \gamma_{i,k} q_i + x_{k-\sigma_k}. \tag{24}$$

Then the error of $Ax_k - b$ is minimized in the Euclidean norm, i.e.,

$$||e_k|| = \min_{\gamma_{k-\sigma_k,k},\ldots,\gamma_{k-1,k}} \left\| \sum_{i=k-\sigma_k}^{k-1} \gamma_{i,k} q_i + x_{k-\sigma_k} - x \right\|, \tag{25}$$

by the solution $(\gamma_{k-\sigma_k,k}, \ldots, \gamma_{k-1,k})$ of the linear system

$$\sum_{i=k-\sigma_k}^{k-1} \gamma_{i,k} q_i^T q_j = b^T y_j - x_{k-\sigma_k}^T q_j, \tag{26}$$

$j = k - \sigma_k, \ldots, k-1$. The following orthogonalities are especially valid for $j = k - \sigma_k, \ldots, k-1$:

$$e_k^T q_j = 0. \tag{27}$$

Proof. The error $||e_k||$ is minimized if

$$\frac{1}{2} \frac{\partial}{\partial \gamma_{j,k}} ||e_k||^2 = e_k^T q_j = 0 \quad \text{for} \quad j = k - \sigma_k, \ldots, k-1,$$

with

$$\begin{aligned} e_k^T q_j &= (x_k - x)^T q_j \\ &= \sum_{i=k-\sigma_k}^{k-1} \gamma_{i,k} q_i^T q_j - x^T q_j + x_{k-\sigma_k}^T q_j \\ &= \sum_{i=k-\sigma_k}^{k-1} \gamma_{i,k} q_i^T q_j - b^T y_j + x_{k-\sigma_k}^T q_j = 0 \end{aligned}$$

following from (26). □

The q_i are update directions for the iterate x_k, and the y_i are auxiliary vectors needed for the computation of the coeffients $\gamma_{i,k}$. By means of Lemma 5 a whole family of error-minimizing Krylov subspace methods can be derived.

DEFINITION 6. *Let x_0 be any initial guess for the solution of the system $Ax = b$. Choose an auxiliary starting vector $y_0 \neq 0$, $q_0 = A^T y_0$. The following recurrence is called a generalized minimum error method* (GMERR). *Choose a preconditioning matrix P and calculate the following for $k \geq 1$:*

$$x_k = \sum_{i=k-\sigma_k}^{k-1} \gamma_{i,k} q_i + x_{k-\sigma_k}. \tag{28}$$

Choose $y_k \in K_k(PA^T, y_0)$ so that

$$q_k^T Z q_{k-i} = 0 \tag{29}$$

for $i = 1, \ldots, \bar{\sigma}_k$, where Z is an auxiliary, nonsingular matrix and

$$q_k = A^T y_k. \tag{30}$$

The $\gamma_{i,k}$ are determined from

$$||e_k|| = \min_{\gamma_{k-\sigma_k,k},\ldots,\gamma_{k-1,k}} \left\| \sum_{i=k-\sigma_k}^{k-1} \gamma_{i,k} q_i + e_{k-\sigma_k} \right\| . \tag{31}$$

The method is called exact *if* $\bar{\sigma}_k = k$, restarted *if* $\bar{\sigma}_k = (k-1) \bmod \sigma_{\text{res}} + 1$ *with* σ_{res} *fixed,* truncated *if* $\bar{\sigma}_k = \min(k, \sigma_{\max})$ *with* $\sigma_{\max}$ *fixed, and* combined *if the truncated method is restarted. The method is called* consistent *if* $\sigma_k = \bar{\sigma}_k$.

The coefficients $\gamma_{i,k}$ can be calculated by (26). If $ZA^T P = P^T AZ$, then the iterates q_k can be calculated by a simple three-term recurrence, and the methods are equivalent to the classical algorithm of Fridman [6]. The optimal choice of y_0 would be $y_0 = A^{-T} e_0 = A^{-T} A^{-1} r_0$ because then $q_0 = e_0$, and the solution is obtained in the first iteration step.

From Definition 6 follow directly

$$||e_k|| \leq ||e_{k-1}|| , \tag{32}$$

$$x_k \in x_0 + K_{k-1}(A^T P, q_0), \tag{33}$$

$$r_k \in r_0 + A\, K_{k-1}(A^T P, q_0), \tag{34}$$

$$q_k \in K_k(A^T P, q_0) . \tag{35}$$

From Definition 6 and condition (34) follow for the residuals and the errors:

LEMMA 7. *For* GMERRs *the following hold*:

$$e_k = \sum_{i=0}^{k-1} \nu_{i,k} (A^T P)^i q_0 + e_0, \tag{36}$$

$$r_k = \sum_{i=0}^{k-1} \nu_{i,k} A\, (A^T P)^i q_0 + r_0 . \tag{37}$$

Proof. The proof is trivial. □

From Lemma 7 the norm of the error can be estimated.

THEOREM 8. *For exact, consistent,* GMERRs *holds*

$$||e_k|| = \min_{\theta_0,\ldots,\theta_{k-1}} \left\| \sum_{i=0}^{k-1} \theta_i (A^T P)^i q_0 + e_0 \right\| . \tag{38}$$

Proof. Equation (38) follows directly from (31) and (36). □

From (38) it follows directly that for $P = ZA$ and $y_0 = Zr_0$ we also obtain the error-minimizing generalized cg methods of the previous section as GMERRs (see Lemma 4):

$$||e_k|| = \min_{\beta_1,\ldots,\beta_k} \left\| \sum_{i=1}^{k} \beta_i (A^T ZA)^i e_0 + e_0 \right\| . \tag{39}$$

Thus we have the following duality: The preconditioning matrix $P = ZA$ for GMERRs (see (39)) corresponds to the preconditioning matrix $P = A^T Z$ of generalized cg methods (see (23)).

The next theorem shows the interconnections of GMERRs to a one-dimensional minimization technique, the smoothing algorithm.

THEOREM 9. *If* $Z = I$, *then the calculation of* x_k *simplifies for exact, consistent* GMERRs *to*

$$x_k = x_{k-1} + \gamma_{k-1,k} q_{k-1}, \tag{40}$$

with

$$\gamma_{k-1,k} = \frac{b^T y_{k-1} - x_{k-1}^T q_{k-1}}{q_{k-1}^T q_{k-1}}. \tag{41}$$

Proof. The orthogonalities in Equation (29) simplify (26). Therefore, $\gamma_{k-i,j} = \gamma_{k-i,l}$ for all j and l, and the $\gamma_{k-i,k}$ can be calculated explicitly for $i = 1, \ldots, \sigma_k$ by

$$\begin{aligned}\gamma_{k-i,k} &= \frac{b^T y_{k-i} - x_0^T q_{k-i}}{q_{k-i}^T q_{k-i}} \\ &= \frac{b^T y_{k-i} - (\sum_{j=0}^{k-i-1} \gamma_{j,k-i} q_j + x_0)^T q_{k-i}}{q_{k-i}^T q_{k-i}}\end{aligned}$$

because of the orthogonalities

$$= \frac{b^T y_{k-i} - x_{k-i}^T q_{k-i}}{q_{k-i}^T q_{k-i}}.$$

Thus $x_k = x_0 + \sum_{i=0}^{k-1} \gamma_{i,k} q_i = x_{k-1} + \gamma_{k-1,k} q_{k-1}$. □

Equations (41) and (40) can be considered a smoothing algorithm to minimize the error. Schönauer introduced a smoothing algorithm to minimize the residual (see, e.g., [12]). In [14] it is shown that this algorithm transforms generalized cg methods that minimize the pseudoresidual to methods that minimize the residual. Gutknecht [7] gives the reverse of this smoothing algorithm, which is itself a smoothing algorithm. Zhou and Walker [16] show that a smoothing algorithm transforms the BCG method [3], [9] to the QMR method [5] and that it transforms the CGS method [13] to TFQMR [4].

If $\gamma_{k-1,k}$ is calculated according to (41), then the algorithm will be stable in the sense that x_k is depending only on values of the previous iteration step.

A GMERR can be implemented as shown in the following algorithm:

Algorithm I. *Let x_0 be any initial guess. Choose an auxiliary starting vector y_0 and calculate $q_0 = A^T y_0$. Choose δ and $\bar{\delta}$ accordingly. For $k \geq 1$ choose a preconditioning matrix P and calculate*:

$$\gamma_{k-1,k} = \frac{b^T y_{k-1} - x_{k-1}^T q_{k-1}}{\|q_{k-1}\|^2}, \tag{42}$$

$$\text{if } |\gamma_{k-1,k}| \leq \delta \text{ for } k > 1,$$

then restart

$$x_k = x_{k-1} + \gamma_{k-1,k} q_{k-1}, \tag{43}$$

$$\alpha_{i,k} = -\frac{q_{k-i}^T Z A^T P q_{k-1}}{\|q_{k-i}\|_Z^2}, \quad \text{for } i = 0, \ldots, \sigma_k \tag{44}$$

$$\bar{q}_k = A^T P q_{k-1} + \sum_{i=1}^{\sigma_k} \alpha_{i,k} q_{k-i}\,, \tag{45}$$

$$\text{if } \|\bar{q}_k\|_Z \leq \bar{\delta},$$

then restart

$$\phi_k = \frac{1}{\|\bar{q}_k\|_Z}, \tag{46}$$

$$q_k = \phi_k \bar{q}_k, \tag{47}$$

$$y_k = \phi_k \left(P q_{k-1} + \sum_{i=1}^{\sigma_k} \alpha_{i,k} y_{k-i} \right). \tag{48}$$

This implementation is an Arnoldi-like algorithm for q_k. If we calculate

$$\phi_k = \frac{1}{\sum_{i=1}^{\sigma_k} \alpha_{i,k}}$$

instead of (46), then Algorithm I resembles an ORTHORES algorithm. Without the restart function the algorithm would break down if $\bar{q}_k = 0$. An invariant subspace or the whole space would then be spanned, and the iteration would be restarted from the defect correction equation.

The algorithm is also restarted if the iterate x_k does not change sufficiently. By numerical engineering the value of δ has been optimized to $\delta = 3 \cdot 10^{-3} \|x_{k-1}\|$. Note that the iteration restarts from the defect correction equation so that the iterate x_k is decreasing, thus justifying the choice of δ.

4. Comparison of residual-minimizing/error-minimizing methods. We test the convergence behavior of the residual-minimizing method GMRES [11], the error-minimizing generalized cg method CGNE [2], and a GMERR method. The intention is to get a feeling for how GMERR methods behave in comparison with the corresponding GMRES methods and the error-minimizing cg methods. GMRES is an exact generalized cg method that minimizes the Euclidean norm of the residuals ($Z = A$, $P = I$). Correspondingly, we select the exact GMERR method with $Z = P = I$ according to Algorithm I with $\bar{\delta} = 10^{-8}$ (14 digits accuracy). For CGNE holds $P = A^T$, $Z = I$. For all methods we choose $x_0 = 0$, and for GMERR $y_0 = r_0$.

Example 1 (continuation). As predicted, CGNE behaves worse than do GMRES and GMERR because the eigenvalues of the iteration matrix are more scattered. Unfortunately, GMERR does not converge faster than GMRES, but for GMERR the norm of the residuals is connected to the norm of the errors. Thus the user is not misled by the residuals, see Fig. 6. The bad convergence is dependent on the bad starting vector $x_0 = 0$.

Examples MIT1–MIT8. The examples are taken from [10]. We omit the first example MIT1 because the system matrix is the unit matrix and the solution is obtained in the first iteration step. The examples "nail down the space of matrices at every corner" [10]. The dimension of the system is 40 for MIT2, MIT3, MIT4, MIT6, and MIT7. The dimension of the system is 400 for MIT5 and MIT8. The solution was prescribed by the random number generator, and the right-hand side was calculated accordingly.

MIT2: A is a random matrix of dimension 40. All methods are similarly bad (see Fig. 7).

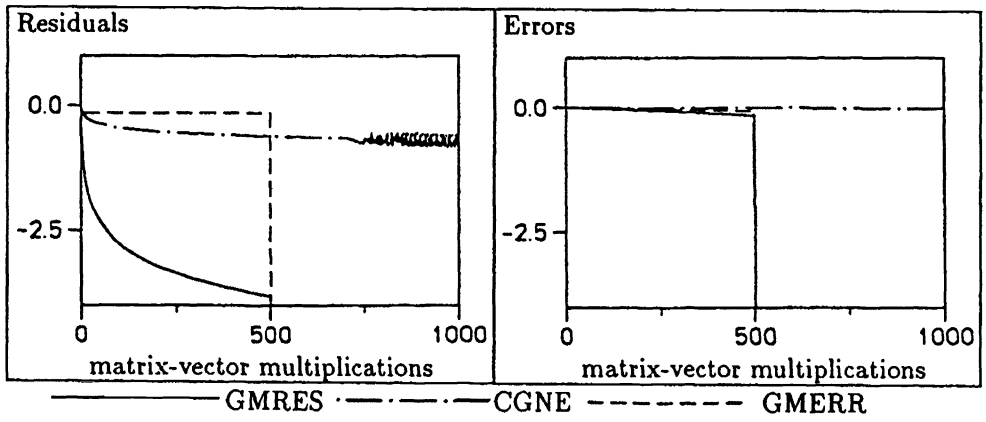


FIG. 6. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for Example* 1 *(logarithmic scale).*

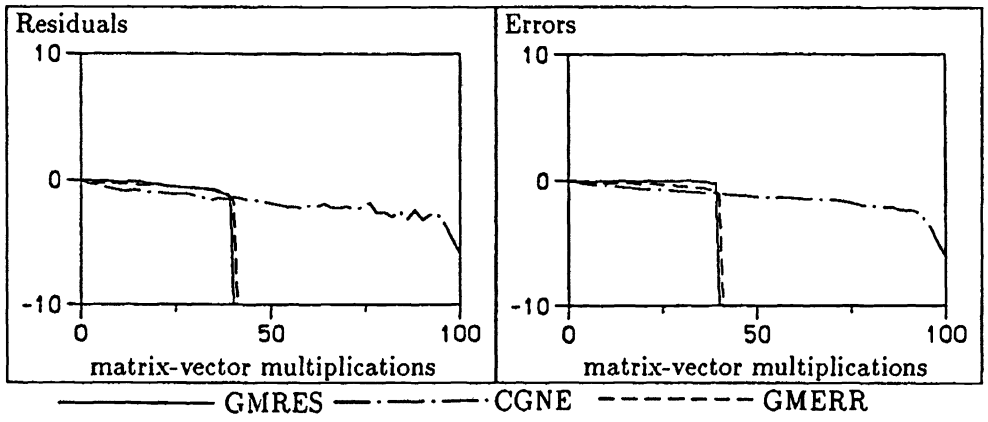


FIG. 7. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for example* MIT2 *(logarithmic scale).*

MIT3: The matrix of MIT3 is

$$A = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & & & \ddots & 1 \\ 1 & 0 & \dots & \dots & 0 \end{pmatrix}.$$

GMERR beats GMRES and gets the solution immediately because $q_0 = A^T y_0 = A^T r_0 = A^T A e_0 = e_0$ has an optimal direction. CGNE converges immediately because $A^T A = I$ (see Fig. 8). The eigenvalues of A are scattered and the eigenvalues of $A^T A$ are clustered. Therefore, CGNE should be better than GMERR. The opposite is true because of the artificial and special choice of A.

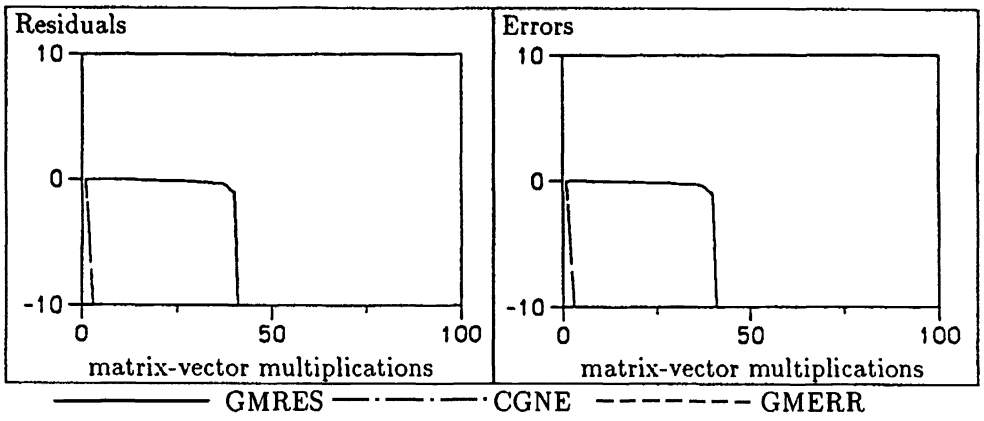


FIG. 8. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for example* MIT3 *(logarithmic scale).*

The structure of the matrix for MIT4, MIT6, MIT7, and MIT8 is

$$A = \begin{pmatrix} A_1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & A_{nn} \end{pmatrix}.$$

MIT4: $A_i = \begin{pmatrix} 1 & i-1 \\ 0 & 1 \end{pmatrix}$ The dimension is 40. GMRES is faster than GMERR and GMERR is faster than CGNE (see Fig. 9).

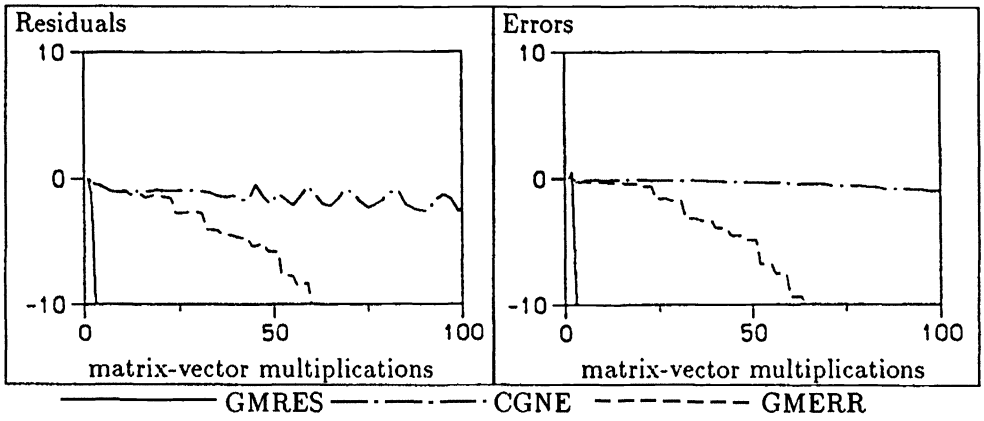


FIG. 9. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for example* MIT4 *(logarithmic scale).*

MIT5:

$$A = \operatorname{diag}(\xi_1, \dots, \xi_n) \quad \text{with} \quad \kappa = \left(\frac{1 + \epsilon^{\frac{1}{2\sqrt{n}}}}{1 - \epsilon^{\frac{1}{2\sqrt{n}}}} \right)^2,$$

$\xi_i = 1 + \frac{1}{2}(y_i + 1)(\kappa - 1)$, $y_i = \cos\frac{(i-1)\pi}{n-1}$, and $\epsilon = 10^{-10}$. The dimension is 400. GMRES is faster than GMERR, and GMERR is faster than CGNE (see Fig. 10).

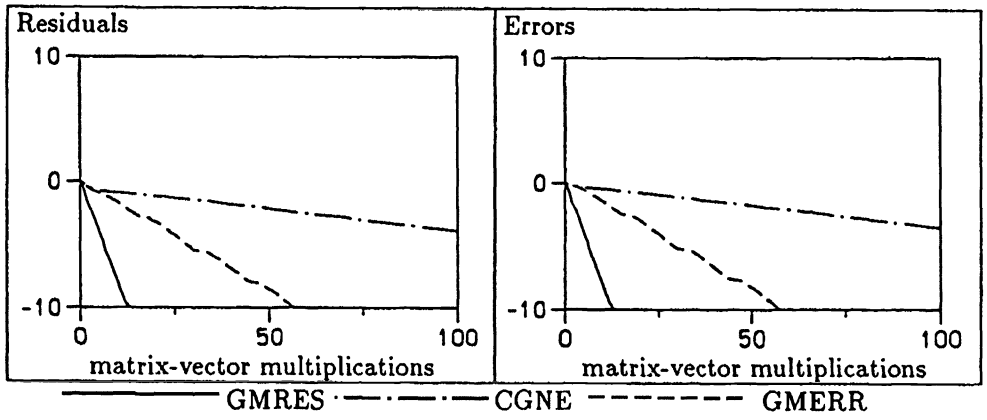


FIG. 10. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for example* MIT5 *(logarithmic scale).*

MIT6: $A_i = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$. The dimension is 40. All methods are very fast because $A^2 = A^{2T} = -I$ and $AA^T = I$ (see Fig. 11).

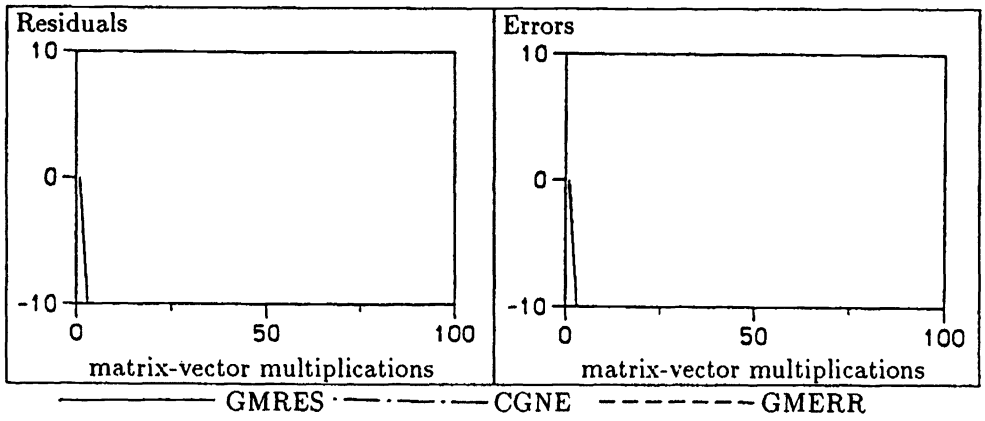


FIG. 11. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for example* MIT6 *(logarithmic scale).*

MIT7: $A_i = \begin{pmatrix} 1 & i-1 \\ 0 & -1 \end{pmatrix}$. The dimension is 40. GMRES is very fast, GMERR is very slow, and CGNE is even worse (see Fig. 12).

MIT8:

$$A_i = \begin{pmatrix} \xi_i & \nu_i \\ 0 & \frac{\kappa}{\xi_i} \end{pmatrix} \quad \text{with} \quad \kappa = \left(\frac{1 + \epsilon^{\frac{\sqrt{nn}}{2}}}{1 - \epsilon^{\frac{\sqrt{nn}}{2}}} \right)^2,$$

$\xi_i = 1 + \frac{1}{2}(y_i + 1)(\kappa - 1)$, $y_i = \cos\frac{(i-1)\pi}{nn-1}$ and $\epsilon = 10^{-10}$. The dimension is 400. GMERR is twice as fast as GMRES; CGNE is as fast as GMERR (see Fig. 13).

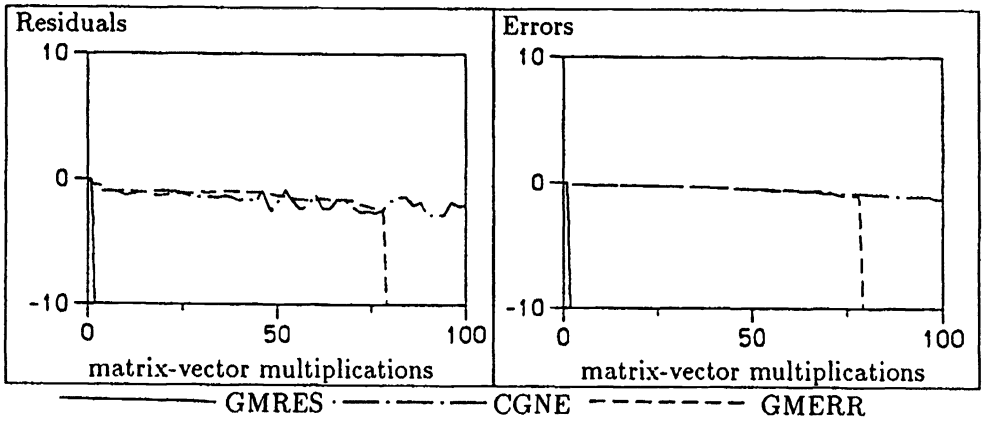


FIG. 12. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for example* MIT7 *(logarithmic scale).*

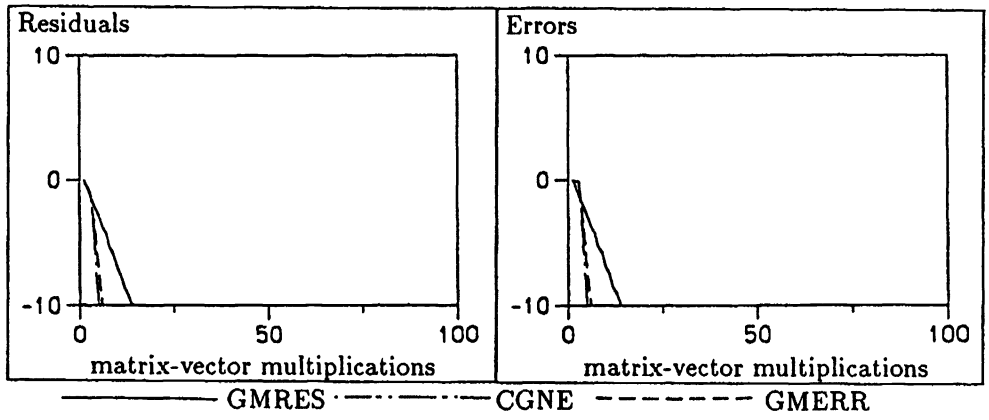


FIG. 13. *Norm of the relative residuals and errors of* GMRES, CGNE, *and* GMERR *for example* MIT8 *(logarithmic scale).*

For MIT2 and MIT6 GMRES and GMERR deliver comparable results. For MIT4, MIT5, and MIT7 GMRES is better; and for MIT3 and MIT8 GMERR beats GMRES. GMERR is better than CGNE for MIT4 and MIT5. For all other cases the two methods are comparable. In Table 2 the results are collected.

5. Outlook. The tests with the MIT examples indicate that exact, consistent GMERR according to Algorithm I is better than CGNE, and it is in some cases better than GMRES. But for more realistic problems, i.e., large problems arising from the discretization and linearization of partial differential equations, the exact methods are not feasible for storage requirements. Therefore, truncated and restarted versions have to be applied. While it was possible to optimize the restart and truncation parameters for GMRES-like methods, see e.g. [12], this optimization still lies ahead for GMERR methods. First tests indicate that this will be a more difficult task.

We further remark that GMERRs can be combined with generalized cg methods as we

TABLE 2
Comparison of GMRES, CGNE, *and* GMERR, *on a scale of* 1–3 (1=*best method*; 3=*worst method*).

Example	GMRES	CGNE	GMERR
MIT2	all methods bad		
MIT3	3	both 1	
MIT4	1	3	2
MIT5	1	3	2
MIT6	all methods good		
MIT7	1	both 3	
MIT8	3	both 1	

shall describe. For the biconjugate gradients (BCG) [3], [9] the double system $\hat{A}\hat{x} = \hat{b}$, i.e.,

$$\begin{pmatrix} A & 0 \\ 0 & A^T \end{pmatrix} \begin{pmatrix} x \\ x^* \end{pmatrix} = \begin{pmatrix} b \\ b^* \end{pmatrix},$$

is considered. b^* is arbitrary. The residuals have the form $\hat{r} = \binom{r}{r^*}$ and $Z = Z_B = \binom{0 \quad I}{I \quad 0}$. A generalized cg method is applied to this double system. As $Z_B \hat{A} = \hat{A}^T Z_B$, the method is exact for $\sigma = 2$, i.e., the sequence terminates automatically. Of course this doubling technique can be adapted to GMERRs in order to get terminating sequences. But we can also exploit this doubling differently. In the original BCG method the second sequence (*) is used only for the orthogonalization without exploitation of the iterates of the second sequence. But we can exploit the information of the second sequence for a generalized minimum error method especially because the iteration matrix of the second sequence is A^T as needed. Thus by modifying BCG slightly we obtain a generalized cg and a GMERR in parallel.

6. Conclusion. For problems with a single cluster of eigenvalues, i.e., in the symmetric case for well-conditioned problems, the errors decrease with the residuals, and the convergence of generalized cg methods is fast.

For problems with scattered eigenvalues, i.e., ill-conditioned problems in the symmetric case, the decrease of the residuals may be excellent while the errors do not decrease. As a consequence, we use error-minimizing methods. We have proposed two techniques. One is based on generalized cg methods with the iteration matrix $A^T Z A$, the other is based on GMERRs with the iteration matrix $A^T P$. We have shown various interconnections between the two techniques.

Because the speed of convergence is dependent on the eigenvalue distribution of the iteration matrix both techniques are suited for different problem types. The first is superior for scattered eigenvalues of the system matrix but more clustered eigenvalues of $A^T Z A$, i.e., for clustered singular values if $Z = I$. The second is preferable if the eigenvalues of $A^T Z A$, i.e., the singular values if $Z = I$, are more scattered than the eigenvalues of the system matrix.

REFERENCES

[1] C. BREZINSKI, M. R. ZAGLIA, AND H. SADOK, *Avoiding breakdown and near-breakdown in Lanczos type algorithms*, Numer. Algorithms, 1 (1991), pp. 261–284.

[2] E. J. CRAIG, *The n-step iteration procedures*, Math. Phys., 34 (1955), pp. 64–73.
[3] R. FLETCHER, *Conjugate gradient methods for indefinite systems*, in Proceedings of the Dundee Biennial Conference on Numerical Analysis, G. A. Watson, ed., Springer-Verlag, Berlin, New York, 1975, pp. 73–89.
[4] R. W. FREUND, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Comput., 14 (1993), pp. 470–482.
[5] 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.
[6] V. M. FRIDMAN, *The method of minimum iterations with minimum errors for a system of linear algebraic equations with a symmetrical matrix*, U.S.S.R. Comput. Math. and Math. Phys., 2 (1963), pp. 362–363.
[7] M. H. GUTKNECHT, *Changing the norm in conjugate gradient type algorithms*, SIAM J. Numer. Anal., 30 (1993), pp. 40–56.
[8] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–435.
[9] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.
[10] 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.
[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] W. SCHÖNAUER, *Scientific Computing on Vector Computers*, North-Holland, Amsterdam, New York, 1987.
[13] P. SONNEVELD, CGS, *a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.
[14] R. WEISS, *Convergence behavior of generalized conjugate gradient methods*, Internal Report 43/90, Computing Center, University of Karlsruhe, Karlsruhe, Germany, 1990.
[15] R. WEISS AND W. SCHÖNAUER, *Preconditioned conjugate gradient methods: What do we have and what do we need*? in Advances in Computer Methods for Partial Differential Equations, R. Vichnevetsky, D. Knight, and G. Richter, eds., International Association for Mathematics and Computers in Simulation, New Brunswick, NJ, 1992, pp. 806–812.
[16] L. ZHOU AND H. F. WALKER, *Residual smoothing techniques for iterative methods*, SIAM J. Sci. Comput., 15 (1992), pp. 297–312.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 528–544, May 1994

003

POSITIVITY CONDITIONS FOR QUARTIC POLYNOMIALS*

GARY ULRICH† AND LAYNE T. WATSON‡

Abstract. Simple necessary and sufficient conditions that a quartic polynomial $f(z)$ be nonnegative for $z>0$ or $a\leq z\leq b$ are derived, and illustrated geometrically. The geometry provides considerable insight and suggests various approximations and computational simplifications. The theory is applied to monotone quintic spline interpolation, giving necessary and sufficient conditions and an algorithm for monotone Hermite quintic interpolation.

Key words. monotone quintic spline interpolation, nonnegative polynomial, polynomial interpolation, quintic Hermite interpolation

AMS subject classifications. 41A05, 65D05, 65D07

1. Introduction.

Consider the fourth degree polynomial with real coefficients,

$$f(z) = az^4 + bz^3 + cz^2 + dz + e, \tag{1}$$

where $ae \neq 0$ (the problem reduces to consideration of a cubic if $ae = 0$). This paper outlines conditions under which this polynomial has positivity, i.e., $f(z) \geq 0$ for every $z > 0$. This is a rather general condition since positivity on any fixed interval (u, v) can be directly related to positivity on the positive reals through the transformation

$$t = \frac{u + zv}{1 + z}.$$

The property of positivity has a number of important applications to mathematics and computer science, and to shape preserving polynomial approximations in particular. For example, Jury and Mansour [6] relate positivity to a number of problems in control theory. Fritsch and Carlson [4], who derived positivity conditions in the case of quadratic polynomials, use the result to construct monotone cubic spline interpolants. Schmidt and Heß [8] provided conditions for positivity of cubic polynomials and used their result to construct positive cubic splines with minimum curvature and complex rational cubic splines. For the case of the fourth degree polynomial, Jury and Mansour [6] use the discriminant of (1) along with other characteristic expressions from the theory of equations to derive an algorithm for verifying positivity of quartics. Their conditions are difficult to implement and provide no geometric insight into the underlying mathematical phenomena. Dougherty, Edelman, and Hyman [3] derive conditions for monotonicity and convexity of quintic Hermite interpolants, but

* Received by the editors December 17, 1990; accepted for publication (in revised form) March 12, 1993.

† Systems Technology Center, AT&T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 07974-0636 (ulrich@ulysses.att.com).

‡ Departments of Computer Science and Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061-0106. The work of this author was supported in part by Department of Energy grant DE-FG05-88ER25068 and Air Force Office of Scientific Research grant 89-0497 (ltw@vtopus.cs.vt.edu).

explicitly state only sufficient conditions. Our results are very similar to those of [3], but differ in that we give elegant sharp (necessary and sufficient) conditions for positivity of a quartic directly in terms of its coefficients, whereas [3] couches its sufficient conditions in terms of derivatives of Hermite quintics. We also give sharp conditions on the second derivatives, for given fixed first derivatives, such that a quintic Hermite interpolant is monotone; this question is not addressed by [3].

Some time ago, deBoor and Swartz [2] addressed monotone spline interpolation in general and the cubic case in detail, and monotonicity was also recently considered by Huynh [5] and Ulrich and Watson [11].

The present paper uses a simpler, but equivalent, form of the polynomial (1) to obtain positivity conditions which are as elegant as those available for the quadratic and cubic polynomials. The quartic polynomial is reparameterized to this simpler form in §2 where the regions of positivity are described and a formal proof of positivity is provided. The geometric characterization of positivity in §2 provides considerable insight, and easily leads to various approximate criteria and computational simplifications. Section 3 outlines the conditions used to test for positivity and provides some heuristics on constraining a nonpositive polynomial to be positive. Section 4 briefly describes applications of this result in polynomial interpolation.

All of the algebra in this paper was done with Mathematica [12], and thus algebraic derivations showing intermediate steps are not given here.

2. Regions of positivity. Consider the transformation used in [8], e.g.,

$$x^4 = \frac{a}{e} z^4.$$

This reparameterization should exist since a necessary condition for positivity of the polynomial (1) is that the coefficients a and e are both positive. The polynomial $f(z)/e$ then becomes

$$x^4 + ba^{-3/4}e^{-1/4}x^3 + ca^{-1/2}e^{-1/2}x^2 + da^{-1/4}e^{-3/4}x + 1,$$

which can be written as the polynomial

$$p(x;\alpha,\beta,\gamma) = x^4 + \alpha x^3 + \beta x^2 + \gamma x + 1, \tag{2}$$

where α, β, and γ are defined appropriately. This reparameterized polynomial has only three coefficients. In previous work on the quadratic and cubic equations, the regions of nonpositive roots were bounded by parametric curves corresponding to double roots of the polynomial. The same holds true for the quartic polynomial (2) above. A well-known result in the theory of equations is that double roots exist when the discriminant

$$\Delta = 4\left[c^2 - 3bd + 12ae\right]^3 - \left[2c^3 + 27ad^2 + 27b^2e - 9bcd - 72ace\right]^2$$

of the polynomial (1) is zero (a proof is given in [7]). Setting $\Delta = 0$ leads to the sixth order equation

$$\Delta = 4\left[\beta^2 - 3\alpha\gamma + 12\right]^3 - \left[72\beta + 9\alpha\beta\gamma - 2\beta^3 - 27\alpha^2 - 27\gamma^2\right]^2 = 0. \tag{3}$$

The discriminant for the quadratic and cubic polynomials was a second and fourth degree polynomial, respectively, and region inequalities could be represented and derived in a straightforward manner. In the case of the quartic, the discriminant is a sixth order polynomial and we see no way of approaching the problem directly; we will construct a geometric argument which utilizes the parametric form of the double root boundary. The presence of a double root (say t) implies that

$$p'(t) = 4t^3 + 3\alpha t^2 + 2\beta t + \gamma = 0 \quad \text{and} \quad 4p(t) - t\,p'(t) = \alpha t^3 + 2\beta t^2 + 3\gamma t + 4 = 0,$$

where the latter equation comes from the fact that $1/t$ is also a double root of $x^4\,p(1/x)$. Solving simultaneously for α and γ (for fixed β), we obtain parametric equations for $t \in (-\infty, \infty)$,

$$\alpha(t) = \frac{1 - \beta t^2 - 3t^4}{2t^3} \quad \text{and} \quad \gamma(t) = \frac{t^4 - \beta t^2 - 3}{2t}. \tag{4}$$

Symmetry clearly shows up in these parametric equations since $\alpha(-t) = -\alpha(t)$, $\gamma(-t) = -\gamma(t)$, $\alpha(1/t) = \gamma(t)$, and $\gamma(1/t) = \alpha(t)$. By applying curve tracing techniques, we can identify three distinct shapes (corresponding to zero, one, or two cusps on each component) for the double root curves as shown in Fig. 1 (important features of these curves are also labeled in the figure). Figure 2 shows the family of double root curves as β varies, plotting only the top portions.

The region of positivity for the quartic, like the quadratic and cubic, is bounded by the double root curve. Trying to prove this directly from (4), however, does not work out well. To prove this result, we employ the following theorem (from [1], [9]):

THEOREM 0. *The quartic polynomial $g(x)$ is nonnegative for all $x \geq 0$ if and only if there exist polynomials $u(x)$ and $v(x)$ such that*

$$g(x) = u^2(x) + x\,v^2(x).$$

Rewriting (2) as

$$p(x;\alpha,\beta,\gamma) = \left[x^2 + rx + s\right]^2 + x\left[(\alpha - 2r)x^2 + (\beta - (r^2 + 2s))x + (\gamma - 2rs)\right]$$

for some r and for $s = \pm 1$, then the above theorem implies that $p(x)$ is nonnegative for all $x \geq 0$ if and only if there exists real r such that

$$(\alpha - 2r)x^2 + (\beta - (r^2 + 2s))x + (\gamma - 2rs) \tag{5}$$

is a perfect square. If (5) is a nontrivial perfect square, the first and last coefficients must be positive. The values of r which make (5) a perfect square correspond to the roots of the quartic

$$q(r;\alpha,\beta,\gamma) = \left[\beta - (r^2 + 2s)\right]^2 - 4(\alpha - 2r)(\gamma - 2rs) = 0, \tag{6}$$
$$\text{where} \quad \alpha - 2r > 0 \quad \text{and} \quad \gamma - 2rs > 0.$$

(The limiting cases $\alpha - 2r = 0$ and $\gamma - 2rs = 0$ corresponding to $\beta - (r^2 + 2s) = 0$ are also possible and have trivial solutions, so we will not clutter the discussion by mentioning

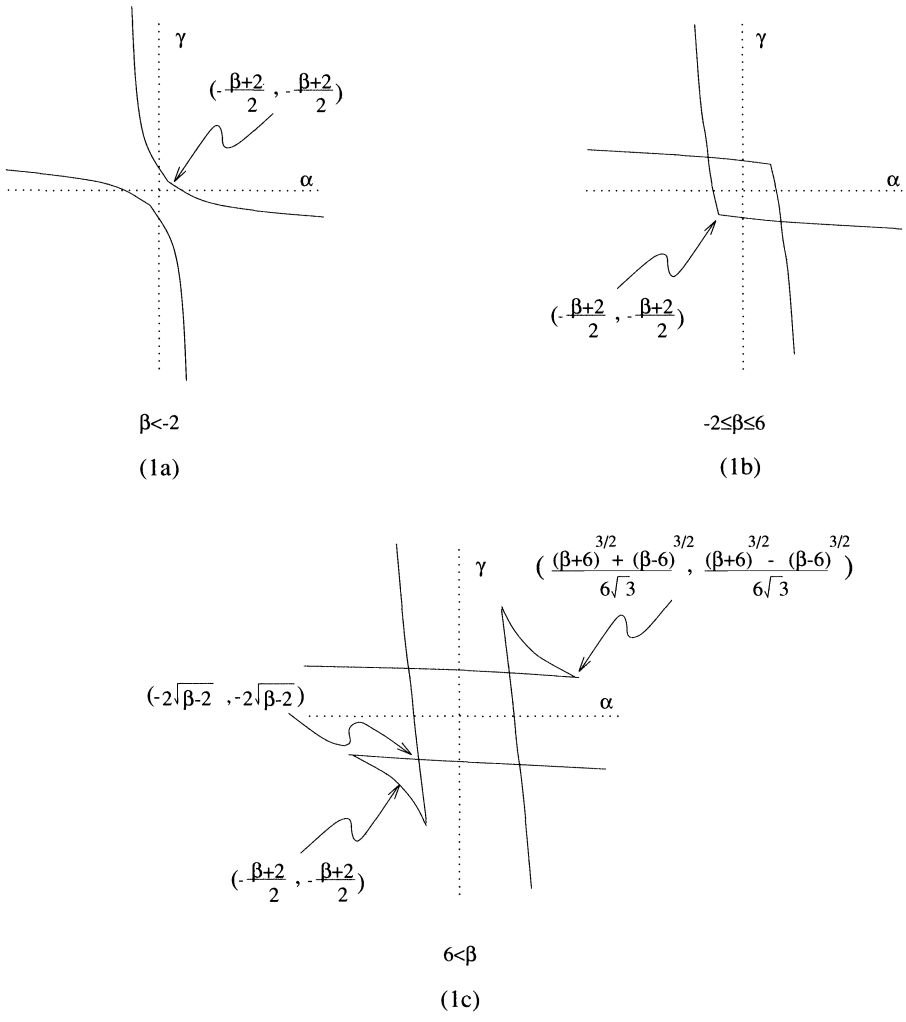


FIG. 1. *Parametric curves from* (4).

these special cases each time.) The discriminant of this quartic is proportional to (3), indicating that the quartic equations (2) and (6) have the same double root boundary; even though the double root boundary is the same for the two quartics (i.e., equations (2) and (6)), their parametric representations differ. Proceeding from (6), as was done from (2) to (4), gives parametric expressions $\alpha(r)$, $\gamma(r)$ in terms of a double root r of (6). Cleaner expressions result from writing the double root as $\pm\,\mathrm{sgn}(t)\,t$, where t is simply a parameter with restrictions obvious from the structure of the formulas. For $s = 1$ and $t^2 \geq 4$, a double root of (6) is $-\,\mathrm{sgn}(t)\,t$ and the corresponding boundary is parameterized as

$$\alpha(t) = \mathrm{sgn}(t)\left\{-2t + \frac{(t^2 - \beta + 2)}{4}\left[t + \sqrt{t^2 - 4}\right]\right\} \tag{7a}$$

and

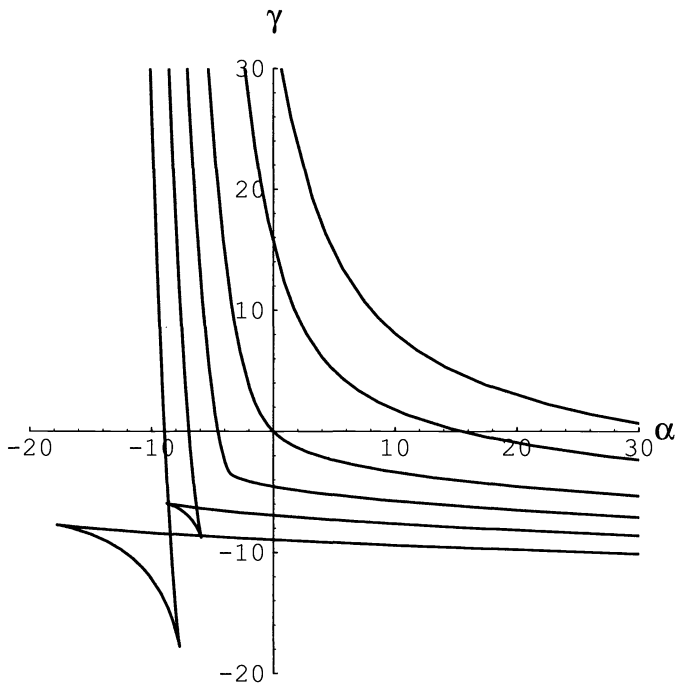


FIG. 2. *Top components of curves from* (4) *for* β=−20, −12, −2, 5, 12, 20 (*top to bottom*).

$$\gamma(t) = \operatorname{sgn}(t)\left\{-2t + \frac{(t^2-\beta+2)}{4}\left[t - \sqrt{t^2-4}\right]\right\}. \tag{7b}$$

This curve $(\alpha(t), \gamma(t))$ is the top half of the double root boundary in Fig. 1. The bottom half of the double root boundary corresponds to the double root $\operatorname{sgn}(t)\, t$ and parameterization $(-\alpha(t), -\gamma(t))$.

For $s = -1$ and $t \in (-\infty, \infty)$, the double root is t and the parametric representation of the top half of the double root boundary is

$$\alpha(t) = 2t + \frac{(t^2-\beta-2)}{4}\left[t + \sqrt{t^2+4}\right] \tag{8a}$$

and

$$\gamma(t) = -2t - \frac{(t^2-\beta-2)}{4}\left[t - \sqrt{t^2+4}\right]. \tag{8b}$$

The bottom half of the double root boundary corresponds to the double root t and parameterization $(-\gamma(t), -\alpha(t))$. The substitution of $w^2 = t^2 - 4$ in equation (7) would lead to equation (8), showing that these parameterizations are equivalent. For technical reasons that only become apparent much later, (7) is the cleanest form of the three parametrizations (4), (7), (8) to work with. Only equation (7) is needed for the proof of the following theorem concerning the region of positivity:

THEOREM 1. *For fixed* β, *the region of positivity for the quartic polynomial* (2) *is bounded below by the curve* Γ_β, *which is defined by* (7) *for* $t^2 \geq \max\{4, \beta - 2\}$. *In other words, for any* (α, γ), *the polynomial* $p(x; \alpha, \beta, \gamma)$ *has positivity if and only if there exist* (α^*, γ^*) *on* Γ_β *and* $\delta \geq 0$ *such that* $\alpha = \alpha^* + \delta$ *and* $\gamma = \gamma^* + \delta$.

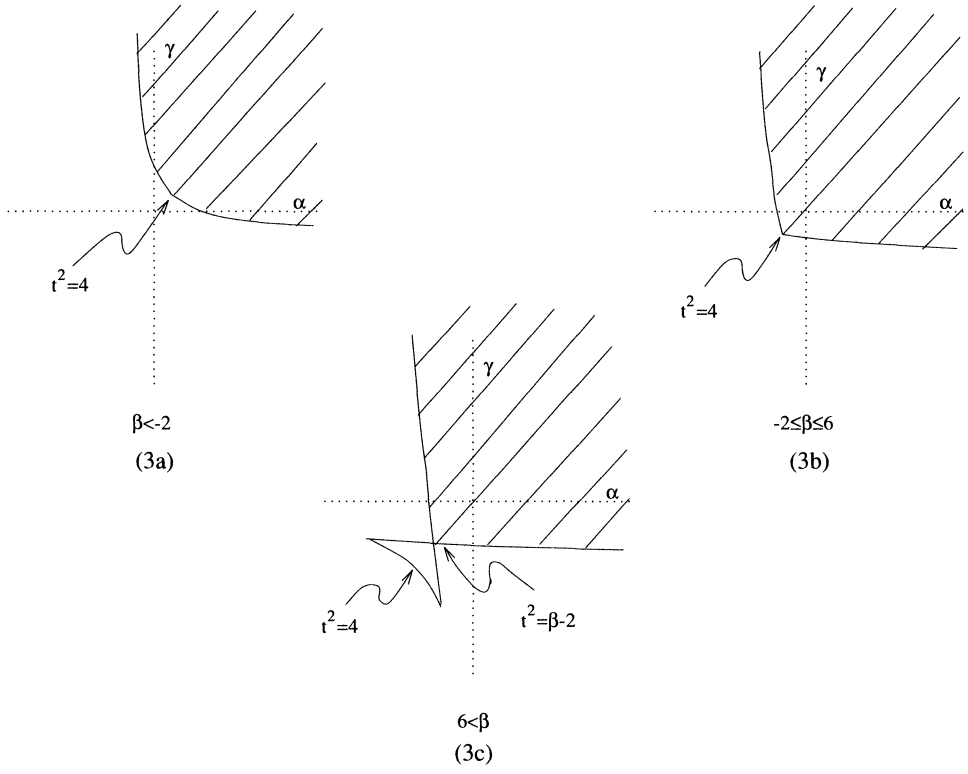


FIG. 3. *Regions of positivity for the quartic polynomial.*

Proof. For any given β, the region of positivity is depicted graphically in Fig. 3. In each case, the parametric curve Γ_β is convex; this is easily proved by looking at the signs of the first and second partial derivatives:

$$\frac{\partial \alpha^*}{\partial \gamma^*} = \frac{\sqrt{t^2-4}+t}{\sqrt{t^2-4}-t} < 0, \quad \text{for} \quad t^2 \geq \max\{4, \beta-2\},$$

$$\frac{\partial^2 \alpha^*}{\partial \gamma^{*2}} = \operatorname{sgn}(t) \frac{4\left(\sqrt{t^2-4}+t\right)}{\left(3t^2-6-\beta\right)\left(t^2-2-t\sqrt{t^2-4}\right)} > 0,$$

$$\text{for} \quad t^2 > \max\{4, \beta-2\} \geq \frac{\beta+6}{3}.$$

(An alternative argument for convexity is that the set of (α, β, γ) for which (2) is nonnegative for all $x \geq 0$ is convex, hence every constant β slice is convex, hence the bounding curve Γ_β must also be convex.) Assume for the moment that $s = 1$. For a fixed β, consider (α', γ') defined by

$$\alpha' = \alpha^*(t) + \delta, \qquad \gamma' = \gamma^*(t) + \delta,$$

for some value of $\delta \geq 0$ and for some t such that $t^2 > \max\{4, \beta-2\}$, and for $(\alpha^*(t), \gamma^*(t))$ given by equation (7). Since the parametric curve, Γ_β, is part of the

double root boundary of equation (6), and since $(\alpha^*(t), \gamma^*(t)) \in \Gamma_\beta$, then q from equation (6) with double root $-\operatorname{sgn}(t)\, t$ can be factored as

$$q(r; \alpha^*(t), \beta, \gamma^*(t)) = \big(r + \operatorname{sgn}(t)\, t\big)^2 \big(r^2 - 2\operatorname{sgn}(t)\, rt + 3t^2 - 2(\beta + 6)\big) \tag{9}$$

and $q(-\operatorname{sgn}(t)\, t; \alpha^*(t), \beta, \gamma^*(t)) = 0$. The double roots satisfy the conditions of equation (6). The other pair of roots,

$$\operatorname{sgn}(t)\, t \pm \sqrt{2(\beta + 6 - t^2)},$$

are complex when $t^2 > \beta + 6$, but do not make the lead and tail coefficients of (5) positive when they are real. Hence, the double roots $-\operatorname{sgn}(t)\, t$ are the only roots satisfying (6) and its side conditions along Γ_β.

We next show that if $p(x; \alpha, \beta, \gamma)$ has positivity, then so does $p(x; \alpha+\delta, \beta, \gamma+\delta)$ for any $\delta > 0$, i.e., moving northeast from a point of positivity (α, γ) preserves positivity. For simplicity, consider first the case of t negative, i.e., $t < \min\{-2, -\sqrt{\beta - 2}\}$ for which we have $\alpha^*(t) \le \gamma^*(t)$ and $q(t; \alpha^*(t), \beta, \gamma^*(t)) = 0$. It is obvious that $\alpha^*(t) - 2t > 0$ and $\gamma^*(t) - 2t > 0$, which means that the conditions of (6) are satisfied and $p(x; \alpha^*(t), \beta, \gamma^*(t))$ is nonnegative for positive x. Substituting $(\alpha', \beta, \gamma')$ into the function q yields

$$\begin{aligned} q(t; \alpha', \beta, \gamma') &= q(t; \alpha^*(t) + \delta, \beta, \gamma^*(t) + \delta) \\ &= q(t; \alpha^*(t), \beta, \gamma^*(t)) - 4\big[\delta(\alpha^*(t) + \gamma^*(t) - 4t) + \delta^2\big] \\ &= -4\big[\delta(\alpha^*(t) + \gamma^*(t) - 4t) + \delta^2\big]. \end{aligned} \tag{10}$$

When $\delta > 0$, this last term is negative since $\alpha^*(t) + \gamma^*(t) - 4t > 0$. Noting that the $\limsup_{r \to -\infty} q(r; \alpha', \beta, \gamma') = +\infty > 0$, it is clear that there exists a root, say $t' < t$, such that

$$q(t'; \alpha', \beta, \gamma') = 0, \quad \alpha' - 2t' > 0, \quad \gamma' - 2t' > 0.$$

Hence, $p(x; \alpha', \beta, \gamma')$ is nonnegative for positive x. Notice that this argument did not require starting on the double root boundary; it uses the fact that t is a root satisfying equation (6) for a given pair of coefficients, (α, γ), to show there exists a root t_δ which satisfies the equation for the pair $(\alpha + \delta, \gamma + \delta)$, for all $\delta > 0$.

For small $\delta < 0$, equation (10) indicates that the effect of the perturbation is to add a positive increment (in the form of a linear term) near the double roots so that they become complex. This has negligible effect on the other two roots which did not satisfy (6) so that, within a sufficiently small neighborhood below the double root boundary, the conditions of (6) are not satisfied. But this in turn proves that for any (α, γ) below the double root boundary, such that $\alpha < \gamma$, the conditions of (6) are not satisfied; if this were not true we could choose a $\delta > 0$ such that $(\alpha + \delta, \gamma + \delta)$ lay in the neighborhood just described, which would be a contradiction of the argument given in (10).

A similar analysis can be performed for positive t, i.e., $t > \max\{2, \sqrt{\beta - 2}\}$, with similar conclusions for $\alpha > \gamma$. Since Γ_β is convex, every (α, γ) in the region of

positivity can be reached from Γ_β through an appropriate choice of $(\alpha^*(t), \gamma^*(t))$ and $\delta > 0$.

We have shown that $p(x; \alpha, \beta, \gamma)$ has positivity for every (α, γ) above the curve Γ_β, and that for (α, γ) below Γ_β, $q(r; \alpha, \beta, \gamma)$ has no roots satisfying (6) for $s = 1$. Thus, all that remains to prove is that for (α, γ) below Γ_β, $q(r; \alpha, \beta, \gamma)$ has no roots satisfying (6) for $s = -1$.

So now assume $s = -1$, β is fixed, $\alpha' = \alpha^*(t) + \delta_1$, $\gamma' = \gamma^*(t) + \delta_2$, $t^2 > \beta + 2$, where $(\alpha^*(t), \gamma^*(t))$ is given by (8). Recall that the double root boundary described by (8) is the same as that given by (7), and thus Γ_β lies on the curve defined by (8). The double root t satisfies (6) if and only if $t^2 \geq \beta + 2$, so only those double roots need be considered. For double root t,

$$q(r; \alpha, \beta, \gamma) = (r - t)^2\big(r^2 + 2rt + 3t^2 + 12 - 2\beta\big),$$

and the other two roots are

$$-t \pm \sqrt{2(\beta - 6 - t^2)},$$

which are both complex since $t^2 > \beta + 2 > \beta - 6$. Therefore, the double root t is the only root satisfying (6) near the portion of Γ_β corresponding to $t^2 > \beta + 2$. Arguing as before, where now the sign of t does not matter, we conclude that $q(r; \alpha', \beta, \gamma')$ has a root satisfying (6) for any $\delta_1 > 0$ and $\delta_2 > 0$, but not for small $\delta_1 < 0$, $\delta_2 < 0$. If $q(r; \alpha, \beta, \gamma)$ had a root satisfying (6) for (α, γ) below Γ_β, then the analog of (10) would prove that $q(r; \alpha + \delta_1, \beta, \gamma + \delta_2)$ also had a root satisfying (6) for $(\alpha + \delta_1, \gamma + \delta_2)$ arbitrarily near and below the part of Γ_β corresponding to $t^2 > \beta + 2$, for appropriately chosen $\delta_1 > 0$, $\delta_2 > 0$. (It is always possible to choose such δ_i because Γ_β is convex and $\partial\gamma/\partial\alpha < 0$ along Γ_β.) Therefore we conclude that q has no roots satisfying (6) for (α, γ) below Γ_β. □

3. Positivity conditions. Verifying that a point (α, γ) lies above Γ_β is nontrivial using (7), and therefore computationally simple tests are sought for the positivity regions in Fig. 3. A continuity argument implies that the discriminant in (3) changes sign every time a double root boundary is crossed. This leads to the distribution of signs for Δ, in the three types of regions, as shown in Fig. 4. The conditions for positivity in the first case ($\beta < -2$) are simply

$$\Delta \leq 0 \quad \text{and} \quad \alpha + \gamma > 0. \tag{11}$$

For the second case, $-2 \leq \beta \leq 6$, we can use (11) as one of the conditions, but need another condition to cover the middle region with Δ positive. Linear bounds on (α, γ) could be used in this case, but a more compact representation is obtained by using a parabola,

$$\Lambda_1 \equiv (\alpha - \gamma)^2 - 16(\alpha + \beta + \gamma + 2) = 0, \tag{12}$$

which surrounds the positive middle region (see Fig. 5b). The algebra required to show that the parabola indeed lies below the curve Γ_β on the positive middle region is straightforward but tedious (because of convexity, it suffices to check the slopes at

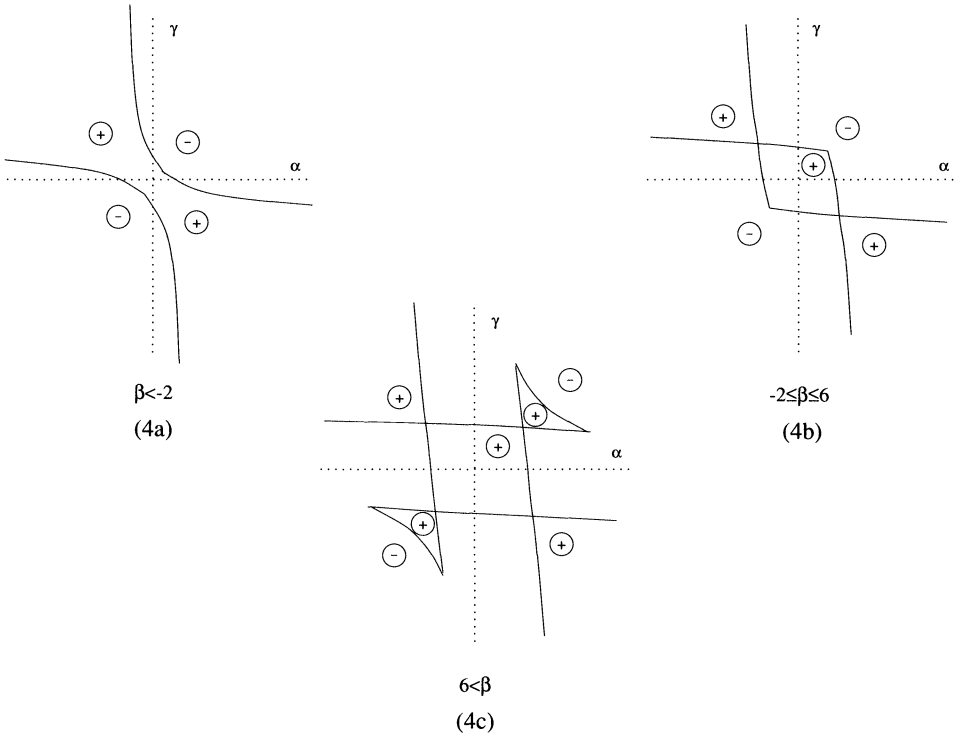


FIG. 4. *Sign distribution for the discriminant* Δ *of the quartic.*

the three points where Λ_1 intersects Γ_β). The resulting pair of conditions in the case of $-2 \leq \beta \leq 6$ are

$$\begin{aligned} &\Delta \leq 0 \quad \text{and} \quad \alpha + \gamma > 0 \\ &\qquad\quad \text{or} \\ &\Delta \geq 0 \quad \text{and} \quad \Lambda_1 \leq 0. \end{aligned} \tag{13}$$

In a similar way, a parabolic curve can be constructed to bound the middle region (see Fig. 5c) in the final case, $\beta > 6$, leading to the curve:

$$\Lambda_2 \equiv (\alpha - \gamma)^2 - \frac{4(\beta+2)}{\sqrt{\beta-2}}\left(\alpha + \gamma + 4\sqrt{\beta-2}\right) = 0. \tag{14}$$

This last case is more complicated than the previous one because the parabolic curve will not contain the tips of the cusp region for large β; however, noting that the cusp is always contained in the first quadrant leads to the following three positivity conditions corresponding to $\beta > 6$:

$$\begin{aligned} &\Delta \leq 0 \quad \text{and} \quad \alpha + \gamma > 0 \\ &\qquad\quad \text{or} \\ &\alpha > 0 \quad \text{and} \quad \gamma > 0 \\ &\qquad\quad \text{or} \\ &\Delta \geq 0 \quad \text{and} \quad \Lambda_2 \leq 0. \end{aligned} \tag{15}$$

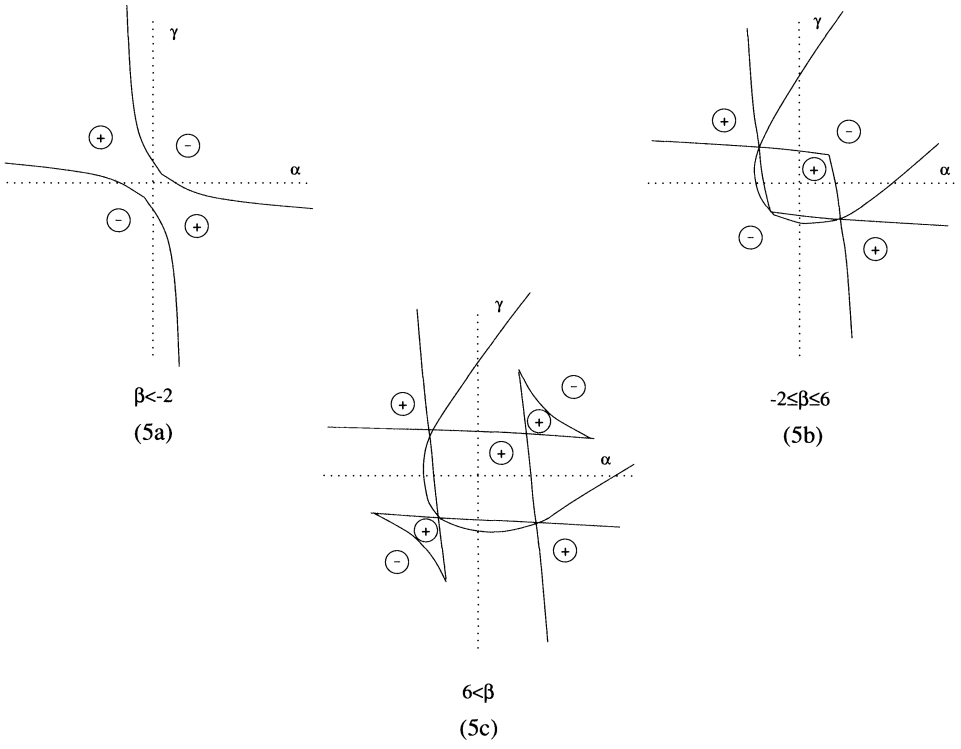


FIG. 5. *Positivity conditions for the quartic polynomial.*

A graphical depiction of these conditions is shown in Fig. 5 for the three cases. The theorem from §2 and these simple tests for positivity can be summarized in:

THEOREM 2. *Let* $f(z) = az^4 + bz^3 + cz^2 + dz + e$ *be a quartic polynomial with real coefficients and* $a > 0$, $e > 0$. *Define*

$$\alpha = ba^{-3/4}e^{-1/4}, \qquad \beta = ca^{-1/2}e^{-1/2}, \qquad \gamma = da^{-1/4}e^{-3/4},$$

$$\Delta = 4\left[\beta^2 - 3\alpha\gamma + 12\right]^3 - \left[72\beta + 9\alpha\beta\gamma - 2\beta^3 - 27\alpha^2 - 27\gamma^2\right]^2,$$

$$\Lambda_1 \equiv (\alpha-\gamma)^2 - 16(\alpha+\beta+\gamma+2), \qquad \Lambda_2 \equiv (\alpha-\gamma)^2 - \frac{4(\beta+2)}{\sqrt{\beta-2}}\left(\alpha+\gamma+4\sqrt{\beta-2}\right).$$

Then $f(z) \geq 0$ *for all* $z > 0$ *if and only if*

$$(1) \qquad \beta < -2 \quad \textit{and} \quad \Delta \leq 0 \quad \textit{and} \quad \alpha + \gamma > 0;$$

$$(2) \qquad -2 \leq \beta \leq 6 \quad \textit{and} \quad \begin{cases} \Delta \leq 0 \quad \textit{and} \quad \alpha + \gamma > 0 \\ \textit{or} \\ \Delta \geq 0 \quad \textit{and} \quad \Lambda_1 \leq 0; \end{cases}$$

$$(3) \qquad 6 < \beta \quad \textit{and} \quad \begin{cases} \Delta \leq 0 \quad \textit{and} \quad \alpha + \gamma > 0 \\ \textit{or} \\ \alpha > 0 \quad \textit{and} \quad \gamma > 0 \\ \textit{or} \\ \Delta \geq 0 \quad \textit{and} \quad \Lambda_2 \leq 0. \end{cases}$$

These conditions can be costly to implement and a set of simplified sufficient conditions can be used as a pretest for positivity. From Fig. 1, it should be clear that the following conditions are sufficient for positivity:

$$\text{(16)}\qquad \begin{aligned} &\text{(i)} \quad \alpha > -\frac{\beta+2}{2} \quad \text{and} \quad \gamma > -\frac{\beta+2}{2} \quad \text{for} \quad \beta \le 6; \\ &\text{(ii)} \quad \alpha > -2\sqrt{\beta-2} \quad \text{and} \quad \gamma > -2\sqrt{\beta-2} \quad \text{for} \quad \beta > 6. \end{aligned}$$

4. Applications. The initial motivation for this work was in the application of Hermite interpolation to construction of random number algorithms for arbitrary continuous distributions. Results from a previous paper [10] suggested that a fifth degree piecewise polynomial approximation could achieve accuracy comparable to that of an exact algorithm implemented in single precision for many common distributions. That same paper outlined an approach for constructing the piecewise polynomial interpolant but indicated that, even though the inverse cdf was monotone, piecewise interpolants using higher order polynomials might not be monotone. The problem of testing for monotonicity of cubic interpolants was solved by Fritsch and Carlson [4], of quartic interpolants by [8], and of quintic interpolants (sufficiency) by and [3] and [11]. Theorem 1 can be used in testing for monotonicity of the quintic interpolant as described below.

Suppose that a quintic interpolant is constructed on the interval (U_0, U_1) to match the ordinates and first and second derivatives of the function $f(U)$, e.g.,

$$\begin{aligned} X_0 &= f(U_0), \quad X_0' = f'(U_0), \quad X_0'' = f''(U_0), \\ X_1 &= f(U_1), \quad X_1' = f'(U_1), \quad X_1'' = f''(U_1). \end{aligned}$$

The fifth degree Hermite polynomial interpolant on (U_0, U_1) can be written as

$$\text{(17)}\qquad \begin{aligned} X(u) = \frac{1}{16}\Big\{ &\big[h^2(X_1'' - X_0'') - 3h(X_0' + X_1') + 3(X_1 - X_0)\big]u^5 \\ &+ \big[h^2(X_0'' + X_1'') - h(X_1' - X_0')\big]u^4 \\ &+ \big[-2h^2(X_1'' - X_0'') + 10h(X_0' + X_1') - 10(X_1 - X_0)\big]u^3 \\ &+ \big[-2h^2(X_0'' + X_1'') + 6h(X_1' - X_0')\big]u^2 \\ &+ \big[h^2(X_1'' - X_0'') - 7h(X_0' + X_1') + 15(X_1 - X_0)\big]u \\ &+ \big[h^2(X_0'' + X_1'') - 5h(X_1' - X_0') + 8(X_0 + X_1)\big]\Big\}, \end{aligned}$$

where $u = (U - \bar{U})/h$, $\bar{U} = (U_0 + U_1)/2$, and $h = (U_1 - U_0)/2$. The polynomial in (17) is monotone on (U_0, U_1) if and only if its derivative is nonnegative over the interval. Taking the derivative and using the transformation $u = \frac{z-1}{z+1}$ we obtain the test polynomial

$$\text{(18)}\qquad Bz^4 + 4(B - D)z^3 + 6(D - C - 2B - 2A + 5)z^2 + 4(A + C)z + A,$$

where $A = \frac{2hX_0'}{\nu}$, $B = \frac{2hX_1'}{\nu}$, $C = \frac{h^2X_0''}{\nu}$, $D = \frac{h^2X_1''}{\nu}$, and $\nu = (X_1 - X_0)$; the quintic in (17) is monotone if and only if (18) is nonnegative for $z > 0$. (The special case $AB = 0$

reduces to a cubic, and will not be considered further. Thus we assume $A > 0$, $B > 0$ henceforth.)

This quartic can be reparameterized to fit the form of equation (2) by defining

$$\alpha = \frac{4(B-D)}{A^{1/4}B^{3/4}}, \qquad \beta = \frac{6(D-C-2B-2A+5)}{A^{1/2}B^{1/2}}, \qquad \gamma = \frac{4(A+C)}{A^{3/4}B^{1/4}}. \tag{19}$$

Theorem 2 can be applied to these coefficient values to determine whether the polynomial in (17) is monotone. If the quintic is not monotone then we can adjust the derivative values to make it monotone over (U_0, U_1). One strategy is to scale the derivative vector $(X_0', X_1', X_0'', X_1'')$ by an appropriate factor $\rho \in (0,1)$, e.g.,

$$\rho\big(X_0', X_1', X_0'', X_1''\big),$$

which will always lead to monotonicity for ρ small enough. To see this, note that, after scaling, the coefficients in (19) become

$$\alpha = \frac{4(B-D)}{A^{1/4}B^{3/4}}, \qquad \beta = \frac{6\left(D-C-2B-2A+\frac{5}{\rho}\right)}{A^{1/2}B^{1/2}}, \qquad \gamma = \frac{4(A+C)}{A^{3/4}B^{1/4}},$$

and decreasing ρ increases β without affecting α or γ. The nature of the positivity regions is such that increasing β will eventually satisfy the conditions of Theorem 2, for fixed α and γ. In particular, equation (16ii) can be used to provide an approximate value for ρ, e.g., choose ρ as

$$\rho \approx \frac{120}{\sqrt{AB}(\delta^2+8) + 24(2A+2B+C-D)}, \quad \text{where } \delta = \min\{\alpha, \gamma\}.$$

A more satisfying strategy for constraining (17) to be monotone would involve adjusting only the second derivatives. For much of what follows, we will assume that the values of A and B are fixed. If a point $\eta_0 = (C_0, D_0)$ could be located within the interior of the monotonicity region, for fixed (A, B), then (17) would be monotone for (A, B, C^*, D^*), where

$$(C^*, D^*) = \rho\,(C, D) + (1-\rho)(C_0, D_0) \tag{20}$$

for sufficiently small ρ.

With fixed (A, B), using (19) and plotting $\Delta = \Delta(C, D) = 0$ in the (C, D) plane gives teardrop-shaped regions for the permissible values of (C, D) [3], [11]. The discussion below, based on [11], derives entirely from translating the positivity region in (α, β, γ) space to a region in (A, B, C, D) space.

We establish the notation $\eta = (C, D)$ for fixed (A, B), and consider the points (determined in conjunction with a previous paper [11] concerned with monotonicity of the quintic Hermite polynomial) defined by

$$
\begin{aligned}
\eta_1 = (C_1, D_1) = \Bigg(& \frac{-\sqrt{A}}{4}\left(7\sqrt{A} + 3\sqrt{B} + \sqrt{5\left(24 + 2\sqrt{AB} - 3(A+B)\right)}\right), \\
& \frac{\sqrt{B}}{4}\left(3\sqrt{A} + 7\sqrt{B} + \sqrt{5\left(24 + 2\sqrt{AB} - 3(A+B)\right)}\right)\Bigg), \\
\eta_2 = (C_2, D_2) = \Bigg(& \frac{-\sqrt{A}\left(6A + 3B - 15 + 2\sqrt{AB} - 4A^{3/4}B^{1/4}\right)}{3(\sqrt{A} + \sqrt{B}) - 4(AB)^{1/4}}, \\
& \frac{\sqrt{B}\left(3A + 6B - 15 + 2\sqrt{AB} - 4A^{1/4}B^{3/4}\right)}{3(\sqrt{A} + \sqrt{B}) - 4(AB)^{1/4}}\Bigg), \\
\eta_3 = (C_3, D_3) = \Bigg(& \frac{-\sqrt{A}}{4}\left(7\sqrt{A} + 3\sqrt{B} - \sqrt{5\left(24 + 2\sqrt{AB} - 3(A+B)\right)}\right), \\
& \frac{\sqrt{B}}{4}\left(3\sqrt{A} + 7\sqrt{B} - \sqrt{5\left(24 + 2\sqrt{AB} - 3(A+B)\right)}\right)\Bigg).
\end{aligned}
\tag{21}
$$

These points lie along the line $(\gamma = \alpha)$

$$
\sqrt{B}(A + C) = \sqrt{A}(B - D), \tag{22}
$$

and we will show that this line passes through the teardrop-shaped region of (C, D) values for which equation (17) is monotone, for fixed (A, B). Also consider Regions I and II, depicted in Fig. 6, showing values of A and B (both positive) over which the η_i will provide endpoints bounding the segment of (C, D) values along (22) for which the Hermite interpolant is monotone. Note that the Region II boundary is the same as the one defined in Fig. 4.2 of [3]. If we define $\alpha(\nu)$, $\beta(\nu)$, $\gamma(\nu)$ to be the coefficients in (19) evaluated at $\nu = \eta_1$, η_2, or η_3, then we can prove the following results:

THEOREM 3. *The coefficients in* (19) *exhibit the following properties*:

i. $\eta_2 = \eta_3$ *along the boundary between Regions* I *and* II;

ii. $\beta(\eta_1) > 6$ *over Regions* I *and* II;

iii. $\beta(\eta_2) \leq 6$ *over Region* I *and* $\beta(\eta_2) > 6$ *over Region* II;

iv. $\beta(\eta_3) \leq 6$ *over Region* I *and* $\beta(\eta_3) > 6$ *over Region* II;

v. $\alpha(\eta_1) = \gamma(\eta_1) = -2\sqrt{\beta(\eta_1) - 2}$;

vi. $\alpha(\eta_2) = \gamma(\eta_2) = -\dfrac{\beta(\eta_2) + 2}{2}$;

vii. $\alpha(\eta_3) = \gamma(\eta_3) = -2\sqrt{\beta(\eta_3) - 2}$.

The proof of these results involves tedious, but straightforward, algebra and therefore will not be given here. The points η_1, η_2, η_3 provide the endpoints to intervals

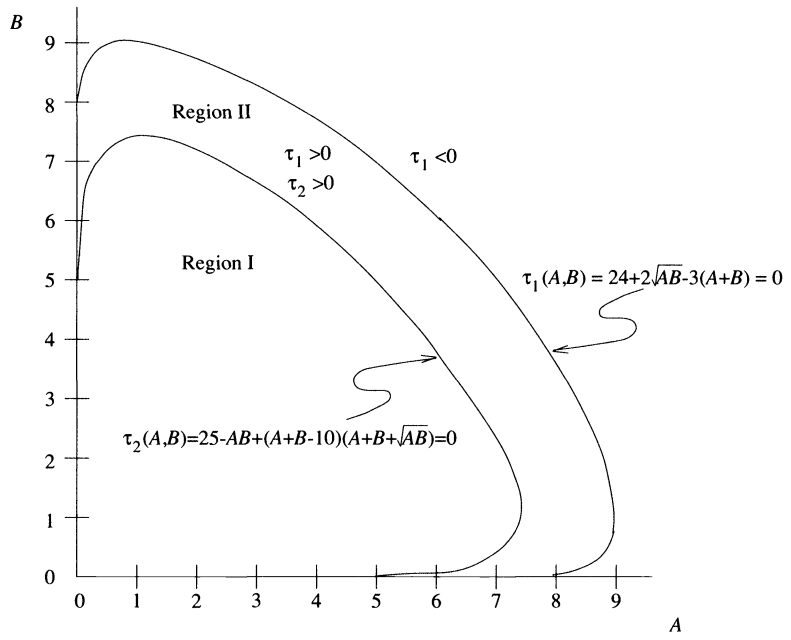


FIG. 6. *Boundary regions on* $\eta(A,B)$.

of monotonicity for η along the line given by equation (22). Note that η_1 and η_3 are complex for (A, B) beyond Region II, a reflection of the fact that equation (17) cannot be monotone for (A, B) outside of Regions I and II. The endpoint relationships can be summarized in the following theorem:

THEOREM 4. *Let* (A, B) *be fixed and contained within Region* I *or* II. *If* $\eta_0 = (C_0, D_0)$ *is any point on the line given by* (22), *then* (17) *is monotone for* (A, B, C_0, D_0) *if and only if*

i. (A, B) *is in Region* I *and* $\eta_0 = \rho\eta_1 + (1-\rho)\eta_2$ *for some* $\rho \in [0, 1]$, *or*
ii. (A, B) *is in Region* II *and* $\eta_0 = \rho\eta_1 + (1-\rho)\eta_3$ *for some* $\rho \in [0, 1]$.

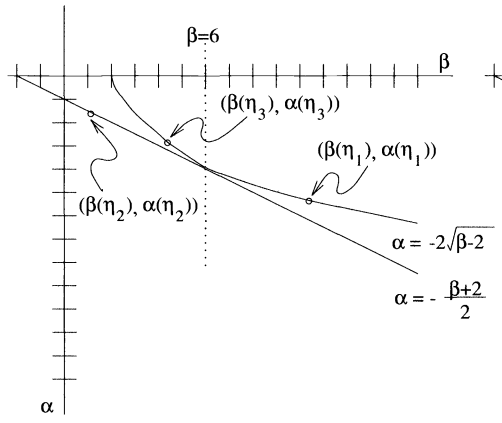
Proof. Sufficiency. Note first that $\alpha(\eta_0) = \gamma(\eta_0)$ since η_0 lies along the line given by (22). Consider first the case of (A, B) in Region II. According to (16), we need to show that $\alpha(\eta_0) \geq -2\sqrt{\beta(\eta_0) - 2}$. Clearly, for fixed (A, B), $\alpha(\eta)$ and $\beta(\eta)$ are affine in η so that (using Theorem 3)

$$\begin{aligned}\alpha(\eta_0) &= \rho\alpha(\eta_1) + (1-\rho)\alpha(\eta_3) = \rho\left(-2\sqrt{\beta(\eta_1) - 2}\right) + (1-\rho)\left(-2\sqrt{\beta(\eta_3) - 2}\right) \\ &\geq -2\sqrt{\left(\rho\beta(\eta_1) + (1-\rho)\beta(\eta_3)\right) - 2} \quad \text{by convexity of } -2\sqrt{x} \\ &= -2\sqrt{\beta(\eta_0) - 2}.\end{aligned}$$

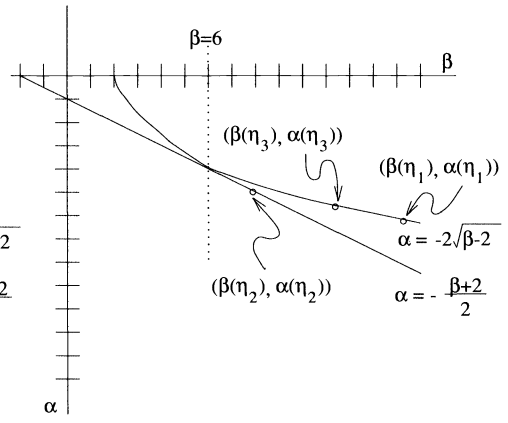
Figure 7b depicts the relationship established in the above inequality.

For the case of (A, B) in Region I, the proof is similar except that we have to consider the convex function defined by

$$\omega(x) = \begin{cases} -\dfrac{x+2}{2}, & x \leq 6; \\ -2\sqrt{x-2}, & x > 6. \end{cases}$$



(7a) (*A*,*B*) in region I.



(7b) (*A*,*B*) in region II.

FIG. 7. *Coefficients evaluated at η values.*

Again using Theorem 3 and (16), as Fig. 7a suggests, we have

$$\begin{aligned}\alpha(\eta_0) &= \rho\alpha(\eta_1) + (1-\rho)\alpha(\eta_2) = \rho\big(-2\sqrt{\beta(\eta_1)-2}\big) + (1-\rho)\left(-\frac{\beta(\eta_2)+2}{2}\right)\\ &= \rho\,\omega(\beta(\eta_1)) + (1-\rho)\omega(\beta(\eta_2))\\ &\geq \omega\big(\rho\beta(\eta_1) + (1-\rho)\beta(\eta_2)\big) \quad \text{by convexity of } \omega(x)\\ &= \omega(\beta(\eta_0)).\end{aligned}$$

Necessity. From Figs. 1 and 3 and Theorem 2, both the situations

$$\alpha(\eta_0) = \gamma(\eta_0) < -\frac{\beta(\eta_0)+2}{2}, \qquad \big(\beta(\eta_0) \leq 6\big),$$

and

$$\alpha(\eta_0) = \gamma(\eta_0) < -2\sqrt{\beta(\eta_0)-2}, \qquad \big(\beta(\eta_0) > 6\big),$$

correspond to nonmonotonicity of (17). These situations obtain precisely when $\rho \notin [0,1]$ for Region I (Fig. 7a) or $\rho \notin [0,1]$ for Region II (Fig. 7b), from the convexity of $\omega(x)$. This proves the necessity and completes the proof. □

By choosing η_0 as any point satisfying the conditions of Theorem 4, we obtain a point interior to the region of monotonicity. The adjustment described in (20) can then be used to constrain the quintic to be monotone. For (A,B) in Region II, a reasonable choice of η_0 is the average of the interval endpoints, η_1 and η_3, namely

$$\eta_0 = \left(-\frac{\sqrt{A}}{4}(7\sqrt{A}+3\sqrt{B}), \frac{\sqrt{B}}{4}(3\sqrt{A}+7\sqrt{B})\right). \tag{23}$$

Actually, this same point is interior to the monotonicity region for (A,B) in Region I as well since η_3 lies between η_1 and η_2 within Region I along the line given in (22).

We will summarize all of these results in algorithmic form. If $\nu = X_1 - X_0 < 0$, monotonically increasing is impossible, and if $\nu = 0$, the solution is $A = B = C = D = 0$. Assume $\nu > 0$ in the following, and that nominal values of A, B, C, D are given (obtained, e.g., by a C^4 quintic spline interpolant or finite difference approximations). The case where (A, B) lies outside of Regions I and II will be handled in the first and second steps of the algorithm.

Algorithm for constructing a monotone increasing Hermite quintic.

1. Set $A := \max\{0, A\}$, $B := \max\{0, B\}$. If $AB = 0$, use the simpler positivity criteria of [8] for cubics.
2. If $\tau_1(A, B) = 24 + 2\sqrt{AB} - 3(A+B) < 0$, then scale the derivative vector (X_0', X_1') until $\tau_1(A, B) > 0$.
3. Let $\eta = (C, D)$. If $\alpha(\eta)$, $\beta(\eta)$, and $\gamma(\eta)$ do not satisfy the conditions of Theorem 2, then find ρ such that $\alpha(\eta^*)$, $\beta(\eta^*)$, and $\gamma(\eta^*)$ do satisfy the conditions, where

$$\eta^* = \rho\eta + (1 - \rho)\eta_0, \qquad \rho \in (0, 1),$$

with η_0 defined in (23).

Following Huynh [5], Step 1 can be preceded by: Set $X_0' := \text{median}\left\{0, X_0', \frac{7\nu}{2h}\right\}$, $X_1' := \text{median}\left\{0, X_1', \frac{7\nu}{2h}\right\}$, and then compute (A, B). Step 2 can be replaced by a simpler sufficient condition which would scale the derivative vector till the first derivatives lie within the square imbedded within Region II (e.g., the square defined by $[0, 6] \times [0, 6]$). Yet another alternative to Step 2 would be to scale the derivatives X_0', X_1' back independently until $\tau_1(A, B) > 0$ [2], [3]. In a similar fashion, we can use the conditions in (16) to simplify verification of monotonicity of the quintic in Step 3 of the algorithm.

Note that the above algorithm is for a *single* monotone increasing Hermite quintic polynomial piece, and does not address at all the iterative adjustments required to achieve a monotone Hermite quintic *spline*. From Theorem 4 and (21) it is clear, however, that such an adjustment is always possible: for sufficiently small $(A, B) > 0$ the admissible (C, D) line segments are arbitrarily long. An efficient and robust computer algorithm for monotone Hermite quintic spline interpolation based on the teardrop regions of [11] will be the topic of a future paper.

5. Conclusion. Elegant necessary and sufficient conditions that a quartic polynomial $f(z)$ be nonnegative for $z > 0$ have been derived. Simple and computationally cheap sufficient conditions were deduced from the more general conditions. Important applications are to monotone quintic spline interpolation and the efficient generation of random variates from arbitrary continuous distributions. Applying the theory to monotone quintic interpolation, sharp necessary and sufficient conditions for monotonicity were derived. Simpler sufficient conditions for monotonicity were condensed into an algorithm suitable for quintic spline interpolation.

REFERENCES

[1] L. Brickman and L. Steinberg, *On nonnegative polynomials*, Amer. Math. Monthly, 69 (1962), pp. 218–221.

[2] C. DEBOOR AND B. SWARTZ, *Piecewise monotone interpolation*, J. Approx. Theory, 21 (1977), pp. 411–416.

[3] R. L. DOUGHERTY, A. EDELMAN, AND J. M. HYMAN, *Nonnegativity-, monotonicity-, or convexity-preserving cubic and quintic Hermite interpolation*, Math. Comp., 52 (1989), pp. 471–494.

[4] F. N. FRITSCH AND R. E. CARLSON, *Monotone piecewise cubic interpolation*, SIAM J. Numer. Anal., 17 (1980), pp. 238–246.

[5] H. T. HUYNH, *Accurate monotone cubic interpolation*, National Aeronautical and Space Administration Tech. Mem. 103789, Lewis Research Center, Cleveland, OH, 1991.

[6] E. I. JURY AND M. MANSOUR, *Positivity and nonnegativity of a quartic equation and related problems*, IEEE Trans. Automat. Control, 26 (1981), pp. 444–451.

[7] S. NEUMARK, *Solution of Cubic and Quartic Equations*, Pergamon, New York, 1965.

[8] J. W. SCHMIDT AND W. HEß, *Positivity of cubic polynomials on intervals and positive spline interpolation*, BIT, 28 (1988), pp. 340–352.

[9] G. SZEGÖ, *Orthogonal Polynomials*, Amer. Math. Soc. Colloq. Publ., Vol. 23, Providence, RI, 1939.

[10] G. ULRICH AND L. T. WATSON, *A method for computer generation of variates from arbitrary continuous distributions*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 185–197.

[11] ———, *Using piecewise polynomials to generate from arbitrary continuous distributions*, Computer Science and Statistics: Proceedings of the 19th Symposium on the Interface, Philadelphia, PA, 1987, pp. 450–453.

[12] S. WOLFRAM, *Mathematica: A System for Doing Mathematics by Computer*, Addison-Wesley, Redwood City, CA, 1991.

SPECIAL SECTION ON ITERATIVE METHODS IN NUMERICAL LINEAR ALGEBRA

The 1992 Copper Mountain Conference on Iterative Methods in Numerical Linear Algebra was the seventh conference in the Copper Mountain series and the second devoted to general iterative methods. It was attended by approximately 190 mathematicians from all corners of the world. The meeting, held April 9–14, 1992, took place at the Copper Mountain Resort, which is located 70 miles west of Denver, in the heart of Colorado's famous Summit County ski region. The setting of the conference was a cluster of buildings nestled at the base of the resort's ski hill. Morning and evening sessions were scheduled, leaving afternoons open for informal discussions and recreation.

During the five days of the meeting, 109 talks on current research were presented. These talks were organized by content and grouped into sessions. Session topics included the Theory of Iterative Methods (two sessions), Nonsymmetric Systems (four), Preconditioning Strategies (two), Parallel Implementations (three), Applications (four), Multigrid and Multilevel Methods (three), Domain Decomposition (two), Eigenvalue Problems (two), Integral Equations, Nonlinear Systems (two), Indefinite and Complex Matrix Problems, Collocation Matrices, and Software. In addition, there were workshops devoted to Scientific Computing in C^{++} and Common Software Standards.

On the first day of the conference each participant was provided with a two-volume set of Preliminary Proceedings, which contained either a short paper or an abstract by each of the scheduled speakers. Of those, 49 speakers submitted complete papers to this special issue of the *SIAM Journal on Scientific Computing* (*SISC*). This and the previous issue of *SISC* contain the results of the refereeing process. They represent a rich mix of papers on a wide variety of topics related to iterative methods.

In the previous issue the last twelve papers represent the conference. The first three deal with general iterative algorithms. The second and third involve quasi-minimal residual algorithms, a subject that inspired much discussion at the meeting. Papers 4 and 5 analyze interesting theoretical aspects of iterative methods. Papers 6 through 9 deal with precondition strategies. Papers 10 through 12 explore the implications for iterative methods of parallel computing environments.

In this issue the last twelve papers again represent the conference. The first seven of these deal with multigrid or multilevel algorithms. The first of these is a very interesting paper that uses the tools of classical iterative methods to analyze and compare many well-known algorithms. The second and third papers also contribute to the development of the theory of multigrid and multilevel algorithms. Papers 4 through 6 are concerned with domain decomposition methods, which, incidentally, fall into the structure developed in the first paper. The final six papers in this group discuss iterative methods in the context of specific applications and represent the many fine application-specific talks presented at the conference.

A special effort was made to bring students to the meeting. The vehicle for this effort was a Student Paper Competition in which students were asked to submit an original research paper consisting primarily of their own work. We were fortunate enough to be able to provide all student authors participating in this competition with free lodging and registration. Out of thirteen submissions, three winners were selected. First place went to Xian-Zhong Guo from the University of Maryland for his work, *"The Algebraic Hierarchical Basis Multigrid Method."* Second place was awarded to Marlis Hochbruck from the University of Karlsruhe, Germany, for her paper, *"On the Use of Two QMR Algorithms for Solving Singular Systems and Applications in the Markov Chain Modeling."* Third place honors were awarded to Andrew Lumsdaine from MIT for the paper, *"Accelerating Dynamic Iteration Methods with*

Application to Semiconductor Device Simulation." All winners were awarded a travel stipend, and their papers were presented at a special session devoted to the Student Paper Competition.

We would like to thank the following members of the program committee for their help in editing this special issue. They are: Steve Ashby, Howard Elman, Roland Freund, Anne Greenbaum, Seymour Parter, Paul Saylor, Homer Walker, and Olof Widlund. Through their efforts, the articles contained in this special issue were carefully refereed and brought to print on schedule.

We would also 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.

As this issue goes to press, planning for the next conference in this series is in its final stages. It will be called the Colorado Conference on Iterative Methods and will be held April 4–9, 1994, in Breckenridge, Colorado. Plans again include a special journal issue in *SISC*. It is our hope that the lively interaction and the fine quality of presentations and papers that have marked the previous meetings can be duplicated at the upcoming meeting.

Conference Chairmen
Tom Manteuffel
Steve McCormick

Special Issue Editors
Steve Ashby
Howard Elman
Roland Freund
Anne Greenbaum
Seymour Parter
Paul Saylor
Homer Walker
Olof Widlund

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 547–565, May 1994

004

MULTILEVEL ALGORITHMS CONSIDERED AS ITERATIVE METHODS ON SEMIDEFINITE SYSTEMS*

MICHAEL GRIEBEL†

Abstract. For the representation of piecewise d-linear functions instead of the usual finite element basis, a generating system is introduced that contains the nodal basis functions of the finest level and of all coarser levels of discretization. This approach enables the author to work directly with multilevel decompositions of a function.

For a partial differential equation, the Galerkin scheme based on this generating system results in a semidefinite matrix equation that has in the one-dimensional (1D) case only about twice, in the two-dimensional (2D) case about 4/3 times, and in the three-dimensional (3D) case about 8/7 times as many unknowns as the usual system. Furthermore, the semidefinite system possesses not just one, but many solutions. However, the unique solution of the usual definite finite element problem can be easily computed from every solution of the semidefinite problem.

The author shows that modern multilevel algorithms can be considered as standard iterative methods over the semidefinite system. The conjugate gradient (CG) method for the semidefinite system is equivalent to the BPX- or MDS-preconditioned CG method for the linear system that arises from the usual finite element basis. The Gauss–Seidel iteration applied to the semidefinite system is equivalent to the multigrid (MG) method applied to the standard basis system. Consequently, the Gauss–Seidel-preconditioned CG method for the semidefinite system is equivalent to MG-CG for the standard basis system.

This paper includes results of numerical experiments regarding the condition number and the convergence rates of different iterative methods for the semidefinite system.

Key words. partial differential equations, multilevel methods, BPX-preconditioner, conjugate gradients, multigrid methods, Gauss–Seidel iteration, semidefinite system

AMS subject classifications. 65F10, 65N20, 65N30, 65N55, 65N99

1. Introduction.

In recent years, different multilevel methods have been used extensively to obtain approximations to solutions of partial differential equations. Besides the multigrid method (see the references in [10], [13]) and the hierarchical basis multigrid method [3], multilevel methods for the preconditioning of the CG algorithm have also been developed, including the BPX-preconditioner of Bramble, Pasciak, and Xu [5], [21], its generalization multilevel diagonal scaling (MDS) (see [20], [25]), and the hierarchical basis preconditioner of Yserentant [22]. For a comparison of these preconditioners, see [23]. Another interesting type of multilevel preconditioner has been developed in [2]. Furthermore, preconditioners that make use of the multigrid method have been studied in [11].

Under rigorous assumptions on regularity and approximation properties, a convergence rate that is independent of the number of levels can be proved for multigrid methods. In contrast to this, two-level schemes have been proved to converge under weaker assumptions. The same is true for the preconditioner of Xu and, in the 2D case, of Yserentant. Here, no regularity assumptions are needed to prove a condition number of $O(k^2)$ for the preconditioned system, where k denotes the number of employed levels. Under strong regularity assumptions, a condition number estimate of $O(k)$ can be given. See the Appendix in [23].

Recently, Oswald [16]–[18], Xu [20], and Bornemann and Yserentant [4] showed that (even under weak assumptions) the condition number of the BPX-preconditioner is $O(1)$. Also, Zhang [25] gave an elegant proof for the condition number of the MDS operator, which is a slight generalization of BPX. It grows at most linearly with the number of levels in the

*Received by the editors April 30, 1992; accepted for publication (in revised form) December 8, 1992. This research was partially supported by Deutsche Forschungsgemeinschaft-Sonderforschungsbereich SFB 0342 and Deutsche Forschungsgemeinschaft-AZ 477/766/92.

†Institut für Informatik, Technische Universität München, Arcisstraße 21, D-80290 München, Germany (griebel@informatik.tu-muenchen.de).

general case, and is bounded by a constant independent of the mesh sizes and number of levels under the H^2-regularity assumption. In this sense, multilevel methods are optimal.

In this paper we present a new approach to multilevel methods that abandons the usual basis approach in favor of generating functions that span the approximation space, but which are generally linearly dependent. The idea is to use the standard basis functions on *all* levels of discretization. The resulting Galerkin scheme therefore results in a redundant semidefinite system of linear equations that has multiple solutions, but each is equivalent to the unique solution of the standard basis approach. Fortunately, in many cases, the generalized condition number (i.e., the condition number of the matrix restricted to the orthogonal complement of its null space) of the semidefinite system matrix is of the order $O(1)$, and we will see that its singularity does not disrupt the iterative process that we use.

More specifically, there is no need for a basis of the approximation space if energy minimizing iteration methods like the CG method or Gauss–Seidel-type relaxations are used. In this case, even the use of a basis is an obstacle because smooth errors cannot be adequately attenuated by the standard basis functions. Acceleration of these methods can be gained only by quite complex techniques such as BPX-preconditioning or, in the multigrid context, by coarse grid corrections.

In contrast, the use of the generating system allows us to work directly with multilevel decompositions of functions. This gives additional freedom that is beyond the scope of the traditional basis approach and leads in a natural manner to accelerated algorithms. It is of special interest that standard iteration methods for the semidefinite system are equivalent to modern multilevel methods applied to the linear system that arises from the usual finite element basis.

We show that the BPX-preconditioner and its generalization MDS, together with the CG method, are just special implementations of the (diagonally scaled) CG iteration on the semidefinite system. Similarly, the HB-preconditioner can be interpreted in terms of the semidefinite system. Furthermore, we will show that the Gauss–Seidel iteration for the semidefinite system is equivalent to the multigrid method for the standard basis system. For Aitken's double sweep algorithm, which is also known as symmetric Gauss–Seidel (SGS) we obtain the standard (1,1)-MG-V-cycle. In this sense, our method and McCormick's approach of Unigrid and PML (multilevel projection method) are similar in spirit; see [14], [15]. Additionally, the HB-MG method can be explained easily in terms of the semidefinite system. If we use the Gauss–Seidel iteration as a CG-preconditioner for the semidefinite system, we obtain an algorithm that is equivalent to MG-CG for the standard basis system.

The outline of the remainder of this paper is as follows. In §2, we introduce the generating system, derive the semidefinite system of linear equations, and discuss its properties. In §3, we consider iterative methods on the semidefinite system. First, we study the diagonally preconditioned CG method. We show that it is equivalent to the BPX- or MDS-preconditioned CG method for the standard basis system. Then we demonstrate that the Gauss–Seidel iteration for the semidefinite system is equivalent to the multigrid method for the standard basis system. Furthermore, we show that the use of Gauss–Seidel relaxations as a preconditioner for the semidefinite system results in MG-CG for the standard basis system. Finally, the results of numerical experiments illustrating the theory of the earlier sections are given in §4. An extended version of this paper can be found in [7].

2. The semidefinite system. In this section, we introduce a generating system that replaces the usual finite element basis in the discretization of a boundary value problem of a partial differential equation. We then derive the associated semidefinite linear system and discuss its properties.

Consider a partial differential equation in d dimensions with a linear, second-order oper-

ator on the domain $\Omega = (0, 1)^d$, $d = 1, 2, \ldots$,

$$Lu = f \quad \text{on} \quad \Omega, \tag{1}$$

with appropriate boundary conditions and solution u. For reasons of simplicity, we restrict ourselves to homogeneous Dirichlet boundary conditions. Given an appropriate function space V, the problem can be expressed equivalently in its variational form: Find a function $u \in V$ with

$$a(u, v) = (f, v) \quad \forall\, v \in V. \tag{2}$$

(In the case of homogeneous Dirichlet boundary conditions, V would be the Sobolev space $H_0^1(\Omega)$.) Here, a: $V \times V \to V$ is a bounded, positive-definite, symmetric bilinear form and $(.\,,.)$ is the usual linear form for the right-hand side. Let $||.||_a := \sqrt{a(.\,,.)}$ denote the induced energy norm. We assume that V is complete with respect to $||.||_a$, which is true if $a(.\,,.)$ is H_0^1-elliptic. The Lax–Milgram lemma then guarantees the existence and uniqueness of the solution of (2). If we consider directly the energy functional $1/2 \cdot a(u, u) - (f, u)$, the problem can be stated alternatively as minimization of the energy in V.

2.1. Spaces, bases, and the generating system. Assume we are given a sequence of uniform, equidistant, nested grids,

$$\Omega_1 \subset \Omega_2 \subset \cdots \subset \Omega_{k-1} \subset \Omega_k, \tag{3}$$

on Ω with respective mesh sizes $h_l = 2^{-l}$, $l = 1, \ldots, k$, and an associated sequence of spaces of piecewise d-linear functions,

$$V_1 \subset V_2 \subset \cdots \subset V_{k-1} \subset V_k, \tag{4}$$

with dimensions

$$n_l := \dim(V_l) = (2^l - 1)^d, \qquad l = 1, \ldots, k. \tag{5}$$

Here, 1 denotes the coarsest and k the finest level of discretization. Consider also the sequence

$$N_1 \subset N_2 \subset \cdots \subset N_{k-1} \subset N_k \tag{6}$$

of sets of grid points $N_l = \{x_1, \ldots, x_{n_l}\}$ in the interior Ω_l, $l = 1, \ldots, k$.

The standard finite element basis that spans V_l on the equidistant grid Ω_l is denoted by B_l. It contains the nodal basis functions $\phi_i^{(l)}$, $i = 1, \ldots, n_l$, which are defined by

$$\phi_i^{(l)}(x_j) = \delta_{i,j}, \qquad x_j \in N_l. \tag{7}$$

Note that we use rectangular grids. Therefore, the 2D basis functions can be written as the product of two 1D basis functions for the different coordinate directions. The analogue is true for the 3D case, where the basis functions are written as the product of three 1D basis functions.

Any function $u \in V_k$ can be expressed uniquely by

$$u = \sum_{i=1}^{n_k} u_i^{(k)} \phi_i^{(k)}, \tag{8}$$

with the vector $u_k := (u_1^{(k)}, u_2^{(k)}, \ldots, u_{n_k}^{(k)})^T$ of nodal values.

In contrast to the basis approach expressed by the expansion in (8), we instead consider the set of functions E_k defined as the union of all the different nodal bases B_l for the levels $l = 1, \ldots, k$,

$$E_k = B_1 \cup B_2 \cup \cdots \cup B_{k-1} \cup B_k. \tag{9}$$

Clearly, because E_k is a linearly dependent set of functions, it is no longer a basis for V_k, but merely a generating system. See Fig. 1 for a simple 1D example.

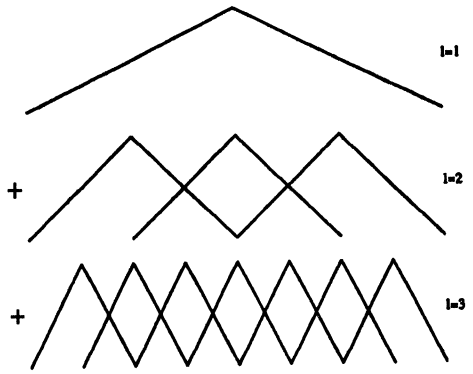


FIG. 1. *Functions of the generating system E_3 in the* 1D *case.*

In any case, an arbitrary function $u \in V_k$ can be expressed in terms of the generating system by

$$u = \sum_{l=1}^{k} \sum_{i=1}^{n_l} w_i^{(l)} \phi_i^{(l)}, \tag{10}$$

with the block vector $w_k^E := (w^{(1)}, w^{(2)}, \ldots, w^{(k)})^T$, were $w^{(l)} = (w_1^{(l)}, w_2^{(l)}, \ldots, w_{n_l}^{(l)})$, $l = 1, \ldots, k$. This represents a *level-wise decomposition* of u into functions $g^{(l)} \in V_l, l = 1, \ldots, k$, where

$$u = \sum_{l=1}^{k} g^{(l)}, \qquad g^{(l)} = \sum_{i=1}^{n_l} w_i^{(l)} \phi_i^{(l)}. \tag{11}$$

To simplify the notation, we will not distinguish explicitly between the functions u or $g^{(l)}$ and their vector representations u_k or $w^{(l)}$, but denote them by their vector representations only. Here and in the following, we denote representations in terms of the generating system E_k by the superscript E. The length of w_k^E is

$$n_k^E = \sum_{l=1}^{k} n_l, \tag{12}$$

which is in the 1D case about twice, in the 2D case about 4/3 times, and in the 3D case about 8/7 times larger than the length of the vectors for the basis representation above. This is due to the geometric rate of decrease of the number of grid points from fine to coarse levels, with

factors of 1/2, 1/4, and 1/8 for 1D, 2D, and 3D, respectively. The generalization of this concept to higher dimensions is straightforward.

Note that the representation of u in terms of E_k is not unique. In general, there exists an n_{k-1}^E-dimensional variety of level-wise decompositions of $u \in V_k$. However, for a given representation w_k^E of u in E_k, we can easily compute its unique representation u_k with respect to B_k. This transformation from w_k^E to u_k resembles a mapping $S_k^E : \mathbb{R}^{n_k^E} \to \mathbb{R}^{n_k}$, which can be described by the product

$$S_k^E = S_k^{k-1,E} \cdot S_{k-1}^{k-2,E} \cdot S_{k-2}^{k-3,E} \cdot \ldots \cdot S_3^{2,E} \cdot S_2^{1,E} = \prod_{l=k}^{2} S_l^{l-1,E} \tag{13}$$

of sparse triangular $(n_k^E - n_{l-1}^E) \times (n_k^E - n_{l-2}^E)$-matrices $S_l^{l-1,E}$. We then have

$$u_k = S_k^E w_k^E. \tag{14}$$

Let P_{l-1}^l be the d-linear interpolation (prolongation) operator that maps V_{l-1} to V_l, $l = k, \ldots, 2$. Let R_l^{l-1} be the restriction operator from V_l to V_{l-1} defined as the adjoint of P_{l-1}^l, $R_l^{l-1} = (P_{l-1}^l)^T$, $l = k, \ldots, 2$. The transformation matrices $S_l^{l-1,E}$, $l = k, \ldots, 2$ can be expressed by means of these intergrid transfer operators as

$$S_l^{l-1,E} = \begin{pmatrix} P_{l-1}^l & I_l & 0 \\ 0 & 0 & I_{k,l+1} \end{pmatrix}. \tag{15}$$

$I_{k,l+1}$ is the identity operator of dimension $\sum_{i=k}^{l+1} n_i$. Due to the generating system approach, these transformation matrices $S_l^{l-1,E}$ cannot be inverted uniquely. Nevertheless, the transposed matrices $(S_l^{l-1,E})^T$, $l = k, \ldots, 2$, are defined by

$$(S_l^{l-1,E})^T = \begin{pmatrix} R_l^{l-1} & 0 \\ I_l & 0 \\ 0 & I_{k,l+1} \end{pmatrix}. \tag{16}$$

Note that the identity relation in the basis representation B_k defines an equivalence relation $\equiv$ in E_k that is different from the simple vector equivalence. Thus, we can identify two different but equivalent representations in E_k of a function in V_k:

$$w_k^{E,1} \equiv w_k^{E,2} \Leftrightarrow_{\text{def}} S_k^E w_k^{E,1} = S_k^E w_k^{E,2}. \tag{17}$$

We see that two different representations $w_k^{E,1}$ and $w_k^{E,2}$ of u_k differ only by a vector $v_k^E = w_k^{E,1} - w_k^{E,2}$ that is in $\mathrm{Ker}(S_k^E)$, the kernel of S_k^E, i.e., $S_k^E v_k^E = 0$.

2.2. Linear systems. Using the nodal basis B_k, the Galerkin approach leads to the linear system of equations $L_k u_k = f_k$, where $(L_k)_{(i,j)} = a(\phi_i^{(k)}, \phi_j^{(k)})$ and $(f_k)_{(j)} = (f, \phi_j^{(k)})$, $i, j = 1, \ldots, n_k$. Alternatively, we can use the discrete energy functional $1/2 \cdot u_k^T L_k u_k - u_k^T f_k$ to describe the problem as energy minimization in the space V_k or $\mathbb{R}^{n_k}$, respectively. For the generating system E_k, the Galerkin approach leads to the system of equations

$$L_k^E w_k^E = f_k^E, \tag{18}$$

where

$$\begin{aligned} (L_k^E)_{(i,j)}^{(l_1,l_2)} &= a(\phi_i^{(l_1)}, \phi_j^{(l_2)}), \\ (f_k^E)_{(j)}^{(l_2)} &= (f, \phi_j^{(l_2)}), \end{aligned} \qquad l_1, l_2 = 1, \ldots, k, \quad i = 1, \ldots, h_{l_1}, \quad j = 1, \ldots, h_{l_2}. \tag{19}$$

Alternatively, we can use the discrete energy functional $1/2 \cdot (w_k^E)^T L_k^E w_k^E - (w_k^E)^T f_k^E$ to describe the problem as (nonunique) minimization in $\mathbb{R}^{n_k^E}$.

The system $L_k^E w_k^E = f_k^E$ has the following properties. The matrix L_k^E is semidefinite. It has the same rank as L_k. Thus $n_{k-1}^E = n_k^E - \text{rank}(L_k)$ eigenvalues are zero. The system is solvable because the right-hand side is constructed in a consistent manner. However, it does not have just one unique solution, but rather numerous different solutions. For two different solutions $w_k^{E,1}$ and $w_k^{E,2}$,

$$S_k^E w_k^{E,1} = S_k^E w_k^{E,2} = u_k \tag{20}$$

holds, where u_k is the unique solution of the system $L_k u_k = f_k$. Therefore, it is sufficient to compute just one solution of the enlarged semidefinite system to obtain via S_k^E the unique solution of the system $L_k u_k = f_k$.

Note that the enlarged matrix L_k^E contains the submatrices L_l that arise from the use of the standard basis B_l, $l = 1, \ldots, k$. A similar property holds for the right-hand sides. The matrix that stems from the hierarchical basis discretization is also contained in L_k^E. For the definition of the hierarchical basis, the resulting linear system, and further explanations, see [22].

Instead of using the bilinear form $a(.,.)$ and the linear form $(f,.)$, which involves explicit integration on all levels $l = 1, \ldots, k$, we can express L_k^E and f_k^E in a simple product-type representation that involves only the intergrid transfer operators P_{l-1}^l and R_l^{l-1}, $l = k, \ldots, 2$ and the discretizations of L and f on the finest level. Since the transformation from E_k to B_k is expressed by $u_k = S_k^E w_k^E$, we get

$$L_k^E = (S_k^E)^T L_k S_k^E, \qquad f_k^E = (S_k^E)^T f_k. \tag{21}$$

This can be interpreted as "semidefinite" preconditioning of the system $L_k u_k = f_k$. It is then easy to see that $\frac{1}{2} \cdot (w_k^E)^T L_k^E w_k^E - (w_k^E)^T f_k^E = \frac{1}{2} \cdot u_k^T L_k u_k - u_k^T f_k$.

Note that the matrix L_k^E is relatively densely populated. In practical applications, however, it is not necessary to know the matrix explicitly. Often only a matrix vector multiplication $L_k^E w_k^E$ is needed, which allows the product type representation (21) of L_k^E to be exploited, so that this matrix vector multiplication can be computed in $O(n_k^E) = O(n_k)$ operations.

Note further that the restriction operator R_l^{l-1} in the mapping $(S_k^E)^T$ is consistent with the Galerkin approach. If, for example, f_k in space V_k is given as $f_k = \sum_{i=1}^{n_k} (f, \phi_i^{(k)}) \phi_i^{(k)}$, then

$$R_k^{k-1} f_k = \sum_{i=1}^{n_{k-1}} (f, \phi_i^{(k-1)}) \phi_i^{(k-1)}. \tag{22}$$

This is an important fact that should be exploited in practical computations. The explicit calculation of the integral $(f,.)$ can be avoided on the coarser levels $l < k$. The linear combination of the coefficient values of f_k according to the restriction stencil is sufficient to reproduce the integral on the next coarser level. Consequently, if f_k on level k has been computed, then f_k^E can be gained easily using $(S_k^E)^T$.

Consider, as an example for the product type representation (21), the case of only two spaces V_{k-1}, V_k. Since the standard basis representation is computed from $w_k^E = (w^{(k-1)}, w^{(k)})^T$ by

$$u_k = \begin{pmatrix} P_{k-1}^k & I_k \end{pmatrix} \begin{pmatrix} w^{(k-1)} \\ w^{(k)} \end{pmatrix}, \tag{23}$$

then the system $L_k^E w_k^E = f_k^E$ can be rewritten as

$$\begin{pmatrix} R_k^{k-1} \\ I_k \end{pmatrix} L_k \begin{pmatrix} P_{k-1}^k & I_k \end{pmatrix} w_k^E = \begin{pmatrix} R_k^{k-1} \\ I_k \end{pmatrix} f_k, \tag{24}$$

and we see that

$$L_k^E = \begin{pmatrix} R_k^{k-1} L_k P_{k-1}^k & R_k^{k-1} L_k \\ L_k P_{k-1}^k & L_k \end{pmatrix}, \qquad f_k^E = \begin{pmatrix} R_k^{k-1} f_k \\ f_k \end{pmatrix}. \tag{25}$$

Note that $R_k^{k-1} L_k P_{k-1}^k$ is just the Galerkin coarse grid matrix L_{k-1}.

From this two-level example, we see that the system $L_k u_k = f_k$ arising from the finite element discretization using the basis B_k can instead be written in terms of the generating system E_k by the following steps:

Apply

$$\begin{pmatrix} R_k^{k-1} \\ I_k \end{pmatrix}$$

to both sides of the equation and set $u_k = \begin{pmatrix} P_{k-1}^k & I_k \end{pmatrix} w_k^E$.

In the first step, new equations are added to the system that are constructed from the original equations by forming linear combinations with the weights of the MG-restriction stencil. The second relation defines the new unknowns and their relation to the original unknowns. A natural two-level decomposition w_k^E of u_k is defined implicitly, which is not unique but nevertheless consistent with the Galerkin approach.

3. Iterative methods for the semidefinite system. In this section, we study iterative methods for the solution of the semidefinite system. We show that the diagonally preconditioned CG method is equivalent to the BPX- or MDS-preconditioned CG approach for the standard system using B_k. The generalized condition number of L_k^E is the same as for the BPX- or MDS-preconditioned matrix L_k. Therefore, proofs for the condition of BPX and MDS are directly valid for the semidefinite system. Results of work by Oswald, Zhang, Xu, Bornemann, and Yserentant for BPX establish a condition number of the order $O(1)$ for L_k^E. Furthermore, we show that the Gauss–Seidel (GS) method and its counterpart SGS are equivalent to multigrid methods for the standard system. A Gauss–Seidel-type relaxation itself can be used as preconditioner for the semidefinite system. This results in MG-CG for the standard basis system.

We want to achieve a solution of the semidefinite system $L_k^E w_k^E = f_k^E$. Because L_k^E is not invertible, we cannot use a conventional direct solver. Instead, we can consider basic iterations $w_k^E := w_k^E + C_k^E(f_k^E - L_k^E w_k^E)$ with positive-definite (or semidefinite) matrices C_k^E. We focus on the CG-type accelerations of such iterations where C_k^E is a diagonal matrix preconditioner. Furthermore, we study Gauss–Seidel-type relaxations, both as an iteration itself (with C_k^E as inverse of the lower triangular part of L_k^E) and as a preconditioner for the CG method.

It is well known that these methods iteratively reduce the energy of the system. They differ in the employed descent directions. While for the CG method the descent directions are based on the residual, the Gauss–Seidel iteration descends along the coordinate axes. It is well known that CG is sensitive to scaling, and that its convergence rate depends on the condition number of the system. On the other hand, Gauss–Seidel relaxations are invariant under scaling of the system. See [12] for further explanations on descent methods.

3.1. CG iteration for the semidefinite system. Here we consider the (diagonally preconditioned) CG method for the semidefinite system. The BPX-preconditioner of Bramble, Pasciak, and Xu and its generalization MDS applied to the standard basis system can be described directly in terms of E_k with simple diagonal matrices C_k^E of full rank n_k^E. This is also true of the HB-preconditioner introduced by Yserentant. It can be described with a simple diagonal matrix C_k^E of deficient rank n_k. If the functions of the generating system are properly scaled, then the C_k^E for BPX or MDS becomes the identity, and the CG method for the semidefinite system $L_k^E w_k^E = f_k^E$ is directly equivalent to BPX-CG or MDS-CG for the definite system $L_k u_k = f_k$. Otherwise, this scaling is done explicitly by the respective matrix C_k^E.

3.1.1. The residual and the CG iteration. In the CG method, a major task is to compute the residual

$$r_k^E = f_k^E - L_k^E w_k^E \tag{26}$$

of the semidefinite system for a given w_k^E. Here, the product type representation (21) of L_k^E and f_k^E can be exploited. We get

$$r_k^E = (S_k^E)^T f_k - (S_k^E)^T L_k S_k^E w_k^E \tag{27}$$

and, with $u_k = S_k^E w_k^E$ and $r_k = f_k - L_k u_k$, we have

$$r_k^E = (S_k^E)^T (f_k - L_k u_k) = (S_k^E)^T r_k. \tag{28}$$

Consider as an example once more the two-level case (23)–(25). The residual of the semidefinite system now reads

$$r_k^E = \begin{pmatrix} R_k^{k-1} f_k \\ f_k \end{pmatrix} - \begin{pmatrix} R_k^{k-1} L_k P_{k-1}^k w^{(k-1)} + R_k^{k-1} L_k w^{(k)} \\ L_k P_{k-1}^k w^{(k-1)} + L_k w^{(k)} \end{pmatrix}. \tag{29}$$

If we use the product-type representation (21) of L_k^E and f_k^E, we are able to implement the computation of the residual efficiently. We have

$$r_k^E = \begin{pmatrix} R_k^{k-1} \\ I_k \end{pmatrix} f_k - \begin{pmatrix} R_k^{k-1} \\ I_k \end{pmatrix} L_k \begin{pmatrix} P_{k-1}^k & I_k \end{pmatrix} \begin{pmatrix} w^{(k-1)} \\ w^{(k)} \end{pmatrix}, \tag{30}$$

and the same result is obtained if we first compute $u_k = P_{k-1}^k w^{(k-1)} + w^{(k)}$, then the fine level residual $r_k = f_k - L_k u_k$, then finally

$$r_k^E = \begin{pmatrix} R_k^{k-1} r_k \\ r_k \end{pmatrix}. \tag{31}$$

This avoids computing the expressions individually on the coarse and fine levels, which differ only by the coarse grid term R_k^{k-1}.

Note that all representations w_k^E in E_k of a given function u_k have exactly the *same residual vector* r_k^E. Therefore, even if our generating system allows many nonunique representations for a function u_k, it is unique with respect to the residual. In fact,

$$w_k^{E,1} \equiv w_k^{E,2} \Rightarrow L_k^E w_k^{E,1} = L_k^E w_k^{E,2}. \tag{32}$$

In the following, let C_k^E be a diagonal matrix. The central step in the CG method for solving $L_k^E w_k^E = f_k^E$ is the computation of the residual and its scalar product. With C_k^E-preconditioning, we must compute $(r_k^E)^T C_k^E r_k^E$. Since $r_k^E = (S_k^E)^T r_k$ (see (28)), then

$$(r_k^E)^T C_k^E r_k^E = r_k^T S_k^E C_k^E (S_k^E)^T r_k. \tag{33}$$

Hence, the C_k^E-preconditioned CG algorithm for solving $L_k^E w_k^E = f_k^E$ is equivalent to the CG algorithm with preconditioner $S_k^E C_k^E (S_k^E)^T$ for solving $L_k u_k = f_k$.

3.1.2. BPX- and related preconditioners. In the following, we show that the BPX-, MDS-, and HB-preconditioners can be expressed by the operator $S_k^E C_k^E (S_k^E)^T$ for appropriate diagonal matrices C_k^E. In [23], the finest level system $L_k u_k = f_k$ is treated by CG-accelerated iterations with a general selfadjoint, positive-definite preconditioning operator $C_k : V_k \to V_k$. Specific choices for C_k for the BPX- and HB-preconditioners were given by the respective matrices C_k^X and C_k^Y defined as follows:

$$C_k^X r_k = L_0^{-1} \sum_{i=1}^{n_0} (r_k, \phi_i^{(0)}) \phi_i^{(0)} + \sum_{l=1}^{k} \sum_{i=1}^{n_l} \frac{1}{d_i^{(l)}} (r_k, \phi_i^{(l)}) \phi_i^{(l)} \qquad \text{(BPX)}, \tag{34}$$

$$C_k^Y r_k = L_0^{-1} \sum_{i=1}^{n_0} (r_k, \phi_i^{(0)}) \phi_i^{(0)} + \sum_{l=1}^{k} \sum_{i=n_{l-1}+1}^{n_l} \frac{1}{d_i^{(l)}} (r_k, \phi_i^{(l)}) \phi_i^{(l)} \qquad \text{(HB)} \tag{35}$$

with the scaling factors $d_i^{(l)} = 4^l (1, \phi_i^{(l)})$. Yserentant suggested replacing the scaling factors $d_i^{(l)}$ by $\hat{d}_i^{(l)} = a(\phi_i^{(l)}, \phi_i^{(l)})$, which leads to the preconditioner C_k^Z defined as

$$C_k^Z r_k = L_0^{-1} \sum_{i=1}^{n_0} (r_k, \phi_i^{(0)}) \phi_i^{(0)} + \sum_{l=1}^{k} \sum_{i=1}^{n_l} \frac{1}{\hat{d}_i^{(l)}} (r_k, \phi_i^{(l)}) \phi_i^{(l)} \qquad \text{(MDS)}. \tag{36}$$

C_k^Z is further analyzed by Zhang in [25] as a multilevel diagonal scaling variant of the multilevel additive Schwarz method. This preconditioner can be expected to work in practice better than BPX for nonmodel problems since, with $\hat{d}_i^{(l)}$, it takes information into account that reflects more closely the properties of the problem. However, for elliptic problems with constant coefficients, MDS is equivalent to BPX up to a constant multiple.

Note that for our simplified case of Dirichlet boundary conditions, $V_0 = \{0\}$ and the term $L_0^{-1} \sum_{i=1}^{n_0} (r_k, \phi_i^{(0)}) \phi_i^{(0)}$ vanishes. For more general boundary conditions, a system with the level 0 discretization matrix L_0 must be solved here.

These preconditioners can now be expressed easily in terms of our semidefinite system $L_k^E w_k^E = f_k^E$. Computation of the residual $r_k^E = f_k^E - L_k^E w_k^E$ produces

$$r_k^E = (S_k^E)^T r_k = \begin{pmatrix} r^{(1)^T} & r^{(2)^T} & \dots & r^{(k-1)^T} & r^{(k)^T} \end{pmatrix}^T, \tag{37}$$

where

$$(r^{(l)})_i = (r_k, \phi_i^{(l)}) \phi_i^{(l)}, \quad l = k, \dots, 1, \quad i = 1, \dots, n_l. \tag{38}$$

The BPX- and HB- preconditioners resemble just a multiplication of the residual r_k^E of the enlarged system with the matrix C_k^E. For all preconditioners introduced above, this matrix C_k^E has diagonal form. The preconditioners differ only in the diagonal entries. Therefore, in terms of the enlarged system, BPX-, MDS-, or HB-preconditioning amounts to nothing but diagonal scaling of the residual.

For the BPX-preconditioner C_k^X, its analogue $C_k^{E,X}$ has full rank. In fact

$$C_k^{E,X} = \operatorname{diag}\left(\frac{1}{d_i^{(l)}}\right). \tag{39}$$

If the functions of our generating system use a special scaling, then $C_k^{E,X} = I_k^E$, and the BPX-preconditioner yields the usual CG method on the semidefinite system. This special scaling is expressed by

$$\bar{\phi}_i^{(l)} = h_l^{(2-d)/2}\phi_i^{(l)}, \quad l = k, \ldots, 1, \quad d = 1, 2, \ldots. \tag{40}$$

Note that in the 2D case no scaling is actually necessary.

Since the modified scaling factors $\hat{d}_i^{(l)} = a(\phi_i^{(l)}, \phi_i^{(l)})$ are used in the MDS scheme, we see that

$$C_k^{E,Z} = \operatorname{diag}\left(1/(L_k^E)_{(i,i)}^{(l,l)}\right). \tag{41}$$

Here, the MDS-preconditioner reduces to just a Jacobi-preconditioner for the semidefinite system (Jacobi-CG). If the functions of our generating system use a special scaling, then $C_k^{E,Z} = I_k^E$, and the MDS-preconditioner yields the usual CG method on the semidefinite system. This special scaling is expressed by

$$\hat{\phi}_i^{(l)} = a(\phi_i^{(l)}, \phi_i^{(l)})^{-1/2}\phi_i^{(l)}, \qquad l = k, \ldots, 1. \tag{42}$$

Note that Jacobi iteration on the semidefinite system alone does not generally converge, nor do the simple iterations based on BPX- or MDS-preconditioners. It is the combination of CG and preconditioning that makes the method properly (and quickly) converge.

For the HB-preconditioner C_k^Y, its analogue $C_k^{E,Y}$ has deficient rank. Only the n_k diagonal entries associated with the functions appearing in the hierarchical basis are nonzero. (They contain the inverse of the scaling factors $d_i^{(l)}$.) For further details, see [7] and [22].

If we want to express the resulting vector $C_k^{E,J} r_k^E$, $J \in \{X, Y, Z\}$ in terms of the basis B_k, we must multiply it by S_k^E, which amounts to the explicit level-wise summation of the BXP-, MDS-, or HB-preconditioner. However, using the generating system approach, this summation is avoided because of the direct, multilevel representation of the functions of V_k, implying that the summation is really done implicitly.

It is now possible to characterize the BPX-, MDS-, and HB-preconditioning in terms of B_k by the matrix notation

$$C_k^J = S_k^E C_k^{E,J} (S_k^E)^T, \qquad J \in \{X, Y, Z\}. \tag{43}$$

It is easy to see that the scalar products involved in the preconditioned CG methods for $L_k u_k = f_k$ and for $L_k^E w_k^E = f_k^E$ are the same. Since $r_k^E = (S_k^E)^T r_k$, then

$$(r_k)^T C_k^J r_k = (r_k^E)^T C_k^{E,J} r_k^E, \qquad J \in \{X, Y, Z\}. \tag{44}$$

We have seen that the use of the generating system, instead of the standard basis approach for the representation of the functions of V_k, allows an easy and simple description of the BPX-, MDS-, and HB-preconditioned CG method. The somewhat complicated form of the BPX-, MDS-, and HB-preconditioning reduces in E_k to simple diagonal scaling of the residual r_k^E. If we choose specially scaled generating functions in E_k, then C_k^E reduces to the identity matrix, which means that only the standard CG algorithm is performed on the associated semidefinite system.

3.1.3. Condition number considerations. Here we consider the condition number of the C_k^E-preconditioned semidefinite system. Of course, in general, the conventional condition number is infinity because L_k^E is singular. However, we will show here that the condition numbers of the preconditioned enlarged and original systems are the same, provided we use a generalized condition number that considers the preconditioned matrix restricted to the orthogonal complement of its null space. In fact, each eigenvalue of $C_k L_k$ is also an eigenvalue of $C_k^E L_k^E$, and the rest of the eigenvalues of $C_k^E L_k^E$ are zero. This holds for all symmetric, positive-definite preconditioning matrices C_k with $C_k^E = (S_k^E)^T C_k S_k^E$. It follows from considering the cases $e_k^E \in \text{Ker}(S_k^E)$ and $e_k^E \notin \text{Ker}(S_k^E)$ and from observing that the relation

$$C_k^E (S_k^E)^T L_k S_k^E e_k^E = \lambda_k^E e_k^E \tag{45}$$

implies the relation

$$S_k^E C_k^E (S_k^E)^T L_k e_k = \lambda_k^E e_k, \tag{46}$$

where $e_k = S_k^E e_k^E$. The generalized condition number that we use is given by

$$\kappa(C_k^E L_k^E) = \lambda_{\max}/\lambda_{\min}, \tag{47}$$

where $\lambda_{\max}$ denotes the largest eigenvalue of $C_k^E L_k^E$ and $\lambda_{\min}$ its smallest nonzero eigenvalue. We thus have

$$\kappa(C_k L_k) = \kappa(C_k^E L_k^E). \tag{48}$$

Note that $\kappa(C_k L_k)$ reduces to the usual definition of condition number.

Any results on the condition number of $C_k L_k$ therefore extend directly to the semidefinite, preconditioned matrix $C_k^E L_k^E$ provided C_k is symmetric, positive-definite, as is the case for the BPX-, and MDS-preconditioners. It has been shown by Oswald [16]–[18], Xu [20], Zhang [25], [26], and Bornemann and Yserentant [4] that the condition number of the BPX-preconditioner is of the order $O(1)$. So we may conclude that

$$\kappa(C_k^{E,J} L_k^E) = O(1), \qquad J \in \{X, Z\}. \tag{49}$$

Thus, if we use specially scaled functions in E_k, then $C_k^{E,J} = I_k^E$ and the semidefinite system matrix that now results from the Galerkin approach has a generalized condition number that is independent of k.

3.2. Gauss–Seidel iteration for the semidefinite system. Here we consider relaxation type iterations for the semidefinite system. We focus on the GS method and its counterpart SGS, and show that the multigrid method for the standard system can be interpreted as Gauss–Seidel-type iteration for the semidefinite system. The levels used in the multigrid scheme induce a block partitioning of the system, and switching from level to level corresponds to an outer block Gauss–Seidel iteration for the semidefinite system. The MG-smoothing steps resemble inner Gauss–Seidel iterations for each block. The multigrid V-cycle with one pre- and post-smoothing step by Gauss–Seidel iterations reduces to the SGS method, which is also known as Aitken's double sweep scheme. Other MG-cycle types can also be modeled by different orderings of the block Gauss–Seidel traversal. The case of multiple smoothing steps corresponds to multiple inner iterations. Furthermore, other smoothers can be incorporated in the block GS method as inner iterations, and the SGS iteration for the semidefinite system can be used itself as a preconditioner of the CG method.

As usual, we decompose the semidefinite matrix L_k^E by $L_k^E = F_k^E + G_k^E + (G_k^E)^T$, where F_k^E and G_k^E denote the diagonal and strictly lower triangular parts of L_k^E, respectively. Then the GS method is expressed by using $C_k^{E,GS} := (F_k^E + G_k^E)^{-1}$ and SGS is expressed by using $C_k^{E,SGS} := (F_k^E + G_k^E)^{-T} F_k^E (F_k^E + G_k^E)^{-1}$. Note that $F_k^E + G_k^E$ is generally not symmetric, but it is positive definite and invertible. In what follows, we order the unknowns level-wise starting with the coarsest level and the unknowns of each level are ordered lexicographically.

3.2.1. Gauss–Seidel and multigrid. Here we demonstrate that Gauss–Seidel-type relaxations for the semidefinite system are equivalent to the multigrid method for the standard system. Without loss of generality, we restrict ourselves for simplicity to the two-level case and consider the block system

$$\begin{pmatrix} R_k^{k-1} L_k P_{k-1}^k & R_k^{k-1} L_k \\ L_k P_{k-1}^k & L_k \end{pmatrix} \begin{pmatrix} w^{(k-1)} \\ w^{(k)} \end{pmatrix} = \begin{pmatrix} R_k^{k-1} f_k \\ f_k \end{pmatrix}. \tag{50}$$

The two-level block-partitioned Gauss–Seidel iteration (with some inner relaxation scheme like point Gauss Seidel) for the semidefinite system then proceeds as follows:

0. Choose some $w_k^{E,0} = (w^{(k),0}, w^{(k-1),0})^T$ as a starting value.
   For $i = 1$ to convergence:
   1. Relax on $L_k w^{(k),i-1} = f_k - L_k P_{k-1}^k w^{(k-1),i-1}$ to obtain $w^{(k),i}$.
   2. Relax on $L_{k-1} w^{(k-1),i-1} = R_k^{k-1} f_k - R_k^{k-1} L_k w^{(k),i}$ to obtain $w^{(k-1),i}$.

Consider now the well-known correction scheme (CS) form of the multigrid method. Here, instead of exact solving the coarse grid equation, we only iterate it. It consists of the following steps:

0. Choose some u_k^0 as a starting value.
   For $i = 1$ to convergence:
   1. Relax on $L_k u_k^{i-1} = f_k$ to obtain u_k^i.
   2. Set $u_{k-1}^{i-1} = 0$ and relax on $L_{k-1} u_{k-1}^{i-1} = R_k^{k-1}(f_k - L_k u_k^i)$ to obtain u_{k-1}^i.
   3. Form the correction $u_k^i = u_k^i + P_{k-1}^k u_{k-1}^i$.

It is easy to see that principally the same equations arise in both of these schemes. The coarse grid equations are exactly the same, and although the fine grid equations are not, their difference is compensated for in Step 3 of the CS scheme. This correction step performs just the analogue of the transformation S_k^E from the nonunique representation of the approximation of the solution in E_k to its unique representation in the standard basis B_k. Therefore, in the CS scheme, the approximation to the solution is represented after each coarse grid correction step in standard basis. This is not necessary for the block Gauss–Seidel scheme because it works directly with the nonunique representation based on the generating system.

Thus, the only difference between these two schemes is their representation. They are otherwise equivalent. The interpretation of the block entries of w_k^E is therefore clarified if the correction scheme of the multigrid method is kept in mind: In our two-level example we can see $w^{(k)}$ as some sort of implicit approximation of the solution on level k and $w^{(k-1)}$ as an implicit coarse grid correction to it. In this sense, our method and McCormick's approach of Unigrid and PML are in similar spirit; see [14], [15].

The V-cycle with one pre- and one post-smoothing step is equivalent to two Gauss–Seidel iterations for the semidefinite system, where the second step is performed in reversed ordering. This is just the well-known Aitken double sweep or SGS method. Multiple pre- and post-smoothing iterations can now be modeled by block sweeps over the semidefinite system with multiple inner iterations. Of course, other smoothers can also be applied in the inner iteration, and the outer block iteration can be reordered so that other MG-cycling strategies are implemented.

The Gauss–Seidel or SGS iteration can be considered as a method for the minimization of the energy functional, where the search directions are just the coordinate axes. Using the enlarged generating system, we can interpret the unknowns of the coarse grid equations of the multigrid method as additional coordinate axes which are given via $(S_k^E)^T$ as linear combinations of the fine grid coordinate axes. The coarse grid correction now gives additional search directions for the minimization of the energy, which makes multigrid fast to converge.

Furthermore, it is easy to see that the HB-MG method of [3] can be interpreted in the extended system as well: Iteration on the unknowns of $w^{(l)}, l = k, \ldots, 2$, which belong to grid points that are contained in the next coarser grid Ω_{l-1}, is simply omitted and its values are frozen there.

Note that we obtain the usual Gauss–Seidel iteration on the fine grid system by omitting iteration over the blocks that are associated with the coarser levels. This would mean that the associated values of w_k^E values are never changed. But if we transform the resulting solution to its standard basis representation, we obtain exactly the same result as for an iteration on the fine grid system only. For remarks on an estimate concerning the condition number of the SGS-preconditioned semidefinite matrix, we refer to [7].

4. Numerical experiments. Consider the simple model problem

$$\Delta u = f \quad \text{on} \quad \Omega = (0,1)^d, \qquad d = 1, 2, \tag{51}$$

with Dirichlet boundary conditions, where $\Delta = \sum_{i=1}^{d} \partial^2/\partial x_i^2$.

4.1. Eigenvalues and condition numbers. The numerically computed eigenvalues and generalized condition numbers of the semidefinite matrix L_k^E and the MDS-preconditioned semidefinite matrix $C_k^{E,Z} L_k^E$ are shown for the 1D case in Table 1. (The BPX-preconditioned matrix $C_k^{E,X} L_k^E$ has the same condition numbers and, up to the factor 2, the same eigenvalues.) Here, s and r denote the respective size and rank of the matrix.

TABLE 1
Eigenvalues and generalized condition numbers in the 1D *case.*

			L_k^E				$C_k^{E,Z} L_k^E$			
k	s	r	$\lambda_{\min}$	$\lambda_{\max}$	κ	$\frac{\kappa(k-1)}{\kappa(k)}$	$\lambda_{\min}$	$\lambda_{\max}$	κ	$\frac{\kappa(k-1)}{\tilde{\kappa}(k)}$
1	1	1	4.0000	4.0000	1.0000	-	1.0000	1.0000	1.0000	-
2	4	3	5.5279	14.472	2.6180	2.618	1.0000	2.0000	2.0000	2.0000
3	11	7	6.6046	31.206	4.7249	1.8048	1.0000	2.8666	2.8666	1.4333
4	26	15	7.1989	63.575	8.8312	1.8690	1.0000	3.4839	3.4839	1.2156
5	57	31	7.5073	127.77	17.019	1.9272	1.0000	3.9834	3.9834	1.1434
6	120	63	7.6642	255.88	33.386	1.9616	1.0000	4.3891	4.3891	1.1018
7	247	127	7.7433	511.94	66.114	1.9803	1.0000	4.7239	4.7239	1.0763
8	502	255	7.7831	1024.00	137.57	1.9900	1.0000	5.0029	5.0029	1.0591

Note that L_k^E has a generalized condition number of $O(h_k^{-1})$, which is by a factor h_k^{-1} better than the condition number of the standard basis matrix L_k. The diagonally preconditioned matrix $C_k^{E,Z} L_k^E$, however, has the constant 1.0 as the smallest eigenvalue and a very slowly growing largest eigenvalue. Its condition number actually behaves like $O(1)$.

Using Fourier analysis [19], it is easy to see that, with $N = 2^k, h_k = \frac{1}{N}$, the $N/2$th eigenvalue

$$\lambda_{N/2} = \frac{2}{h_k} \tag{52}$$

of L_k is also an eigenvalue of L_k^E. Since $C_k^{E,Z}$ has the diagonal entries $h_l/2$, $l = 1, \ldots, k$ (cf. (41)), it is clear that 1.0 is an eigenvalue of $C_k^{E,Z} L_k^E$. This is in fact the smallest nonzero eigenvalue of $C_k^{E,Z} L_k^E$, although we have yet to prove it. Applying Gerschgorin's theorem, it is easy to show that the largest eigenvalue $\lambda_{\max}$ of $C_k^{E,Z} L_k^E$ is bounded from above by a constant, which can be estimated to be 6.8284. This is due to the fact that, at least in the 1D case, the entries of the spectrally equivalent matrix $(C_k^{E,Z})^{1/2} L_k^E (C_k^{E,Z})^{1/2}$ decrease in a certain geometric progression.

For the 2D case, the condition number and the eigenvalues of the MDS-preconditioned matrix $C_k^{E,Z} L_k^E$ are shown in Table 2. (The BPX-preconditioner again results in the same condition numbers, although the eigenvalues are now multiplied by the factor 8/3.)

TABLE 2
Eigenvalues and generalized condition numbers for $C_k^{E,Z} L_k^E$ in the 2D case.

k	s	r	$\lambda_{\min}$	$\lambda_{\max}$	κ	$\frac{\kappa(k-1)}{\kappa(k)}$
1	1	1	1.0000	1.0000	1.0000	-
2	10	9	0.8232	1.6971	2.0615	2.0615
3	59	49	0.7690	2.2785	2.9629	1.4372
4	284	225	0.7548	2.7102	3.5906	1.2118
5	1245	961	0.7512	3.0576	4.0702	1.1336

Once again, using Fourier analysis, it can be shown that the eigenvalues

$$\lambda_{n_1,n_2} = \frac{2}{3}(4 - \cos(\pi n_1 h_k) - \cos(\pi n_2 h_k) - 2\cos(\pi n_1 h_k)\cos(\pi n_2 h_k)), \tag{53}$$

of L_k, $n_1 = 1, \ldots, N-1$, $n_2 = N/2$, and $n_2 = 1, \ldots, N-1$, $n_1 = N/2$, where $N = 2^k$, $h_k = 1/N$, are also eigenvalues of L_k^E. Since $C_k^{E,Z}$ is a diagonal matrix with the constant diagonal entries 3/8, it follows that $3/8 \cdot \lambda_{n_1,N/2}$ and $3/8 \cdot \lambda_{N/2,n_2}$ are eigenvalues of $C_k^{E,Z} L_k^E$. In fact (compare Table 2), the smallest eigenvalue of $C_k^{E,Z} L_k^E$ appears to be $3/8 \cdot \lambda_{(1,N/2)} = 3/8 \cdot \lambda_{(N/2,1)}$. (Again we have yet to prove this.) Hence,

$$\begin{aligned} \lambda_{\min}(C_k^{E,Z} L_k^E) &= \frac{1}{4}\left(4 - \cos(\pi h_k) - \cos\left(\frac{\pi h_k N}{2}\right) - 2\cos(\pi h_k)\cos\left(\frac{\pi h_k N}{2}\right)\right) \\ &= 1 - \frac{1}{4}\cos\left(\frac{\pi}{2^k}\right). \end{aligned} \tag{54}$$

The largest eigenvalue grows very moderately. However, Gerschgorin's theorem can no longer be used since the entries of $(C_k^{E,Z})^{1/2} L_k^E (C_k^{E,Z})^{1/2}$ no longer decrease geometrically for all rows. Note that in the 2D case the standard generating functions in E_k are already specially scaled with respect to h_l. $C_k^{E,Z}$ preconditioning results only in a scaling by the constant factor 3/8.

We now consider the SGS-CG approach. We computed the eigenvalues and generalized condition numbers of the preconditioned matrix $C_k^{E,SGS} L_k^E$. The results for the 1D case are shown in Table 3. Compared to the BPX- and MDS-preconditioner, the SGS-preconditioner improves the condition number roughly by a factor 4.

For the 2D case the condition number and eigenvalues of $C_k^{E,SGS} L_k^E$ are shown in Table 4.

Once more, the SGS-preconditioner improves the condition number by about a factor 4 and the formula $\kappa(C_k^{E,SGS} L_k^E) \approx 1 + 1/(4\lambda_{\min}(C_k^{E,Z} L_k^E))$ gives a good estimate of its upper bound.

TABLE 3
Eigenvalues and generalized condition numbers for $C_k^{E,SGS} L_k^E$ *in the* 1D *case.*

k	s	r	$\lambda_{\min}$	$\lambda_{\max}$	κ	$\frac{\kappa(k-1)}{\kappa(k)}$
1	1	1	1.0000	1.0000	1.0000	-
2	4	3	0.8125	1.0000	1.2308	1.2308
3	11	7	0.8002	1.0000	1.2496	1.0153
4	26	15	0.8003	1.0000	1.2495	0.9999
5	57	31	0.8000	1.0000	1.2500	1.0000
6	120	63	0.8000	1.0000	1.2500	1.0000
7	147	127	0.8000	1.0000	1.2500	1.0000
8	402	255	0.8000	1.0000	1.2500	1.0000

TABLE 4
Eigenvalues and generalized condition numbers for $L_k^{E,SGS}$ *in the* 2D *case.*

k	s	r	$\lambda_{\min}$	$\lambda_{\max}$	κ	$\frac{\kappa(k-1)}{\kappa(k)}$
1	1	1	1.0000	1.0000	1.0000	-
2	10	9	0.9007	1.0000	1.1102	1.1102
3	59	49	0.8472	1.0000	1.1804	1.0632
4	284	225	0.8325	1.0000	1.2012	1.0176
5	1245	961	0.8288	1.0000	1.2065	1.0044

4.2. Convergence factors and iteration steps. Here we study the convergence properties of the different preconditioned CG and GS methods. As shown in [1], the CG method needs at most $s = \text{int}\left[1/2\sqrt{\kappa}\ |ln(2/\varepsilon)| + 1\right]$ steps to reduce the energy of the initial error by a factor $\varepsilon \in (0, 1)$, where κ is the (generalized) condition number of the preconditioned matrix to be considered. The reduction factor after s steps can be estimated by the quantity $2 \cdot \rho_{cg}^s$, where $\rho_{cg} := (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$ is the (worst-case) convergence factor. Note that, due to the sophisticated convergence properties of the CG method taking full advantage of the eigenvalue distribution of the preconditioned matrix, the upper bound $2 \cdot \rho_{cg}^s$ may be pessimistic.

We are able to derive estimates of the reduction factors from the computed condition numbers of the above tables. For example, in the case k=5, the generalized condition number of the MDS-preconditioner is close to 4.0 (compare Tables 1 and 2), so, we can expect a theoretical convergence factor of about 0.333. Also, since the generalized condition number of the SGS-preconditioner is about 1.25 (compare Tables 3 and 4), we can expect a convergence factor of about 0.0557. If we invest the additional work necessary for the Gauss–Seidel iterations on each level, we get an improvement of the convergence factor by a multiple of about 6.0.

For comparison, we performed numerical experiments to measure the number of CG steps necessary to reduce the L_2 norm of the residual r_k^E by the factors 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, and 10^{-10} for the MDS-preconditioned and the SGS-preconditioned semidefinite system. The results are given in Tables 5 and 6. For the measurements, we directly computed r_k^E, the residual of the semidefinite system, which is related to the residual r_k by $(S_k^E)^T$, as in (28). Note that the SGS-preconditioner reduces the number of iteration steps roughly by a factor of 2.5 in comparison to the MDS-preconditioner.

These results should be compared to the performance of the simple iterative method associated with the respective preconditioner for which the worst-case convergence factor is

$$\rho(I_K^E - C_k^{E,J} L_k^E) = \max_i |1 - \lambda_i(C_k^{E,J} L_k^E)|, \qquad J \in \{Z, SGS\}, \tag{55}$$

where λ_i denotes the respective eigenvalues. We see from the Tables 1 and 2 directly that the

TABLE 5
1D *case*: *Required number of* CG *steps for various residual reduction factors.*

	$C_k^{E,Z}L_k^E$					$C_k^{E,SGS}L_k^E$				
k	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
2	2	2	2	2	2	2	2	2	2	2
3	4	4	4	4	4	2	3	4	5	5
4	4	6	8	8	8	2	3	5	7	8
5	5	8	11	13	14	2	3	5	7	8
6	5	9	13	16	18	2	3	5	7	8
7	5	9	13	17	21	2	3	5	7	8
8	5	10	14	18	22	2	3	5	7	8

TABLE 6
2D *case*: *Required number of* CG *steps for various residual reduction factors.*

	$C_k^{E,Z}L_k^E$					$C_k^{E,SGS}L_k^E$				
k	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
2	2	3	3	3	3	2	3	4	5	6
3	4	6	8	9	10	2	3	5	6	7
4	5	8	12	15	18	2	3	5	6	8
5	5	9	13	16	20	2	3	5	6	8

Jacobi iteration for the semidefinite system is divergent, with spectral radius is greater 1. Note that a damping parameter could be introduced to enforce convergence.

For the symmetric Gauss–Seidel iteration on the semidefinite system, which is equivalent to the (1,1)-MG-V-cycle for the standard system, we get

$$\rho_{sgs} = 1 - \lambda_{\min}(C_k^{E,SGS}L_k^E) = 1 - 1/\kappa(C_k^{E,SGS}L_k^E). \tag{56}$$

Since the generalized condition number is close to 1.25 (compare Tables 3 and 4), we can expect a theoretical convergence factor of about 0.2. If we invest the additional work necessary for the CG iteration to gain SGS-CG, we see that the convergence factor is improved by a multiple of about 3.588.

Tables 7 and 8 show the number of steps that were needed by the Gauss–Seidel and SGS iteration in numerical experiments to reduce the residual r_k^E by the factors 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, and 10^{-10}. These iterations resemble directly a V-cycle with one post-smoothing step by (lexicographical) Gauss–Seidel and zero or one presmoothing step, respectively.

TABLE 7
1D *case*: *Required number of Gauss–Seidel and* SGS *steps for various residual reduction factors.*

	$(C^{E,Z})^{1/2}L_k^E(C^{E,Z})^{1/2}$									
	Gauss–Seidel					SGS				
k	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
2	2	2	2	2	2	2	5	8	11	13
3	4	6	8	9	11	2	5	8	11	13
4	4	8	11	14	16	2	5	8	10	13
5	4	8	12	16	20	2	5	8	10	13
6	4	8	12	16	20	2	5	8	10	13
7	4	8	12	16	21	2	5	8	10	13
8	4	8	12	17	21	2	5	7	10	13

TABLE 8
2D *case*: *Required number of Gauss–Seidel and* SGS *steps for various residual reduction factors.*

	$(C^{E,Z})^{1/2} L_k^E (C^{E,Z})^{1/2}$									
	Gauss–Seidel					SGS				
k	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
2	3	4	6	8	10	2	4	6	8	10
3	3	6	8	10	12	2	4	7	9	12
4	4	7	9	11	14	2	4	7	9	12
5	4	7	10	14	18	2	5	7	10	12

4.3. Efficiency considerations. To compare the different methods with respect to their efficiency, we restrict ourselves to the 2D model problem. BPX/MDS-CG is realized by a multigrid V-cycle followed by a CG step on the fine grid equations, where explicit smoothing iterations are omitted in the V-cycle. Note that smoothing now only takes place implicitly by simply (weighted) summing of the residuals on all levels. SGS-CG is implemented by a multigrid V-cycle with one pre- and one post-smoothing iteration (here we use lexicographical Gauss–Seidel) followed by a CG step on the fine grid equations.

Table 9 shows the number of operations per fine grid point needed by the major processes of the methods we consider. Operations $+$, $-$, $\times$, $/$, are all counted equal. Here, C_{gs} denotes the number of operations per grid point of one Gauss–Seidel smoothing iteration in the V-cycle, C_t denotes the number of operations per grid point that are necessary for the intergrid transfer operators in the V-cycle, and C_{cg} denotes the number of operations per grid point that are necessary for the CG method on the fine grid equations. Compare also [6] for the operation count of the different MG components and [1] for the CG operation count. The number, op, of operations per grid point for the different methods itself is then given in Table 10.

TABLE 9
2D *case*: *Number of operations per fine grid point for the major algorithmic processes.*

C_{gs}	C_t	C_{cg}
12	12	15

TABLE 10
2D *case*: *Number of operations per fine grid point for the different algorithms.*

	BPX-CG	SGS	SGS-CG
	$C_t + C_{cg}$	$2 \cdot C_{gs} + C_t$	$2 \cdot C_{gs} + C_t + C_{cg}$
op	27	36	51

The efficiency of these schemes is defined as the value

$$\text{eff} = -\text{op}/\ln(\rho), \tag{57}$$

which gives the number of operations per fine grid point necessary to reduce the norm of the residual at least by a factor e. For the case $k = 5$, Table 11 shows the theoretical convergence factors ρ_{th} and the resulting efficiencies derived from the condition numbers in Tables 2 and 4.

For comparison, we ran numerical tests and measured the average reduction factors after

enough steps, s, to reduce the residual by about $\varepsilon = 10^{-10}$. This average factor is defined to be

$$\rho_{\text{ave}} = \left(\frac{|r_k^{E,s}|_2}{|r_k^{E,0}|_2} \right)^{1/s} \approx \varepsilon^{1/s}. \tag{58}$$

For the results, see also Table 11.

TABLE 11
2D *case: Convergence factors and efficiencies for the different algorithms with* $k = 5$.

	BPX-CG	SGS	SGS-CG
ρ_{th}	0.333	0.200	0.0557
eff	24.58	22.37	17.66
ρ_{ave}	0.355	0.203	0.074
eff	26.06	22.58	19.68

All three methods considered perform comparably with respect to efficiency even though their convergence rates differ strongly. BPX-CG is slightly (by about a multiple of 1.15) less efficient than SGS, which in turn is slightly (by about a multiple of 1.15) less efficient than SGS-CG. This shows that it is possible to speed up multigrid somewhat by CG. Note, however, that multigrid allows other relaxation orderings and other choices for the number of pre- and post-smoothing steps that might allow for improvement in its efficiency. BPX-CG may also be improved by additional Gauss–Seidel iterations in the BPX-preconditioning process.

5. Concluding remarks. In this paper we have introduced the idea of using a generating system instead of the usual basis approach, which allows for the direct use of multilevel decompositions of a function. A Galerkin scheme based on the generating system results in a semidefinite matrix equation.

We showed that modern multilevel methods can be interpreted as standard iterative methods for the semidefinite system. The BPX- or MDS-preconditioner can be seen as simple diagonal scaling of the semidefinite system that resembles a Jacobi iteration step. Multigrid algorithms are equivalent to Gauss–Seidel iterations and can be used as preconditioners for CG as well, resulting in a substantial improvement of the convergence rate.

Of course, an efficient implementation of these methods must follow the traditional way that is already employed in the multigrid method. From this point of view, we have presented nothing really new. However, the interpretation of different multilevel methods in terms of a generating system as standard iteration techniques on a semidefinite system fully clarifies the relations and differences between them and provides a simple and natural approach to their general treatment. We believe that the development of new multilevel methods will be substantially simplified by this perspective.

Generalizations of the semidefinite system can be derived using coarse grid spaces other than the standard coarse grid spaces used here. For example, E_k can be enlarged to contain all possible coarse grid basis functions with respect to standard coarsening and semicoarsening in all coordinate directions, allowing multilevel methods to be derived for sparse grids [8], [9], [24].

REFERENCES

[1] O. AXELSSON AND V. BARKER, *Finite Element Solutions of Boundary Value Problems*, Academic Press, New York, 1984.

[2] O. AXELSSON AND P. VASSILEVSKI, *Algebraic multilevel preconditioning methods*, Numer. Math., 56 (1989), pp. 157–177.

[3] R. BANK, T. DUPONT, AND H. YSERENTANT, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[4] F. BORNEMANN AND H. YSERENTANT, *A basic norm equivalence for the theory of multilevel methods*, Zuse Institut Berlin, Berlin, preprint 92-1, 1992.

[5] J. BRAMBLE, J. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 31 (1990), pp. 333–390.

[6] M. GRIEBEL, *Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen-Transformations-Mehrgitter-Methode*, TUM-I9007 TU München, Institut f. Informatik; SFB-Report 342/4/90 A, 1990.

[7] ———, *Multilevel algorithms considered as iterative methods on indefinite systems*, TUM-I9143 TU München, Institut f. Informatik; SFB-Report 342/29/91 A, 1991.

[8] ———, *Parallel multigrid methods on sparse grids*, in Internat. Series of Numer. Math., Vol. 98, Birkhäuser-Verlag, Basel, 1991.

[9] ———, *A parallelizable and vectorizable multi-level algorithm on sparse grids*, in Parallel Algorithms for Partial Differential Equations, Proc. 6th GAMM-Seminar, Kiel, Jan. 19–21, 1990, Notes on Numerical Fluid Mechanics, Vol 31, W. Hackbusch, ed., Vieweg-Verlag, 1991.

[10] W. HACKBUSCH, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, Heidelberg, New York, 1985.

[11] R. KETTLER, *Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods*, in Multigrid Methods, Lecture Notes in Mathematics, 960, Springer-Verlag, 1982.

[12] D. LUENBERGER, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, MA, 1973.

[13] S. MCCORMICK, *Multigrid Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[14] ———, *Multilevel adaptive methods for partial differential equations*, SIAM Frontiers in Applied Mathematics 6, Philadelphia, PA, 1989.

[15] ———, *Multilevel projection methods for partial differential equations*, CBMS-NSF Regional Conference Series in Applied Mathematics 62, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[16] P. OSWALD, *Approximationsräume und Anwendungen in der Theorie der Multilevel-Verfahren*, Manuskript, Friedrich Schiller Universität Jena, 1991.

[17] ———, *Two remarks on multilevel preconditioners*, Bericht Nr. Math/91/1, Friedrich Schiller Universität Jena, Mathematische Fakultät, 1991.

[18] ———, *Norm equivalencies and multilevel Schwarz preconditioning for variational problems*, Bericht Nr. Math/92/1, Friedrich Schiller Universität Jena, Mathematische Fakultät, 1992.

[19] K. STÜBEN AND U. TROTTENBERG, *Multigrid methods: fundamental algorithms, model problem analysis and applications*, in Multigrid methods, Lecture Notes in Mathematics, 960, Springer-Verlag, 1982.

[20] J. XU, *Iterative Methods by Space Decomposition and Subspace Correction: A Unifying Approach*, Report No. AM67, Dept. of Mathematics, Pennsylvania State University, University Park, PA, 1990. Revised 1992.

[21] ———, *Theory of Multilevel Methods*, Report No. AM48, Dept. of Mathematics, Pennsylvania State University, University Park, 1989.

[22] H. YSERENTANT, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

[23] ———, *Two Preconditioners Based on the Multi-Level Splitting of Finite Element Spaces*, Report No. SC89-9, Konrad-Zuse-Zentrum für Informationstechnik, Berlin, 1990.

[24] C. ZENGER, *Sparse Grids*, in Parallel Algorithms for Partial Differential Equations, Proc. 6th GAMM-Seminar, Kiel, Jan. 19–21, 1990, Notes on Numerical Fluid Mechanics, Vol 31, W. Hackbusch, ed., Vieweg-Verlag, 1991.

[25] X. ZHANG, *Multilevel Additive Schwarz Methods*, Tech. Report 582, Department of Computer Science, Courant Institute, New York University, 1991.

[26] ———, *Multilevel Schwarz Methods*, Tech. Report CS-TR-2894, UMIACS-TR-92-52, Dept. of Computer Science, University of Maryland, College Park, MD, 1992.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 566–576, May 1994

005

ANALYSIS OF V-CYCLE MULTIGRID ALGORITHMS FOR FORMS DEFINED BY NUMERICAL QUADRATURE*

J. H. BRAMBLE†, C. I. GOLDSTEIN‡, AND J. E. PASCIAK‡

Abstract. The authors describe and analyze certain V-cycle multigrid algorithms with forms defined by numerical quadrature applied to the approximation of symmetric second-order elliptic boundary value problems. This approach can be used for the efficient solution of finite element systems resulting from numerical quadrature as well as systems arising from finite difference discretizations. The results are based on a regularity free theory and hence apply to meshes with local grid refinement as well as the quasi-uniform case. It is shown that uniform (independent of the number of levels) convergence rates often hold for appropriately defined V-cycle algorithms with as few as one smoothing per grid. These results hold even on applications without full elliptic regularity, e.g., a domain in R^2 with a crack.

Key words. multigrid methods, elliptic equations, finite elements, numerical quadrature

AMS subject classifications. 65F10, 65N30

1. Introduction.

In recent years, multigrid methods have been extensively used to efficiently solve the discrete equations that arise in the numerical approximation of partial differential equations (see [9], [14], [17] and references cited therein). In conjunction, there has been intensive research into the theoretical understanding of the convergence properties of these methods (cf. [1]–[3], [6], [11], [12], [14], [16], [17]). These results provide a uniform convergence rate (with respect to the number of grid levels) for the V-cycle algorithm in the case of full elliptic regularity and a quasi-uniform mesh. It was shown in [5], using a new general multigrid theory, that uniform estimates hold for the V-cycle algorithm, with only one smoothing per grid per iteration, even in the absence of full regularity and in the presence of mesh refinements. The results in [1]–[3], [5], [6], [11], [12], [14], [16], [17] were applied to the finite element method with stiffness matrix computed exactly.

In practice, the stiffness matrix is usually computed approximately using a suitable numerical quadrature. Furthermore, many standard finite difference schemes can be expressed in terms of the finite element method with a suitable quadrature rule. The V-cycle multigrid algorithm with numerical quadrature on each grid level was analyzed in [13]. In [13], the bilinear forms on the coarser grids were defined by using coarser grid quadrature. The results in [13] provide a uniform convergence rate for approximations of problems with full regularity utilizing a quasi-uniform mesh.

The purpose of the present paper is to show that the theoretical results in [5] yield a uniform convergence rate for the natural V-cycle multigrid algorithm that uses the fine grid quadrature to define the forms on all grid levels. These results hold with mesh refinements and for problems without full elliptic regularity (e.g., an L-shaped domain or a domain with a crack boundary). Uniform convergence estimates are also provided for a standard finite difference scheme.

The remainder of the paper is outlined as follows. In §2, we describe the theoretical multigrid framework in [5] and provide an abstract perturbation lemma. In §3, we consider

*Received by the editors May 4, 1992; accepted for publication (in revised form) December 1, 1992. This manuscript has been authored under U.S. Department of Energy contract DE-AC02-76CH00016. Accordingly, the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. This work was also supported in part by National Science Foundation grant DMS-9007185 and by U.S. Army Research Office through the Mathematical Sciences Institute, Cornell University.

†Department of Mathematics, Cornell University, Ithaca, New York 14853-7901 (bramble@math.cornell.edu).

‡Applied Mathematics Department, Brookhaven National Laboratory, Long Island, New York 11973 (goldstei@bnl.gov, pasciak@jep.das.bnl.gov).

elliptic boundary value problems approximated by the finite element method with numerical quadrature. Using the perturbation lemma of §2 and the theory in [5], we prove uniform convergence results for corresponding multigrid V-cycle algorithms. This type of analysis is applied to finite difference multigrid in §4.

2. Abstract multigrid theory. We begin by briefly describing the general multigrid framework and results in [5]. We then provide a perturbation lemma that will be used to analyze the case of numerical quadrature.

Suppose we are given a nested sequence of finite dimensional vector spaces $M_1 \subseteq M_2 \subseteq \cdots \subseteq M_J$. Associated with this sequence, assume we are given additional subspaces, $\tilde{M}_k \subseteq M_k$, for $k = 2, 3, \ldots, J$. The multigrid algorithm will involve smoothing only on the subspaces $\{\tilde{M}_k\}$. Let $A(\cdot, \cdot)$ and $(\cdot, \cdot)$ be symmetric positive definite bilinear forms defined on $M_J \times M_J$ and let $\| \cdot \|_A$ and $\| \cdot \|$ denote the corresponding norms, respectively. We define a V-cycle multigrid algorithm for the solution of the problem: Given $f \in M_J$, find $u_J \in M_J$ satisfying

$$A(u_J, v) = (f, v) \quad \text{for all } v \in M_J. \tag{1}$$

The multigrid algorithm is defined in terms of auxiliary operators. For $k = 1, \ldots, J$, define the operator $A_k : M_k \to M_k$ by

$$(A_k w, v) \equiv A(w, v) \quad \text{for all } v \in M_k. \tag{2}$$

The operator A_k is clearly symmetric (in both the $A(\cdot, \cdot)$ and $(\cdot, \cdot)$ inner products) and positive definite. Let λ_k denote the largest eigenvalue of A_k. Also define the orthogonal projectors $P_k, Q_k : M_J \to M_k$ by

$$A(P_k w, v) = A(w, v) \quad \text{for all } v \in M_k,$$

and

$$(Q_k w, v) = (w, v) \quad \text{for all } v \in M_k.$$

It is easily seen that $Q_\ell A_k = A_\ell P_\ell$ if $\ell \leq k$. The operators $\tilde{A}_k$, $\tilde{P}_k$, and $\tilde{Q}_k$ are defined analogously by replacing M_k with $\tilde{M}_k$ above.

To introduce smoothing into the multigrid algorithm, we use "generic" smoothing operators $R_k : M_k \to \tilde{M}_k$, $k = 2, \ldots, J$, satisfying conditions (S.1)–(S.3) below. Let R_k^t denote the $(\cdot, \cdot)$ adjoint of R_k and set

$$K_k = I - R_k A_k$$

and

$$\bar{R}_k = (I - K_k^* K_k) A_k^{-1}.$$

Here $K_k^* = I - R_k^t A_k$ denotes the adjoint of K_k with respect to the $A(\cdot, \cdot)$ inner product. We set $R_1 = A_1^{-1}$, i.e., we solve on the coarsest space. Finally, set $T_k \equiv R_k Q_k A_J = R_k A_k P_k$. Note that $T_1 = P_1$.

We make the following three assumptions on R_k for $k = 2, \ldots, J$.

(S.1) There is a constant $C_R \geq 1$ that does not depend on k such that the smoothing procedure satisfies

$$\frac{\| u \|^2}{\lambda_k} \leq C_R(\bar{R}_k u, u) \quad \text{for all } u \in \tilde{M}_k. \tag{3}$$

Note that (3) holds with $C_R = 1$ for $k = 1$ since $\bar{R}_1 = A_1^{-1}$. The next assumption states that the smoothers must be properly scaled.

(S.2) There is a constant $\theta < 2$ not depending on k such that

$$A(T_k v, T_k v) \leq \theta A(T_k v, v) \quad \text{for all } v \in M_k. \tag{4}$$

The last assumption implies that $\tilde{M}_k$ is an invariant subspace under R_k^t.

(S.3) $R_k = R_k \tilde{Q}_k$.

Note that (S.3) is automatically satisfied for symmetric smoothers since the range of R_k is contained in $\tilde{M}_k$. Conditions (S.1)–(S.3) were shown to hold in [4] for smoothing operators corresponding to various Jacobi and Gauss–Seidel iterative procedures.

We next define the multigrid operator $B_J : M_J \to M_J$ by induction. For each $f \in M_J$, $B_J f$ is to approximate the solution $u_J = A_J^{-1} f$ of (1).

DEFINITION 2.1. *Define* $B_1 = A_1^{-1}$. *For* $k > 1$, $B_k g$ *for* $g \in M_k$ *is defined as follows*:

Step 1. *Set*

$$x = R_k^t g. \tag{5}$$

Step 2. *Set* $y = x + q$, *where* q *is given by*

$$q = B_{k-1} Q_{k-1}(g - A_k x). \tag{6}$$

Step 3. *Set* $B_k g = y + R_k(g - A_k y)$.

Definition 2.1 defines a symmetric V-cycle algorithm with one smoothing before and after the coarse grid correction (Step 2). As in [5], subsequent theorems hold for the W-cycle and other algorithms that use more than one iteration for the coarse grid correction. The results also hold for algorithms that use more smoothings per grid level as long as one alternates between R_k and R_k^t in Steps 1 and 3. This again results in a symmetric operator B_J [7]. Finally, an analogous contraction result holds for nonsymmetric cycling, where smoothing is only done either before or after the correction step, i.e., Step 1 or Step 3 is skipped.

The main convergence result proved in [5] for the multigrid operator B_J shows that, under suitable assumptions, the following estimate holds:

$$0 \leq A((I - B_J A_J)u, u) \leq \delta A(u, u) \quad \text{for all } u \in M_J, \tag{7}$$

where $\delta \in (0, 1)$ is independent of J. The proof of (7) in [5] was based on two conditions (as well as assumptions (S.1)–(S.3) on the smoother).

The first condition may be stated as follows:

(A.1) There is a constant $C_0 \geq 1$ satisfying

$$A(w, w) \leq C_0 \left[A(P_1 w, w) + \sum_{k=2}^{J} \frac{\| \tilde{A}_k \tilde{P}_k w \|^2}{\lambda_k} \right] \quad \text{for all } w \in M_J. \tag{8}$$

The second condition can be written as follows:

(A.2) There is a positive number $\epsilon < 1$ and a positive constant $\tilde{C}$ such that for $k = 2, \ldots, J$ and $l \leq k$,

$$\lambda_k^{-1} \| \tilde{A}_k \tilde{P}_k w \|^2 \leq (\tilde{C} \epsilon^{k-\ell})^2 A(w, w) \quad \text{for all } w \in M_\ell. \tag{9}$$

In this paper, we are interested in the behavior of the above conditions under perturbation of forms. To this end, let $A^E(\cdot, \cdot)$ be an alternative quadratic form on M_J and define the

operator A_k^E as above with $A(\cdot,\cdot)$ replaced by $A^E(\cdot,\cdot)$. Let λ_k^E denote the largest eigenvalues of A_k^E. We then have the following lemma.

LEMMA 2.1. *Assume that conditions* (A.1) *and* (A.2) *hold for* $A^E(\cdot,\cdot)$. *In addition, suppose that*

$$K_0 A(w,w) \le A^E(w,w) \le K_1 A(w,w) \quad \textit{for all } w \in M_J \tag{10}$$

and for some $\alpha > 0$,

$$|A^E(\phi,\psi) - A(\phi,\psi)| \le C\lambda_k^{-\alpha/2}\|\phi\|_A\|\psi\|_A \quad \textit{for all } \phi \in \tilde{M}_k,\ \psi \in M_k. \tag{11}$$

Finally, if $\lambda_k/\lambda_{k+1} \le \eta$ *for fixed* $\eta < 1$, *then conditions* (A.1) *and* (A.2) *hold for* $A(\cdot,\cdot)$.

Proof. We first note that condition (A.1) for $A(\cdot,\cdot)$ follows from (10) and Lemma 3.1 in [5]. We are thus left with proving that condition (A.2) holds.

By (10), it immediately follows that λ_k and λ_k^E are of comparable size. Let $w \in M_\ell$ with $1 \le \ell \le k$. Note that

$$\lambda_k^{-1} \parallel \tilde{A}_k \tilde{P}_k w \parallel^2 = \lambda_k^{-1}\left(\sup_{\phi\in\tilde{M}_k} \frac{|A(w,\phi)|}{\|\phi\|}\right)^2. \tag{12}$$

We clearly have that

$$\sup_{\phi\in\tilde{M}_k} \frac{\|A(w,\phi)\|}{\|\phi\|} \le \sup_{\phi\in\tilde{M}_k} \frac{|A^E(w,\phi)|}{\|\phi\|} + \sup_{\phi\in\tilde{M}_k} \frac{|A(w,\phi) - A^E(w,\phi)|}{\|\phi\|}. \tag{13}$$

Since condition (A.2) holds for $A^E(\cdot,\cdot)$, there exists a constant $\tilde{C}$ and a positive number $\epsilon < 1$ satisfying

$$(\lambda_k^E)^{-1} \sup_{\phi\in\tilde{M}_k} \frac{|A^E(w,\phi)|^2}{\|\phi\|^2} \le \tilde{C}\epsilon^{2k-2\ell} A(w,w). \tag{14}$$

By (11), the second term in (13) is bounded by

$$\frac{|A(w,\phi) - A^E(w,\phi)|}{\|\phi\|} \le C\lambda_k^{-\alpha/2} A(w,w)^{1/2} \frac{A(\phi,\phi)^{1/2}}{\|\phi\|}.$$

Using the definition of λ_k gives

$$\lambda_k^{-1} \sup_{\phi\in\tilde{M}_k} \frac{|A(w,\phi) - A^E(w,\phi)|^2}{\|\phi\|^2} \le C\lambda_k^{-\alpha} A(w,w). \tag{15}$$

The inequality (A.2) for $A(\cdot,\cdot)$ follows from (12)–(15). This completes the proof of the lemma. □

3. The finite element method with numerical integration. In this section we describe the finite element method with numerical quadrature for approximately solving elliptic boundary value problems. We then apply Definition 2.1 to define an efficient multigrid algorithm for solving the resulting system of equations. The main result (Theorem 3.1 below) shows that this multigrid algorithm has a convergence rate independent of the number of grid levels when a sufficiently accurate quadrature scheme is used. This accuracy condition is also required for the finite element error analysis of numerical integration (cf. [10]) and is referred to as the "patch test."

We employ standard notation for Sobolev spaces [15]. Let $H^s(\Omega)$ denote the Sobolev space of square-integrable functions on Ω with square-integrable derivatives up to order s, where s is a nonnegative integer. Let $\|\cdot\|_{H^s} \equiv \|\cdot\|_{H^s(\Omega)}$ denote the corresponding Sobolev norm. The inner product and norm on $L^2(\Omega) = H^0(\Omega)$ are denoted by $(\cdot,\cdot)_{L^2}$ and $\|\cdot\|_{L^2}$, respectively.

In the remainder of this paper, we let C, with or without subscript, denote a generic positive constant. It will take on different values in different places, but it will always be independent of the mesh parameters and the number of levels in the multigrid algorithm.

Let Ω be a bounded domain in R^d with polygonal boundary. We include the case when $\Omega \subset R^2$ is a domain with a crack. We consider the Dirichlet problem

$$\begin{aligned} Lu &= f \quad \text{in} \quad \Omega, \\ u &= 0 \quad \text{on} \quad \partial\Omega, \end{aligned} \tag{16}$$

where

$$Lv = -\sum_{i,j=1}^{2} \frac{\partial}{\partial x_i}\left(a_{ij}\frac{\partial v}{\partial x_j}\right).$$

We assume that the coefficients are in $C^{0,\alpha}(\Omega)$ for some α greater than zero. Furthermore, we assume that the coefficient matrix $\{a_{ij}(x)\}$ is symmetric positive definite with smallest eigenvalue bounded away from zero independently of $x \in \Omega$.

We consider the variational formulation of (16). Let $H_0^1(\Omega)$ denote the subset of $H^1(\Omega)$ consisting of functions with vanishing trace on $\partial\Omega$. The solution u of (16) satisfies

$$A^E(u, v) = (f, v)_{L^2} \quad \text{for all } v \in H_0^1(\Omega). \tag{17}$$

Here $A^E(\cdot,\cdot)$ is the generalized Dirichlet form given by

$$A^E(v, w) = \sum_{i,j=1}^{2} \int_\Omega a_{ij}\frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j}\,dx. \tag{18}$$

The superscript E above indicates that the form is computed by exact integration. Subsequent forms will be defined by numerical quadrature.

We shall consider finite element spaces with a quasi-uniform mesh as well as mesh refinements. The quasi-uniform mesh partitioning is standard, but we include the details for completeness.

(i) *Quasi-uniform mesh.* To define the approximation spaces, we first define the underlying mesh partitioning. We assume that a unit size coarse finite element triangulation of the original domain Ω is given. Associated with the mesh partitioning, we are given a rule for refinement. For example, one triangle can be refined into four by connecting the midpoints of the edges.

The mesh triangulations $\{\mathcal{T}_k\}$ can be defined by induction. The coarse triangulation above provides $\mathcal{T}_1$. Given that a triangulation $\mathcal{T}_{k-1}$ has been defined, $\mathcal{T}_k$ is then defined by refining the triangles $\tau_k^i \in \mathcal{T}_{k-1}$ using the refinement rule. Thus each $\mathcal{T}_k$ consists of triangles $\{\tau_k^i\}$ such that $\bar{\Omega} = \cup_i \bar{\tau}_k^i$.

The finite element space M_k for each $k = 1, \ldots, J$ is defined to be a space of piecewise polynomial functions of degree $\leq K$ with respect to the triangulation $\mathcal{T}_k$. These functions are continuous on Ω and vanish on $\partial\Omega$. (We refer to [10] for comprehensive discussions of various specific finite element spaces.) We say the family of spaces $\{M_k\}$ has degree K. In the present example, each $\tilde{M}_k = M_k$, i.e., we smooth on all functions in M_k.

(ii) *General mesh refinement.* We also consider finite element approximations that utilize a locally refined mesh as described in [8]. Such mesh refinements are convenient for accurate modeling of problems with various types of singular behavior. We consider for simplicity the piecewise linear finite element space (i.e., $K = 1$), although we allow a very general form of refinement.

Following [8], we start with a coarse quasi-uniform triangulation. The refinement triangulation is defined in terms of a sequence of (open) mesh domains

$$\Omega_J \subseteq \Omega_{J-1} \subseteq \cdots \subseteq \Omega_1 = \Omega.$$

The only restrictions on the mesh domains $\{\Omega_k\}$ are that the boundary of Ω_k, for $k > 1$, consists of edges of mesh triangles in the triangulation $\mathcal{T}_{k-1}$ and that there is at least one edge of $\mathcal{T}_{k-1}$ contained in Ω_k. These mesh domains control the region of refinement. If τ_{k-1}^i is a triangle contained in Ω_k, then it is broken into four smaller triangles (in the triangulation $\mathcal{T}_k$) by the lines connecting the midpoints of the edges. Alternatively, if τ_{k-1}^i is in the complement of Ω_k, then it is not subdivided but is directly included into the triangulation $\mathcal{T}_k$. A simple example of this construction is the case of a unit square with local refinement near the corner. In this case we take $\Omega_k = \Omega$ for $k = 1, \ldots, j$ and $\Omega_k = (1 - 2^{j-k}, 1) \times (1 - 2^{j-k}, 1)$ for $k = j + 1, \ldots, J$.

The space M_k is defined to be the set of piecewise linear functions with respect to the triangulation $\mathcal{T}_k$ that are continuous on Ω and vanish on $\partial\Omega$. The continuity condition implies that the finer grid nodes on a coarse-fine boundary are "slave nodes" in the sense that the values of the function there are completely determined by the values of the function on the nearby coarse grid points. In this case, if $\Omega_{k-1} \neq \Omega_k$, then the subspace $\tilde{M}_k$ on which we smooth is a proper nonzero subspace of M_k. In fact, we define $\tilde{M}_k$ to be the functions in M_k that are zero outside of Ω_k. Thus we smooth on a given level just in the region where new nodes are being added in the refinement scheme. It is easy to see that the mesh corresponding to the space $\tilde{M}_k$ is quasi uniform of size $h_k \equiv 2^{-k+1}h_1$.

The finite element approximation, $u_J^E \in M_J$, to the solution of (16) is given by

$$A^E(u_J^E, v) = (f, v) \quad \text{for all } v \in M_J.$$

Here $(\cdot, \cdot)$ denotes the inner product in $L^2(\Omega)$. We will be interested in applying the multigrid algorithms and analysis to a perturbed form defined by numerical integration.

To define the form by numerical integration, we start with a numerical quadrature scheme Q_J^i over each element $\tau_J^i \in \mathcal{T}_J$. Consider the reference triangle τ and introduce the reference quadrature

$$\int_\tau \hat{\phi}(\hat{x})d\hat{x} \approx \sum_{\ell=1}^{L} w_\ell \hat{\phi}(b_\ell),$$

where the w_ℓ are positive weights and the $b_\ell \in \tau$ are quadrature points. The quadrature rule on each fine grid triangle τ_J^i is taken to be

$$\int_{\tau_J^i} \phi(x)dx \approx \sum_{\ell=1}^{L} w_{J,\ell}^i \phi(b_{J,\ell}^i) \equiv Q_J^i[\phi],$$

where $\phi(x) = \hat{\phi}(\hat{x})$ and the weights $w_{J,\ell}^i$ and quadrature points $b_{J,\ell}^i$ are defined in terms of the w_ℓ and b_ℓ by means of an affine mapping from τ_J^i onto τ that takes each x in τ_J^i into $\hat{x}$ in τ. We refer to [10] for detailed descriptions of numerical integration in connection with the finite element method.

We require that the above quadrature rule be exact when ϕ is a polynomial of degree $2K-2$. This is referred to as the patch test and is also a basic assumption used in finite element error analysis (see [10, §4.1], and references cited therein for specific examples).

The approximate form $A(v, w)$ is given by

$$A(v,w) = \sum_{\tau_J^l \in T_J} Q_J^l\left[\sum_{i,j=1}^{2}\left(a_{ij}\frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j}\right)\right] \quad \text{for all } v, w \in M_J. \tag{19}$$

We present results for multigrid algorithms for computing the solution of the discrete problem: Find $u_J \in M_J$ satisfying

$$A(u_J, v) = F(v) \quad \text{for each } v \in M_J. \tag{20}$$

Here F is a linear functional defined on M_J. Typically, F is defined by using some form of numerical quadrature. The implementation of the algorithm developed in §2 can be used to solve (20) as well as problem (1) [6].

Assume that we have smoothing operators $R_k : M_k \to M_k$ satisfying (S.1)–(S.3) for each $k = 2, \ldots, J$. R_k may be defined in the same way as for the finite element method without numerical integration ([4], [14], [17]). Definition 2.1 provides a multigrid operator, $B_J : M_J \to M_J$, for which the following result holds.

THEOREM 3.1. *Assume that conditions* (S.1)–(S.3) *in* §2 *hold for the smoothing operators. Assume that a fixed quadrature scheme, which is exact on polynomials of degree* $2K-2$, *is used for every term defining* $A(\cdot,\cdot)$. *Let* B_J *be defined by Definition* 2.1. *Then* (7) *holds with* $\delta \in [0, 1)$ *independent of* J.

Proof. By Theorem 3.2 of [5], conditions (A.1) and (A.2) hold for $A^E(\cdot,\cdot)$. By Lemma 2.1, it suffices to verify (10) and (11). Since the same quadrature scheme was used for every term in defining $A(\cdot,\cdot)$, Theorem 4.1.2 of [10] implies that (10) holds.

We next show that (11) holds. Let ϕ and ψ be in $\tilde{M}_k$ and M_k, respectively. By definition, the support of ϕ is contained in Ω_k. Consequently,

$$|A^E(\phi,\psi) - A(\phi,\psi)| = \left|\sum_{i,j=1}^{2}\Bigg(\sum_{\tau_J^l \subseteq \Omega_k}\int_{\tau_J^l} a_{ij}\frac{\partial\phi}{\partial x_i}\frac{\partial\psi}{\partial x_j}\,dx - Q_J^l\left(a_{ij}\frac{\partial\phi}{\partial x_i}\frac{\partial\psi}{\partial x_j}\right)\Bigg)\right|. \tag{21}$$

Since $\tau_J^l \subseteq \Omega_k$, its diameter is bounded by h_k. Using the property that the quadrature is exact on polynomials of degree $2K-2$, a typical term in (21) can be written

$$\begin{aligned}&\int_{\tau_J^l} a_{ij}\frac{\partial\phi}{\partial x_i}\frac{\partial\psi}{\partial x_j}\,dx - Q_J^l\left(a_{ij}\frac{\partial\phi}{\partial x_i}\frac{\partial\psi}{\partial x_j}\right)\\ &= \int_{\tau_J^l}(a_{ij}(x)-\bar{a}_{ij})\frac{\partial\phi}{\partial x_i}\frac{\partial\psi}{\partial x_j}\,dx - Q_J^l\left((a_{ij}(x)-\bar{a}_{ij})\frac{\partial\phi}{\partial x_i}\frac{\partial\psi}{\partial x_j}\right).\end{aligned} \tag{22}$$

Here $\bar{a}_{ij}$ denotes the average value of a_{ij} on τ_J^l. We clearly have that the integral on the right-hand side of (22) is bounded by

$$Ch_k^{\alpha}\|a_{ij}\|_{C^{0,\alpha}(\tau_J^l)}\|\phi\|_{H^1(\tau_J^l)}\|\psi\|_{H^1(\tau_J^l)}.$$

Similarly, the quadrature in (22) is bounded by

$$Ch_k^{\alpha}\|a_{ij}\|_{C^{0,\alpha}(\tau_J^l)} \sum w_{J,m}^l \left|\frac{\partial\phi}{\partial x_i}(b_{J,m}^l)\right| \left|\frac{\partial\psi}{\partial x_j}(b_{J,m}^l)\right|$$

$$\leq Ch_k^{\alpha}\|a_{ij}\|_{C^{0,\alpha}(\tau_J^l)} \left(\sum w_{J,m}^l \left|\frac{\partial\phi}{\partial x_i}(b_{J,m}^l)\right|^2\right)^{1/2} \left(\sum w_{J,m}^l \left|\frac{\partial\psi}{\partial x_j}(b_{J,m}^l)\right|^2\right)^{1/2}$$

$$= Ch_k^{\alpha}\|a_{ij}\|_{C^{0,\alpha}(\tau_J^l)} \left\|\frac{\partial\phi}{\partial x_i}\right\|_{L^2(\tau_J^l)} \left\|\frac{\partial\psi}{\partial x_j}\right\|_{L^2(\tau_J^l)}.$$

We again used the exactness property of the quadrature rule in the last step. Combining the above inequalities and summing over τ_J^l, i, and j shows that

$$|A^E(\phi,\psi) - A(\phi,\psi)| \leq Ch_k^{\alpha}\|\phi\|_1\|\psi\|_1.$$

Equation (11) follows from the fact that $h_k \leq C\lambda_k^{-1/2}$. This completes the proof of the theorem. □

Remark 3.1. The hypothesis that the same quadrature was used on each term in the definition of $A(\cdot,\cdot)$ was only required to prove (10). Note that (11) holds even if different quadrature rules (satisfying the exactness property) are used on different terms. Moreover, (11) implies (10) in the quasi-uniform case if h_J is sufficiently small. In the refinement case, (11) implies (10) if h_1 is sufficiently small.

Remark 3.2. There is no difficulty in proving that (10) and (11) hold in the case of an operator with lower-order term with coefficient a provided that (i) the coefficient a is nonnegative and in $C^{0,\alpha}(\Omega)$ for some α greater than zero, and (ii) that the quadrature rule for this term has positive weights and is exact on constants.

One first proves that on any triangle τ_J^l in Ω_k, for $\phi \in \tilde{M}_k$ and $\psi \in M_k$,

$$\left|\int_{\tau_J^l} a\phi(\psi - \bar{\psi})\,dx\right| + \left|\tilde{Q}_J^l(a\phi(\psi - \bar{\psi}))\right| \leq ch_k\|\phi\|_{L^2(\tau_J^l)}\|\psi\|_{H^1(\tau_J^l)}.$$

Here, $\tilde{Q}^i{}_J$ is the quadrature used on the low-order term and $\bar{\psi}$ denotes the mean value of ψ on τ_J^l. Inequality (11) then easily follows from

$$\begin{aligned}\int_{\tau_J^l} a\phi\psi\,dx - \tilde{Q}_J^l(a\phi\psi) &= \int_{\tau_J^l} a\phi(\psi - \bar{\psi})\,dx - \tilde{Q}_J^l(a\phi(\psi - \bar{\psi})) \\ &\quad + \int_{\tau_J^l} a(\phi - \bar{\phi})\bar{\psi}\,dx - \tilde{Q}_J^l(a(\phi - \bar{\phi})\bar{\psi}) \\ &\quad + \int_{\tau_J^l} (a - \bar{a})\bar{\phi}\bar{\psi}\,dx - \tilde{Q}_J^l((a - \bar{a})\bar{\phi}\bar{\psi}).\end{aligned}$$

Thus, Theorem 3.1 holds for this case as well.

The multigrid algorithm defined by Definition 2.1 uses the fine grid quadrature to define the form on each of the subspaces. The sequence of stiffness matrices can be computed as follows. Let $N_k \equiv \dim(M_k)$ and let $\{\phi_k^i\}$, $i = 1,\ldots,N_k$ denote the standard nodal basis for M_k for each $k = 1,\ldots,J$. For $k = J$, we compute $A(\phi_J^i, \phi_J^j)$ directly from (19). If $k < J$, we express the basis functions for M_k as linear combinations of those for M_{k+1}. The number

of nonzero coefficients is bounded independent of k. Consequently, any stiffness matrix entry for the kth grid can be calculated from those of the $k+1$st grid with a fixed cost. This process is repeated, starting from the $J-1$st grid and descending to the coarsest grid. The total work is obviously bounded by a fixed multiple of the number of points on the finest grid.

Remark 3.3. An alternative multigrid algorithm can be defined as in [13] using numerical quadrature with respect to the kth grid to define the form on M_k. The resulting algorithm does not fit into the framework of [5] and uniform convergence results for the V-cycle algorithm were only proved in the case of full elliptic regularity [13].

Remark 3.4. Although the operator Q_{k-1} appears in the definition of the multigrid algorithm (Definition 2.1), it does not appear in the actual implementation provided that an appropriate choice of smoother is used, for example, Gauss–Seidel or Jacobi. This is discussed in detail in [5].

4. A finite difference application. In this section, we show that the stiffness matrix for a standard finite difference scheme in two spatial dimensions is equal to that from piecewise linear finite elements with an appropriate quadrature. Thus, we can apply Theorem 3.1 to obtain multigrid convergence estimates, independent of the number of grid levels, for variational multigrid methods applied to standard finite difference schemes.

We assume for simplicity that the domain Ω is a union of squares in a coarse square mesh with uniform mesh size h_1 with mesh lines with x_1,x_2 values that coincide with integer multiples of h_1. Ω need not be convex (e.g., Ω could be an L-shaped domain or slit domain). The coarse finite difference mesh is formed by the above-mentioned squares. Successively finer grids are formed by breaking each square into four subsquares in the obvious way. We will only consider the quasi-uniform case although the refinement case can be handled similarly provided that one uses the finite element quadrature to develop the finite difference schemes near the coarse/fine boundaries.

We employ the following finite difference notation. Set $\bar{\Omega}^h \equiv \{x_{\ell,m} \equiv (\ell h, mh) \in \bar{\Omega}\}$. Define $\Omega_h = \Omega \cap \bar{\Omega}_h$ and $\partial\Omega^h \equiv \bar{\Omega}^h \cap \partial\Omega$. Let S_h denote the set of grid functions on $\bar{\Omega}^h$ vanishing on $\partial\Omega^h$. If $V \in S_h$, then $V_{i,j}$ denotes the value of V at (ih, jh).

In this section, we take

$$Lu = -\frac{\partial}{\partial x_1} a \frac{\partial u}{\partial x_1} - \frac{\partial}{\partial x_2} b \frac{\partial u}{\partial x_2} + cu. \tag{23}$$

We assume that the coefficients a, b satisfy the conditions of the previous section. We further assume that c is nonnegative and in $C^{0,\alpha}(\Omega)$.

We consider the following standard five-point finite difference scheme approximating the solution of (16): For $F \in S_{h_J}$ find $U \in S_{h_J}$ satisfying

$$L_{h_J} U = F,$$

where $L_{h_J} : S_{h_J} \mapsto S_{h_J}$ is given by

$$\begin{aligned}(L_{h_J} V)_{i,j} = {} & a_{i+1/2,j}(V_{i,j} - V_{i+1,j}) + a_{i-1/2,j}(V_{i,j} - V_{i-1,j}) \\ & + b_{i,j+1/2}(V_{i,j} - V_{i,j+1}) + b_{i,j-1/2}(V_{i,j} - V_{i,j-1}) + h_J^2 c_{i,j} V_{i,j}.\end{aligned} \tag{24}$$

Here we have used the generic notation $f_{s,t}$ to denote the value of $f(sh_J, th_J)$.

Consider the triangulation $\{\tau_J^i\}$ that results from breaking each square of the Jth finite difference grid into two triangles (for example, along the positive sloping diagonal). Let M_J be the space of continuous piecewise linear functions with respect to this triangulation that vanish on $\partial\Omega$. We consider a finite element approximation that uses a different quadrature on each term in (23). Let τ_J^i be a typical triangle in $\{\tau_J^i\}$ and let:

1. b_1^i denote the midpoint of the edge that is parallel to the x_1 axis;
2. b_2^i denote the midpoint of the edge that is parallel to the x_2 axis;
3. $\tilde{b}_l^i$ for $l = 1, 2, 3$ denote the vertices of the triangle.

We use the following three quadrature schemes, respectively, for the three terms coming from (23):

$$\int_{\tau_J^i} \phi \, dx \approx \frac{h_J^2}{2} \phi(b_1^i),$$

$$\int_{\tau_J^i} \phi \, dx \approx \frac{h_J^2}{2} \phi(b_2^i),$$

$$\int_{\tau_J^i} \phi \, dx \approx \frac{h_J^2}{6} [\phi(\tilde{b}_1^i) + \phi(\tilde{b}_2^i) + \phi(\tilde{b}_3^i)].$$

Let $A(\cdot, \cdot)$ be the resulting quadratic form on the space M_J. It is straightforward to check that the stiffness matrix associated with this finite element numerical quadrature scheme is equal to the matrix associated with the finite difference operator L_{h_J}. Even though we have used different quadrature rules in defining $A(\cdot, \cdot)$, it is elementary to verify that (10) holds without any conditions on h_1 or h_J. Furthermore, these quadratures are exact on constants. Consequently, Remark 3.1 implies that Theorem 3.1 holds for the variational multigrid algorithm using the finite element subspace imbedding formulas to define the finite difference prolongation operators. This is a standard variational multigrid algorithm. We have the following theorem.

THEOREM 4.1. *The variational form of the multigrid algorithm with prolongation corresponding to finite element imbedding applied to the finite difference operator L_{h_J} with Gauss–Seidel smoothing converges with a reduction that is independent of the number of grid levels and unknowns.*

Remark 4.1. It is easy to check that the stiffness matrix for $A(\cdot, \cdot)$ is a five-point stencil on every grid level. The same is true for the coarser grid finite difference stencils developed in the variational multigrid scheme using the finite element prolongations.

REFERENCES

[1] R. E. BANK AND T. DUPONT, *An optimal order process for solving finite element equations*, Math. Comp., 36 (1981), pp. 35–51.

[2] D. BRAESS AND W. HACKBUSCH, *A new convergence proof for the multigrid method including the V-cycle*, SIAM J. Numer. Anal., 20 (1983), pp. 967–975.

[3] J. H. BRAMBLE AND J. E. PASCIAK, *New convergence estimates for multigrid algorithms*, Math. Comp., 49 (1987) pp. 311–329.

[4] ———, *The analysis of smoothers for multigrid algorithms*, Math. Comp., 58 (1992), pp. 467–488.

[5] ———, *New estimates for multigrid algorithms including the v-cycle*, Math. Comp., 60 (1993), pp. 447–471.

[6] J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.

[7] J. H. BRAMBLE, J. E. PASCIAK, AND J. XU, *The analysis of multigrid algorithms with non-nested spaces or non-inherited quadratic forms*, Math. Comp., 56 (1991), pp. 1–34.

[8] ———, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[9] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[10] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, New York, 1978.

[11] N. DECKER, S. PARTER, AND J. MANDEL, *On the role of regularity in multigrid methods*, in Multigrid Methods, Proc. 3rd Copper Mountain Conference, S. McCormick, ed., Marcel Dekker, New York, 1988.

[12] C. I. GOLDSTEIN, *Multigrid methods for elliptic problems in unbounded domains*, SIAM J. Numer. Anal., 30 (1993), pp. 159–183.

[13] C. I. GOLDSTEIN, *Multigrid analysis of finite element methods with numerical integration*, Math. Comp., 56 (1991), pp. 409–436.

[14] W. HACKBUSCH, *Multi-Grid Methods and Applications*, Springer-Verlag, New York, 1985.

[15] J. L. LIONS AND E. MAGENES, *Problèmes aux Limites non Homogènes et Applications*, Dunod, Paris, 1968.

[16] J. F. MAITRE AND F. MUSY, *Algebraic formalization of the multigrid method in the symmetric and positive definite case—a convergence estimation for the V-cycle*, in Multigrid Methods for Integral and Differential Equations, D. J. Paddon and H. Holstien, eds., Clairdon Press, Oxford, 1985.

[17] J. MANDEL, S. MCCORMICK, AND R. BANK, *Variational multigrid theory*, in Multigrid Methods, S. McCormick, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987, pp. 131–178.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 577–586, May 1994

006

ON THE MULTILEVEL ADAPTIVE ITERATIVE METHOD*

U. RÜDE†

Abstract. The *multilevel adaptive iterative method* is a technique for solving the sparse matrix equations that typically arise from partial differential systems. Its core consists of a relaxation scheme and an *active set strategy*. The active set is used to monitor where the iteration efficiently reduces the error. It is incrementally updated by exploiting the current solution and the matrix structure, and most arithmetic operations are restricted to it. The algorithm can be extended to a multilevel structure by additionally tracing the dependencies between unknowns on different levels. It improves the robustness and efficiency of classical multilevel methods; in particular, it is an almost ideal supplement to adaptive refinement techniques.

Key words. sparse matrices, iterative methods, multilevel methods, adaptivity

AMS subject classifications. 65F10, 65F50, 65N22, 65N50, 65N55

1. Introduction.

With the *multilevel adaptive iterative method* we introduce an algorithmic concept that improves both the robustness and efficiency of iterative sparse system solvers. In most iterative methods, the sequence of operations is determined a priori and depends neither on the particular matrix structure nor on the current approximation of the solution. For example, consider the case of a finite element mesh constructed by successive adaptive refinement. Typically, the refinement process is combined with nested iteration such that the initial guess for the solution can be obtained from the solution of the previous system (see Bai and Brandt [1], Bank [2], and McCormick [11]). The initial guess will be close to the new solution in large parts of the domain, while it may be far from the solution in regions where the mesh has been refined. A classical iterative method does not exploit this situation and will therefore be inefficient.

The core of the multilevel adaptive method consists of relaxation supplemented with an *active set strategy*, as introduced in Rüde [14]. In each elementary step, we select an unknown from the active set and perform a preliminary relaxation step at it. If the new value differs from the old value by more than a *characteristic tolerance*, the update is performed, but if the change is small, the old value is retained. In either case, the current unknown is removed from the active set, because a further relaxation step for the same unknown would not provide any new result. In the case where the update has been executed, a further relaxation of the same unknown gives exactly the same value, and if the update has been rejected, a new relaxation would also be rejected. If and only if the update has been performed, the active set must be supplemented by the *neighbors* of the current unknown. For all of these unknowns the residual has changed, and it must therefore be checked to determine whether they must be relaxed. Thus, the active set may grow and shrink to keep track of where the solution is changing. The basic relaxation is repeated until the active set becomes depleted. The sequence of operations in this algorithm depends on the matrix structure as well as on the current state of the iteration. Computational work is concentrated where it can efficiently improve the solution.

To make the algorithm practical, the critical tolerance must be selected and adapted appropriately. In §2, it will be shown that the speed of convergence is directly related to the tolerance parameter used, but that the tolerance parameter necessary to obtain a certain accuracy depends on the condition of the system. Thus, the effectivity of the method depends on whether it can be integrated with preconditioning techniques. In §3, we extend the method to a multilevel structure by introducing a nested system of subspaces. This structure can be

*Received by the editors May 18, 1992; accepted for publication (in revised form) November 2, 1992.

†Institut für Informatik, Technische Universität München, Arcisstr. 21, D-80290 München 9, Germany (ruede@informatik.tu-muenchen.de).

derived naturally in many cases that originate from the discretization of differential equations, but could also be obtained directly from the system matrix, as in the *algebraic multigrid method*; see Ruge and Stüben [17]. It will be shown that the active set strategy can be extended to the multilevel context by tracing dependencies between unknowns on different levels.

This extended algorithm is related to multilevel algorithms and can be shown to have asymptotically optimal convergence for many important special cases. The analysis relies on the theoretical foundation of either multigrid theory (see Hackbusch [8], McCormick [10]) or the theory of multilevel preconditioning (see Bramble, Pasciak, and Xu [3], Dryja and Widlund [6], Oswald [12], Yserentant [19]).

The following presentation focuses on the model case of a positive definite linear system as it arises in the finite element discretization of elliptic second-order partial differential equations. This allows us to explain the process as a minimization algorithm, simplifying the presentation and permitting a simple analysis based on the existing multilevel theory.

2. Adaptive relaxation. Consider the linear system

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

where

$$A = (a_{i,j})_{1\le i,j\le n} \in \mathbb{R}^{n\times n}, \quad x = (x_i)_{1\le i\le n} \in \mathbb{R}^n, \quad b = (b_i)_{1\le i\le n} \in \mathbb{R}^n.$$

Assume that A is symmetric, positive definite, so that the solution of (1) is equivalent to finding the solution of

$$\min_{x\in\mathbb{R}^n}(x^T Ax - 2x^T b). \tag{2}$$

For each unknown i, define the set of its *neighbors* by

$$N(i) = \{j |\ j \neq i, a_{j,i} \neq 0\}, \tag{3}$$

that is the row indices of the nonzero entries in *column* i, excluding the diagonal. Due to symmetry, $N(i)$ is also the set of indices corresponding to nonzero entries in *row* i. The matrix is assumed to be *sparse*, that is, that the number of neighbors is small:

$$|N(i)| \le \bar{N} \ll n, \qquad 1 \le i \le n. \tag{4}$$

An *elementary relaxation* step for equation i consists of a modification of vector x by

$$x' = x + r_i, r_i = \tau_i e_i, \qquad \tau_i = a_{i,i}^{-1} e_i^T (b - Ax), \tag{5}$$

where e_i is the ith coordinate vector, (i.e., the ith column of the unit matrix) and $\tau_i = \tau_i(x)$ is called the *current scaled residual* of equation i (*dynamic residual*); see Brandt [4]. An elementary relaxation step for equation i can be understood as a coordinate descent for the ith coordinate direction applied to (2).

To evaluate the quality of an approximate solution we use the Euclidean norm, the maximum norm, or the *energy*-norm defined by

$$\|x\|_2 = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^2\right)^{1/2}, \quad \|x\|_\infty = \max_{i=1,\dots,n} |x_i|, \quad \|x\|_E = (x^T Ax)^{1/2},$$

respectively. These norms will be used to measure the error $x - x^*$, $x^* = A^{-1}b$ of an approximate solution x. The effect of a basic relaxation step is described by the following lemma.

LEMMA 2.1. *Let $x - x^*$ be the error before, and $x' - x^*$ be the error after, a basic relaxation step for equation i. Then the energy norm of the error is reduced according to*

$$\|x - x^*\|_E^2 - \|x' - x^*\|_E^2 = a_{i,i}\tau_i^2.$$

Proof. It holds that

$$\begin{aligned}
&\|x - x^*\|_E^2 - \|x' - x^*\|_E^2 \\
&\quad = (x - x^*)^T A(x - x^*) - (x + r_i - x^*)^T A(x + r_i - x^*) \\
&\quad = -2r_i^T A(x - x^*) - r_i^T A r_i \\
&\quad = a_{i,i}\tau_i^2. \qquad \square
\end{aligned}$$

This lemma is exploited in the Gauss–Southwell method [18], where the equation with largest current residual is selected for relaxation in each step. This method is often too expensive, however, because it requires the determination of the maximal residual in each step.

In our algorithm we therefore use weaker conditions. Consider the sets

$$S(\tau, x) = \{i \mid \ |\tau_i(x)| \geq \tau\}.$$

If relaxation could be restricted to these sets, Lemma 2.1 would guarantee a lower bound on the reduction of the energy norm of the error in each elementary relaxation step. Unfortunately, the sets $S(\tau, x)$ are still too expensive to compute. Therefore, we will use *active sets* $\tilde{S}(\tau, x)$ that are computed more efficiently as supersets of $S(\tau, x)$: $\tilde{S}(\tau, x) \supset S(\tau, x)$.

These active sets are the basis of the *sequential adaptive relaxation algorithm* in Fig. 1, whose main loop consists of an elementary relaxation step and an update of the active set. The input for the algorithm in Fig. 1 is the initial guess x, the tolerance τ, and the current active set $\tilde{S}$. The assertions document that the algorithm will only perform properly if $\tilde{S}$ is indeed a superset of $S(\tau, x)$. If this is initially satisfied, it stays a *loop invariant.* On termination, the active set is empty implying that no residuals beyond the critical tolerance τ remain.

```
proc SequentialAdaptiveRelaxation(τ, x, S̃)
        assert(τ > 0)
        assert(S̃ ⊃ S(τ, x))
        while(S̃ ≠ ∅)
                pick i ∈ S
                S̃ ← S̃ \ {i}
                if |τ_i(x)| > τ then
                        x ← x + r_i
                        S̃ ← S̃ ∪ N(i)
                end if
                assert(S̃ ⊃ S(τ, x))
        end while
end proc
```

FIG. 1. *Sequential adaptive relaxation.*

An analogous active set strategy can be used in a simultaneous (Jacobi-style) relaxation. This may be useful if the algorithm is to be combined with other components that need

symmetric operators, such as the conjugate gradient method. In the remainder of this paper, however, we will focus on successive algorithms.

A simple argument (see Rüde [16]) shows that the work is directly proportional to the gain in energy. We state this as a lemma.

LEMMA 2.2. *Assume that algorithm* SequentialAdaptiveRelaxation *is started with initial guess x, tolerance τ, and active set $\tilde{S} \supset S(\tau, x)$. Let x' denote its result. Then the number, W, of passes through the main loop is bounded by*

$$W \leq |S| + \left(\|x - x^*\|_E^2 - \|x' - x^*\|_E^2\right) \frac{\bar{N}}{\tau^2 a_{\min}}, \tag{6}$$

where $\bar{N}$ is the maximal number of neighbors defined in (4) *and*

$$a_{\min} = \min_{i=1,\ldots,m} a_{i,i}.$$

Proof. See Proposition 3.2 in [16]. □

Lemma 2.2 shows that the work depends on the tolerance parameter τ. This raises the question of which values of τ are suitable. We prepare the answer to this question with another lemma that is often useful in the analysis of iterative methods. It relates the error with the gain in one step of the iteration (see Rüde [13]).

LEMMA 2.3. *Let a linear iteration*

$$x^{k+1} = Cx^k + b, \qquad k = 1, 2, 3, \ldots,$$

with $\|C\| \leq c < 1$ be given, and assume that x^* is its fixed point:

$$x^* = (I - C)^{-1} b.$$

Then

$$\|x^* - x^0\| \leq \frac{1}{1 - c} \|x^1 - x^0\|.$$

Proof. It holds that

$$\begin{aligned} \|x^* - x^0\| &= \left\| \sum_{k=1}^{\infty} (x^k - x^{k-1}) \right\| \\ &\leq \|x^1 - x^0\| \sum_{k=1}^{\infty} \|C^k\| \\ &\leq \frac{1}{1 - c} \|x^1 - x^0\|. \end{aligned}$$

□

We can now apply this lemma to the adaptive algorithm as follows.

THEOREM 2.4. *Let $D = \operatorname{diag}(A)$ be the diagonal part of A and assume that*

$$\|I - D^{-1}A\|_2 \leq c < 1. \tag{7}$$

Denote by x' the result of algorithm SequentialAdaptiveRelaxation *called with input $\tau > 0$ and $\tilde{S} \supset S(\tau, x)$. Then*

$$\|x' - x^*\|_2 \leq \frac{\tau}{1 - c}. \tag{8}$$

Proof. On termination, $\tau_i(x') \leq \tau$ for all i. Therefore, letting

$$\bar{r} = \sum_{i=1}^{n} r_i,$$

we get $\|\bar{r}\|_2 \leq \tau$. Now consider the iterative process

$$y^{k+1} = Cy^k$$

for $k = 0, 1, 2, \ldots$, and where $C = I - D^{-1}A$ and the initial value is $y^0 = x' - x^*$. Thus, $\bar{r} = -D^{-1}Ay^0$, and the first step in the iteration becomes $y^1 = y^0 + \bar{r}$. This is just Jacobi's method applied to $Ay = 0$. Lemma 2.3 then implies

$$||x' - x^*||_2 = ||y^0 - 0||_2 \leq \frac{1}{1-c}||y^1 - y^0||_2 = \frac{||\bar{r}||_2}{1-c} \leq \frac{\tau}{1-c}. \qquad \square$$

For the typical case of second-order elliptic equations, $c = 1 - O(h^2)$, where h is the meshsize. In this case, τ must be chosen as $O(h^2\epsilon)$, where ϵ is the desired accuracy. Lemma 2.2 shows that the computational complexity is $O(h^{-4})$, just as for the straightforward Jacobi or Gauss–Seidel iteration. Thus, adaptive relaxation has no better complexity than standard iterative methods. Instead, it should be perceived as a *device to augment an arbitrary iterative method* when some kind of nonuniformity makes a localization of the operations desirable.

In practical multilevel applications there are many situations where this is useful. The multigrid rate of convergence deteriorates in the case of singularities at reentrant corners. For the biharmonic equation discretized as a system of Laplace equations, difficulties at the boundary lead to a much reduced rate of convergence. In all these cases, *local relaxation* (see Brandt [5]) or adaptive relaxation recovers the optimal performance of multigrid algorithms, so that even simple V-cycles converge with full efficiency.

3. Multilevel adaptive iteration. In the multilevel context the situation is different because relaxation is only used as a smoother, not as a solver. A smoother need not eliminate the error, but only make it well approximated by a system with fewer unknowns.

To keep the presentation simple, we restrict the discussion to Galerkin algorithms where the coarse level equations are derived by the so-called Galerkin condition. Let us assume that we are given integers $n_1 \leq n_2 \leq \cdots \leq n_K = n$ and a sequence of spaces

$$\mathbb{R}^{n_k}, \qquad k = 1, 2, \ldots, n_K$$

with *projections* $P_k : \mathbb{R}^{n_{k+1}} \longrightarrow \mathbb{R}^{n_k}$, $k = 1, 2, \ldots, K-1$. Let $k \in \{1, 2, \ldots, K\}$ and define the product projection

$$\bar{P}_k = \prod_{\kappa=1}^{K-k} P_{K-\kappa}$$

that maps $\mathbb{R}^n$ to any of its subspaces $\mathbb{R}^{n_k}$. For $k = K$ we take this definition to mean $\bar{P}_K = I$, the identity on $\mathbb{R}^{n_K}$. The transpose P_k^T of P_k will be an *interpolation* operator. Similar to Griebel [7], we represent a vector $x \in \mathbb{R}^n$ *nonuniquely* as

$$x = \sum_{k=1}^{K} \bar{P}_k^T x_k,$$

where $x_k \in \mathbb{R}^{n_k}$, $k = 1, \ldots, K$. The minimization problem (2) then becomes

$$\min_{x_k \in \mathbb{R}^{n_k},\ k=1,\dots,K} \left(\left(\sum_{k=1}^{K} \bar{P}_k^T x_k \right)^T A \left(\sum_{k=1}^{K} \bar{P}_k^T x_k \right) - 2 \left(\sum_{k=1}^{K} \bar{P}_k^T x_k \right)^T b \right). \tag{9}$$

Multilevel algorithms usually focus on *one* level at a time, so that all but one x_k are kept fixed at any given time. This results in an equation for x_k of the form

$$\bar{P}_k A \bar{P}_k^T x_k = \bar{P}_k \left(b - A \sum_{\kappa=1,\ \kappa \neq k}^{K} \bar{P}_\kappa x_\kappa \right). \tag{10}$$

These systems are treated consecutively for $k = 1, \dots, K$. We are now interested in adaptive relaxation as an elementary process for each such level.

A complete theory of multilevel algorithms is beyond the subject of this paper. However, as an example of how theoretical results for multilevel algorithms can be used to analyze the multilevel adaptive iterative method, we derive some results based on a theorem of Zhang [20].

THEOREM 3.1. *Assume that the matrices A_k, $1 \le k \le K$, $A_K = A$, originate from a discretization of Poisson's equation on a nested sequence of uniformly refined, shape regular triangulations $\mathcal{T}_k$ of a plane, polygonal domain with linear elements. Further, let P_k^T be the (discrete representations of the) associated interpolation operators. Then there exist constants c_1, c_2, independent of K, such that for all $x \in \mathbb{R}^n$,*

$$\frac{c_1}{K} x^T A x \le x^T A \left(\sum_{\kappa=1}^{K} \bar{P}_k^T D_k^{-1} \bar{P}_k \right) A x \le c_2 x^T A x, \tag{11}$$

where $D_k = \operatorname{diag}(A_k)$ is the diagonal part of A_k. If the original problem has H^2-regularity, the lower bound can be improved to be independent of K.

Proof. See Theorem 3.1 of Zhang [20]. □

Theorem 3.1 has been proved only for the model case, so that our following argument is formally also restricted to this particular situation. Similar results for more general situations have been obtained by Bramble, Pasciak, and Xu [3], Dryja and Widlund [6], Oswald [12], and Yserentant [19], and can be applied in a similar fashion.

We define the *current scaled residual* of $x \in \mathbb{R}^n$ for any level $k = 1, 2, \dots, K$ as

$$\bar{r}_k = D_k^{-1} \bar{P}_k (b - Ax). \tag{12}$$

COROLLARY 3.2. *Under the hypothesis of Theorem* 3.1,

$$\|x - x^*\|_2 \le \frac{K}{c_1} \left\| \sum_{k=1}^{K} \bar{P}_k^T \bar{r}_k \right\|_2. \tag{13}$$

Proof. Theorem 3.1 implies that

$$\frac{c_1}{K} \le \frac{v^T A^{1/2} \left(\sum_{\kappa=1}^{K} \bar{P}_k^T D_k^{-1} \bar{P}_k \right) A^{1/2} v}{v^T v} \le c_2,$$

for all $v \in \mathbb{R}^n$. Thus the eigenvalues of the preconditioned matrix

$$\left(\sum_{\kappa=1}^{K} \bar{P}_k^T D_k^{-1} \bar{P}_k \right) A \tag{14}$$

satisfy

$$\rho\left(\sum_{k=1}^{K} \bar{P}_k^T D_k^{-1} \bar{P}_k A\right) \in \left[\frac{c_1}{K}, c_2\right].$$

Since

$$x - x^* = \left(\sum_{k=1}^{K} \bar{P}_k^T D_k^{-1} \bar{P}_k A\right)^{-1} \left(\sum_{k=1}^{K} \bar{P}_k^T \bar{r}_k\right),$$

then (13) follows. □

Theorem 3.1 can also be used to derive *lower bounds* on the L_2 error. Furthermore, bounds on the *energy norm* of the error can be constructed. Corollary 3.2 states that if the scaled residuals *on all levels* are small, then the error must be small. Therefore, we need not consider elementary relaxations with very small residuals. To obtain accuracy ϵ, relaxation need only reduce the residuals on each level to the point that they satisfy $\|\bar{r}_k\|_\infty \leq \epsilon_k$, where the ϵ_k are chosen such that

$$\sum_{k=1}^{K} \epsilon_k = O\left(\frac{\epsilon}{K}\right).$$

Thus, since adaptive relaxation can therefore be used with *large* critical tolerance values, we can now construct efficient solvers.

Unfortunately, the most straightforward idea of using adaptive relaxation directly for the preconditioned system (14) is not satisfactory. This difficulty arises because the preconditioned matrix is not sparse in the sense of (4). Analogous to a regular multilevel algorithm, we must instead use a *factored form*, where the levels are treated consecutively and independently, while trying to minimize the number of interlevel transfers. Fortunately, this can be implemented by the successive application of sequential adaptive relaxation to the systems of (10) for $k = 1, \ldots, K$.

This raises the question of how the active sets should be initialized on each level. For level k, we define

$$S_k(\bar{\tau}_k, x) = \left\{i | 1 \leq i \leq n_k,\ |e_i^T \bar{r}_k| > \bar{\tau}_k\right\}, \tag{15}$$

where $\bar{r}_k$ is the current scaled residual of level k as defined in (12). The trivial construction of a superset $\bar{S}_k \supset S_k(\bar{\tau}_k, x)$ is done by setting by $\bar{S}_k = \{1, 2, \ldots, n_k\}$, but this neglects useful information. To exploit locality, we should attempt to transfer information about the active sets between the levels. This is done essentially as if adaptive relaxation were applied to the semidefinite system (9); however, for efficiency, interlevel information is collected and transferred only when the algorithm actually switches between levels.

To give a concise description of the process we introduce the following notation. For a matrix

$$M = (\mu_{i,j}) \in \mathbb{R}^{m_1} \times \mathbb{R}^{m_2}$$

and a set $S \subset \{1, 2, \ldots, m_2\}$ of indices, let MS denote the set of indices defined by

$$MS = \{i |\ 1 \leq i \leq m_1,\ \exists j \in S : \mu_{i,j} \neq 0\}.$$

If sets are identified with Boolean vectors, MS denotes a Boolean matrix multiplication. In this notation, N, as defined in (3) and extended to be set-valued, could be written as $N = A - D$,

where D is the diagonal part of the system matrix A. Note that for two matrices A and B we only have $(AB)S \subset A(BS)$, not necessarily equality.

In Fig. 2 we display a generic multilevel adaptive iteration routine. The inner loop is precisely the adaptive relaxation applied on the different levels k. It is augmented with sets U_k and V_k to provide information where the relaxation causes changes that must be propagated to higher or lower levels. It is left unspecified whether "upward" or "downward" is selected. In either direction, the information accumulated in the set U_k (and V_k, respectively) is exploited to initialize the active set on the new level. These sets can also be used to *localize* the projection and interpolation operations. Furthermore, the set U (respectively, V) is updated so that information will be passed successively through all levels.

```
proc MLAI(x, S̄_1, τ̄_1, S̄_2, τ̄_2, ..., S̄_K, τ̄_K)
    assert(τ̄_k > 0,  1 ≤ k ≤ K)
    assert(S̄_k ⊃ S_k(τ̄_k, x),  1 ≤ k ≤ K)
    U_k ← ∅, V_k ← ∅, for k = 1, 2, ..., K
    k ← 1
    while(⋃_{k=1}^{K} S̄_k ≠ ∅)
        while(S̄_k ≠ ∅)
            pick i ∈ S̄_k
            S̄_k ← S̄_k \ {i}
            if |e_i^T r̄_k| > τ̄_k then
                x_k ← x_k + (e_i^T r̄_k) e_i
                S̄_k ← S̄_k ∪ N_k(i)
                U_k ← U_k ∪ {i}
                V_k ← V_k ∪ {i}
            end if
            assert(S̄_k ⊃ S_k(τ̄_k, x))
        end while
        if "upward" then
            k ← k + 1
            U_k ← U_k ∪ P_{k-1}^T U_{k-1}
            S̄_k ← S̄_k ∪ A_k(P_{k-1}^T U_{k-1})
            U_{k-1} ← ∅
        else "downward"
            k ← k − 1
            V_k ← V_k ∪ P_{k+1} V_{k+1}
            S̄_k ← S̄_k ∪ A_k(P_{k+1} V_{k+1})
            V_{k+1} ← ∅
        end if
        assert(S̄_k ⊃ S_k(τ̄_k, x))
    end while
end proc
```

FIG. 2. *Multilevel adaptive iteration.*

Several different cycling strategies can be used. One possibility is a V-cycle, or W-cycle, as in classical multigrid methods. Additionally or alternatively, the number of elements in the active sets can be used to adapt the cycling strategy; see Rüde [16].

Certain variants of the above algorithm can be analyzed based on multigrid convergence theory. This possibility has been suggested by Leinen and Yserentant [9]. For this purpose, the adaptive relaxation on level k is slightly modified such that it can be interpreted as one or

more sweeps of regular, global relaxation on that level, perturbed by quantities of size $O(\bar{\tau}_k)$. The effect of one sweep of sequential adaptive relaxation starting with x_k and $\bar{\tau}_k$ on level k can therefore be described by

$$x' - x^* = \left((x - x^*) - \bar{P}_k^T (G_k \bar{P}_k A(x - x^*) + \mu_k)\right),$$

where G_k is the inverse of the lower triangular part of A_k and $\mu_k \in \mathbb{R}^{n_k}$ satisfies $\|\mu_k\|_\infty \leq \bar{\tau}_k$.

If a V-cycle with a single sweep of presmoothing is used, it suffices to argue that subsequent corrections within the same cycle cannot enlarge the error because all operators have norm smaller than 1. In particular, they cannot enlarge the μ_k perturbations, so that their total effect must be bounded by $\sum_{k=1}^{K} \bar{\tau}_k$. Thus, these variants of the multilevel adaptive iterative method have (almost) the same convergence properties as classical multigrid, but can be substantially cheaper.

The termination criterion in Fig. 2 guarantees that the final result will be accurate, provided the assumptions of Theorem 3.1 or their equivalents are satisfied. These assumptions assert that we have a suitable multilevel decomposition of the solution space, which, for classical multilevel algorithms, is the core feature that makes the algorithm work. The adaptive relaxation additionally improves robustness and efficiency.

Furthermore, and equally important, diagnostic information is automatically available within the algorithm. The relaxation within a level usually does not introduce large errors on other levels, because the corresponding components of the solution space are approximately orthogonal. If the orthogonality is severely violated, it becomes obvious in the performance of the algorithm. Cycling between levels will not efficiently reduce the size of the active sets, because removal of errors on one level feeds errors on another level. This indicates that the space decomposition is not optimal, the estimates of Theorem 3.1 are violated (or hold with large c_2/c_1 only), and thus also the error bound for the solution will be poor as determined by Corollary 3.2. In this case the setup of the multilevel hierarchy (that is the system of projection and prolongation operators) has not been successful and must be modified. With the adaptive iterative method, these troubles can be easily monitored. This makes the development of truly robust software possible.

Of course, the practical relevance of the *multilevel adaptive iterative method* depends on whether the necessary operations can be implemented efficiently. As discussed in Rüde [15], [16], this can be accomplished if *small sets* whose cardinality is bounded by a constant are implemented as lists. Typical small sets are $N(i)$. *Large sets*, like $\bar{S}$ or U and V must be represented by flags *and* lists. This is necessary so that the set operations can be performed efficiently.

4. Conclusions. The *multilevel adaptive iterative method* is a framework that can be modified and extended in many ways. The basic idea can be extended easily to nonsymmetric and even nonlinear problems. Moreover, the *multilevel adaptive iterative method* seems to be an almost ideal supplement of adaptive mesh refinement. Rüde [16] introduced a special mesh refinement technique, which is the so-called *virtual global grid*. Adaptive iteration by itself does not waste work for unknowns that are already well approximated by interpolation from a coarser grid. In this sense it can be shown that adaptive relaxation has the properties of an error estimator. This is exploited by the virtual global refinement technique. It relies on the adaptive iteration as a solver and error estimator. Grids are always refined globally; however, memory allocation for nodes is deferred until the node is really accessed and modified by an adaptive relaxation. Unknowns are therefore virtual entities that are physically created only when required by the algorithm. In this sense the multilevel adaptive iterative method can be seen as an algorithm to solve the linear systems when the number of levels tends to infinity

and where the solution process is driven only by the accuracy requirement. The fully adaptive multigrid method is thus a true partial differential equation solver.

Acknowledgment. The author thanks Steve McCormick and the referees for several helpful comments.

REFERENCES

[1] D. BAI AND A. BRANDT, *Local mesh refinement multilevel techniques*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 109–134.

[2] R. BANK, PLTMG: *A Software Package for Solving Elliptic Partial Differential Equations User's Guide* 6.0, in Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[3] J. BRAMBLE, J. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 31 (1990), pp. 333–390.

[4] A. BRANDT, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–65.

[5] ———, *Rigorous local mode analysis of multigrid*, in Preliminary Proc. 4th Copper Mountain Conference on Multigrid Methods, April 9–14, 1989, S. McCormick, ed., University of Colorado at Denver, 1989.

[6] M. DRYJA AND O. WIDLUND, *Multilevel additive methods for elliptic finite element problems*, in Parallel Algorithms for Partial Differential Equations, Proc. 6th GAMM-Seminar, Kiel, Jan. 19–21, 1990, W. Hackbusch, ed., Vieweg-Verlag, Braunschweig, 1991.

[7] M. GRIEBEL, *Multilevel algorithms considered as iterative methods on indefinite systems*, SFB Bericht 342/29/91 A, Institut für Informatik, TU München, Oct. 1991; Also, Preliminary Proc. 2nd Copper Mountain Conference on Iterative Methods, University of Colorado, Denver, CO, 1992.

[8] W. HACKBUSCH, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[9] P. LEINEN AND H. YSERENTANT, *Private communication*, 1991.

[10] S. MCCORMICK, ED., *Multigrid Methods*, Vol. 3, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[11] S. MCCORMICK, *Multilevel Adaptive Methods for Partial Differential Equations*, Vol. 6, Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[12] P. OSWALD, *Two remarks on multilevel preconditioners*, Forschungsergebnisse Math/91/1, Friedrich Schiller Universität, Jena, 1991.

[13] U. RÜDE, *Multiple tau-extrapolation for multigrid methods*, Bericht I-8701, Institut für Informatik, TU München, Jan. 1987.

[14] ———, *Adaptive higher order multigrid methods*, in Proc. 3rd European Conference on Multigrid Methods, Oct. 1–4, 1990, W. Hackbusch and U. Trottenberg, eds., Basel, 1991, Birkhäuser, pp. 339–351. International Series of Numerical Mathematics, Vol. 98.

[15] ———, *Data structures for multilevel adaptive methods and iterative solvers*, Bericht I-9217, Institut für Informatik, TU München, May 1992.

[16] ———, *Fully adaptive multigrid methods*, SIAM J. Numer. Anal., 30 (1993), pp. 230–248.

[17] J. RUGE AND K. STÜBEN, *Efficient solution of finite difference and finite element equations by algebraic multigrid* (AMG), Arbeitspapiere der GMD, 89 (1984).

[18] R. SOUTHWELL, *Relaxation Methods in Theoretical Physics*, Claredon Press, Oxford, 1946.

[19] H. YSERENTANT, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

[20] X. ZHANG, *Multilevel Additive Schwarz Methods*, Tech. Report 582, Department of Computer Science, Courant Institute, New York University, 1991.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 587–603, May 1994

007

MULTIPLICATIVE SCHWARZ METHODS FOR PARABOLIC PROBLEMS*

XIAO-CHUAN CAI†

Abstract. The class of multiplicative Schwarz methods originated from the classical Schwarz alternating method. It has been shown to be one of the most powerful methods for solving finite element or finite difference elliptic problems. In this paper, these methods are extended to a class of singularly perturbed equations that are encountered when discretizing parabolic equations by implicit methods such as the backward Euler or Crank–Nicolson schemes. Several algorithms are discussed, including one-level, two-level, and multilevel overlapping methods. The authors also study how the convergence rates depend on the time and space discretization parameters, as well as subspace decomposition parameters such as the number of subregions and the number of levels to which the finite element space is decomposed. It is shown that in the presence of a fine enough coarse mesh space the algorithms are optimal for both symmetric and nonsymmetric problems, i.e., the convergence rates are independent of all these parameters in both two and three dimensions. If the coarse mesh space is dropped, the algorithms are still optimal but only if the timestep and the coarse mesh size satisfy certain relationships.

Key words. overlapping domain decomposition, multilevel, optimal convergence, parabolic equation, finite elements

AMS subject classifications. 65N30, 65F10, 65N55

1. Introduction.

In a pioneer work of Lions [20] on domain decomposition methods, the classical Schwarz alternating algorithm [25] was extended successfully to a class of parabolic equations. The basic idea is to divide the region into several overlapping subregions and then to solve the parabolic problem in each subregion alternatively with boundary information from the neighboring subregions. In this paper, we further extend this idea to the case of many overlapping subregions, as well as to many levels of overlapping subregions. In this case, at each timestep many independent subproblems are being created. Thus parallel computers can be used more efficiently.

The focus of this paper is the study of the convergence rates and their dependence on the time and space discretization parameters on the subspace decomposition parameters, i.e., the number of subregions and the number of levels into which the finite element space is decomposed. Most of the techniques that we use in this paper are borrowed from the abstract theory of the additive and multiplicative Schwarz methods for elliptic equations; see, e.g., [3], [7], [8], [12], [13], and references therein. The additive version of some of the algorithms of this paper has previously been considered by the author in [4]. With a coarse mesh space, we show that under basically the same assumptions as for the additive Schwarz methods [4], the multiplicative algorithms converge with optimal rates that are independent of the time and space discretization parameters, the number of subregions, and the number of levels into which the space is decomposed. In contrast to the Schwarz methods for elliptic problems in which the coarse mesh space plays an essential role for the optimality, we prove that under the assumption that τ/H^2 is reasonably small, the algorithms remain optimal even if the coarse mesh space is eliminated. Here τ is on the order of the timestep size and H is the diameter of the largest substructure.

Some computational aspects of the algorithms have been studied extensively in the context of solving elliptic problems; see, e.g., the recent paper of Cai, Gropp, and Keyes [6]. Other domain decomposition based algorithms that deal with parabolic problems have recently been developed [2], [9], [10], [16], and [19], and references therein.

*Received by the editors May 28, 1992; accepted for publication (in revised form) January 26, 1993.

†Department of Computer Science, University of Colorado, Boulder, Colorado 80309 (cai@schwarz.cs.colorado.edu). This research was supported in part by National Science Foundation and Kentucky Experimental Program to Stimulate Competitive Research (EPSCoR) grant STI-9108764.

The paper is organized as follows. In the remainder of this section, we present the continuous and discrete parabolic equations and some of their basic properties. In §2, we discuss four overlapping decompositions of the finite element space, including a one-level decomposition, a two-level decomposition, and two multilevel decompositions and prove the uniformly boundedness of these decompositions. Four algorithms are introduced in §3 and their convergence rates are also analyzed. Finally, in §4, we apply these algorithms to some parabolic problems, including a time-dependent convection-diffusion equation. Throughout this paper, c and C, with or without subscripts, denote generic, strictly positive constants. Their values may be different at different occurrences, but are independent of the time and space discretization parameters, as well as the subspace decomposition parameters that will be introduced later as we move along.

Let $\Omega \subset R^d$ be a bounded polygonal region and $d = 2$, or 3. We are interested in the finite element solution of the following problem: Find $u \in H_0^1(\Omega)$, such that

$$d_\tau(u, \phi) \equiv \tau b(u, \phi) + (u, \phi) = (f, \phi) \quad \forall \phi \in H_0^1(\Omega). \tag{1}$$

Here $\tau > 0$ is a small number, which will be specified in §4 of this paper. The bilinear form $b(\cdot, \cdot)$ is defined as

$$b(u, v) = \sum_{i,j=1}^{d} \int_\Omega a_{ij}(x) \frac{\partial u}{\partial x_i} \frac{\partial v}{\partial x_j} dx + \sum_{i=1}^{d} \int_\Omega b_i(x) \frac{\partial u}{\partial x_i} v dx + \int_\Omega c(x) uv dx \tag{2}$$

and $(\cdot, \cdot)$ is the usual $L^2(\Omega)$ inner product. We assume that all coefficients are sufficiently smooth and the matrix $\{a_{ij}(x)\}$ is symmetric and uniformly positive definite. We also assume that the bilinear form $b(\cdot, \cdot)$ is bounded and positive definite, though not necessarily selfadjoint, i.e.,

(a) $|b(u, v)| \leq C\|u\|_{H_0^1(\Omega)} \|v\|_{H_0^1(\Omega)} \quad \forall u, v \in H_0^1(\Omega)$.

(b) $b(u, u) \geq c\|u\|^2_{H_0^1(\Omega)} \quad \forall u \in H_0^1(\Omega)$.

We use an $H_0^1(\Omega)$ equivalent norm, denoted by $\|\cdot\|_a$, defined by $a(u, v) = \frac{1}{2}(b(u, v)+b(v, u))$. In addition, we define the bilinear forms $a_\tau(u, v) = \frac{1}{2}(d_\tau(u, v) + d_\tau(v, u))$, which is symmetric positive definite, and $n_\tau(u, v) = \frac{1}{2}(d_\tau(u, v) - d_\tau(v, u))$, which is skewsymmetric. It is not difficult to see that the norm $\|\cdot\|_{a_\tau}$, defined by $a_\tau(\cdot, \cdot)$, is equivalent to the norm $(\|\cdot\|^2_{L^2(\Omega)} + \tau\|\cdot\|^2_{H_0^1(\Omega)})^{1/2}$. As an immediate consequence of the above assumptions, we have that the bilinear form $d_\tau(\cdot, \cdot)$ is bounded and positive definite in the $\|\cdot\|_{a_\tau}$ norm and that $n_\tau(\cdot, \cdot)$ is bounded: There exists a constant C, such that

$$| n_\tau(u, v) | \leq C\tau \|u\|_{H_0^1(\Omega)} \|v\|_{L^2(\Omega)} \quad \forall u, v \in H_0^1(\Omega).$$

It can be shown that $b(u, v)$ is $H^{1+\alpha}(\Omega)$-regular [17], [21]: For any $g \in L^2(\Omega)$ there exists a unique $u \in H^{1+\alpha}(\Omega) \cap H_0^1(\Omega)$, such that

$$b(v, u) = (g, v) \quad \forall v \in H_0^1(\Omega)$$

and

$$| u |_{H^{1+\alpha}(\Omega)} \leq C\|g\|_{L^2(\Omega)},$$

where α is at least $\frac{1}{2}$. We note that the convergence rates of some algorithms that are developed in this paper depend on the value α.

Let V^h be the usual piecewise linear conforming finite element subspace of $H_0^1(\Omega)$. The standard Galerkin approximation of (1) can then be defined by the following problem: Find $u_h^* \in V^h$, such that

$$d_\tau(u_h^*, \phi_h) = (f, \phi_h) \quad \forall \phi_h \in V^h. \tag{3}$$

In the next section, we shall formally introduce the space V^h and then decompose it into the sum of certain subspaces. Related multiplicative Schwarz methods will then be introduced to solve (3). The main purpose of this paper is the study of the convergence rates of these algorithms.

2. Overlapping decompositions and the stability analysis. In this section, we describe some uniformly overlapping subspace decompositions previously introduced by Dryja and Widlund [11], [14] for solving elliptic problems. These are the one-level, two-level, and multilevel overlapping decompositions. We will also show that these decompositions are uniformly, or nearly uniformly, bounded in the τ dependent norm $\|\cdot\|_{a_\tau}$. In the multilevel case, the uniformity is also with respect to the number of levels.

2.1. One- and two-level decompositions. Both decompositions have been discussed in [4], and the boundedness of the one-level decomposition is discussed only for the case $d = 2$. Here, we cover both $d = 2$ and 3.

Let $\{\Omega_i\}_{i=1}^M$ be a shape regular finite element triangulation, or a coarse mesh, of Ω and Ω_i has diameter of order H. In our second step, we further divide each substructure Ω_i into smaller simplices with diameter of order h. We assume that the resulting elements form a shape regular finite element subdivision of Ω. We call this the fine mesh or the h-level subdivision of Ω with mesh parameter h. Let us denote by $V^H \subset H_0^1(\Omega)$ and $V^h \subset H_0^1(\Omega)$ the continuous, piecewise linear function spaces, with zero trace on $\partial\Omega$, over the H-level and h-level subdivisions of Ω, respectively.

To obtain an overlapping decomposition, we extend each subregion Ω_i to a larger region Ω_i^{ext} such that $\Omega_i \subset \Omega_i^{\text{ext}} \subset \Omega$. Moreover, we assume that distance$(\partial\Omega_i^{\text{ext}} \cap \Omega, \partial\Omega_i \cap \Omega) \geq cH$ for all i. We suppose that $\partial\Omega_i^{\text{ext}}$ does not cut through any h-level elements.

Associated with the decomposition $\{\Omega_i^{\text{ext}}\}_{i=1}^M$, we define an undirected graph in which the nodes represent the extended subdomains and the edges intersections of the extended subdomains. This graph can be colored, using colors $1, \ldots, J$, such that no adjacent nodes have the same color. We regard the union of all subdomains with the same color as one subdomain (which is of course not simply connected), and denote them by $\Omega_1', \ldots, \Omega_J'$. We also denote $\Omega_0' = \Omega$ which has color 0. It is important to note that $J+1$, the number of colors, can be made to be independent of M, the number of substructures.

For each Ω_j', $1 \leq j \leq J$, we define $V_j = \{v \in V^h | \, v(x) = 0, \, x \bar{\in} \Omega_j'\} \subset V^h$. We also use the subspace $V_0 = V^H$. It is easy to see that we have two decompositions of V^h, i.e.,

$$V^h = V_1 + \cdots + V_J \tag{4}$$

and

$$V^h = V_0 + V_1 + \cdots + V_J, \tag{5}$$

which shall be referred as the one-level and two-level uniformly overlapping decompositions of V^h, or as the decomposition without and with the coarse mesh space, respectively.

It was proved in [4] that the two-level decomposition (5) is uniformly bounded in the $\|\cdot\|_{a_\tau}$ norm in both two- and three-dimensional spaces.

LEMMA 2.1. *There exists a constant C_0, such that for any $v \in V^h$, there exist $v_i \in V_i$, $v = v_0 + \cdots + v_J$, and*

$$\|v_0\|_{a_\tau}^2 + \sum_{i=1}^{J} \|v_i\|_{a_\tau}^2 \le C_0^2 \|v\|_{a_\tau}^2 \quad \forall v \in V^h. \tag{6}$$

Here the constant C_0 is independent of τ, h, and H.

If we drop the coarse mesh space V_0 from (5) and v_0 from the decomposition, the following estimate holds for both $d = 2$ and 3. The case $d = 2$ was discussed in [4].

LEMMA 2.2. *There exists a constant C_τ, such that for any $v \in V^h$, there exist $v_i \in V_i$, $v = v_1 + \cdots + v_J$, and*

$$\sum_{i=1}^{J} \|v_i\|_{a_\tau}^2 \le C\left(1 + \frac{\tau}{H^2}\right) \|v\|_{a_\tau}^2 \equiv C_\tau^2 \|v\|_{a_\tau}^2 \quad \forall v \in V^h, \tag{7}$$

where $C > 0$ is independent of τ, h, and H.

Proof. Let $\{\theta_i\}_{i=1}^J$ be a partition of unity and θ_i belongs to $C_0^\infty(\Omega_i' \cap \Omega)$. It can be chosen so that $|\nabla \theta_i|$ is bounded by C/H. Let I_h be the piecewise linear interpolation operator that uses the function values at the h-level nodes. For any $v \in V^h$, we let $v_i = I_h(\theta_i v) \in V_i$. For each subregion Ω_i', it is well known, in the sense of equivalent norms, that

$$|v_i|_{H_0^1(\Omega_i')}^2 = \sum ((\theta_i v)(x_l) - (\theta_i v)(x_m))^2 h^{d-2}, \tag{8}$$

where the sum is taken over all adjacent pairs of h-level nodal points in Ω_i'. Let $K \subset \Omega_i'$ be a single h-level triangle and x_l, $x_m \in \bar{K}$ be two of its vertices. Let $\bar{\theta}_{ilm} = (\theta_i(x_l) + \theta_i(x_m))/2$. We then have

$$\begin{aligned}(\theta_i v)(x_l) - (\theta_i v)(x_m) = (\theta_i(x_l) - \bar{\theta}_{ilm}) v(x_l) - (\theta_i(x_m) - \bar{\theta}_{ilm}) v(x_m) \\ + \bar{\theta}_{ilm}(v(x_l) - v(x_m)),\end{aligned}$$

which can be bounded from above by

$$C\left(\frac{h}{H} \max_{x \in \bar{K}} \{|v(x)|\}\right) + |v(x_l) - v(x_m)|.$$

By squaring this estimate, using the inequality $ab \le \frac{1}{2}(a^2 + b^2)$, summing over all $K \subset \Omega_i'$ and using (8), we obtain

$$\begin{aligned}|v_i|_{H_0^1(\Omega_i')}^2 &\le C\left(H^{-2} \sum_{K \subset \Omega_i'} \max_{x \in \bar{K}} \{|v(x)|^2\} \cdot h^d + |v|_{H^1(\Omega_i')}^2\right) \\ &\le C\left(H^{-2} \|v\|_{L^2(\Omega_i')}^2 + |v|_{H^1(\Omega_i')}^2\right).\end{aligned}$$

Therefore,

$$\sum_{i=1}^{J} \|v_i\|_{H_0^1(\Omega_i')}^2 \le C\left(H^{-2} \|v\|_{L^2(\Omega)}^2 + |v|_{H_0^1(\Omega)}^2\right) \le C H^{-2} \|v\|_{H_0^1(\Omega)}^2. \tag{9}$$

We refer to [4] for the L^2 estimate, i.e.,

$$\sum_{i=1}^{J} \|v_i\|^2_{L^2(\Omega_i')} \leq C\|v\|^2_{L^2(\Omega)}. \tag{10}$$

The proof follows immediately from the estimates (9) and (10). □

Remark 2.1. It is known that in the elliptic case, which corresponds to the use of the $\|\cdot\|_a$ norm in the estimate, if the coarse mesh space is dropped, a factor of $1/H^2$ would appear in the estimate and this makes this decomposition not so useful. However, as shown above, in the parabolic case, only a factor of τ/H^2 appears in the estimate and τ is usually in the order of the timestep size.

2.2. Multilevel decompositions. Following Dryja and Widlund [14] and Zhang [27], we let $\{\mathcal{T}^l\}_{l=0}^{L}$ be a nested sequence of triangulations of Ω, i.e., $\mathcal{T}^0 = \{\tau_i^0\}_{i=1}^{N_0}$ is the initial coarse triangulation and $\mathcal{T}^l = \{\tau_i^l\}(l = 1, \ldots, L)$ is defined by dividing each triangle of $\mathcal{T}^{l-1}$ into several triangles. We assume that all the triangulations are shape regular. Let $h_i^l = \operatorname{diam}(\tau_i^l)$, $h_l = \max_i\{h_i^l\}$, $H = \max_i h_i^0$, and $h = h_L$. We also assume that there exists $0 < r < 1$, such that h_l is proportional to Hr^l, for $l = 0, \ldots, L$. Let V^l be the usual conforming finite element space of continuous piecewise linear functions associated with the triangulation $\mathcal{T}^l$.

We construct L sets of overlapping subdomains $\{\hat{\Omega}_i^l\}_{i=1}^{J_l}, l = 1, 2, \ldots, L$; i.e., for each $1 \leq l \leq L$, we have

$$\Omega = \cup_{i=1}^{J_l} \hat{\Omega}_i^l.$$

Subdomains $\hat{\Omega}_i^l$ are defined as follows. Each triangle $\tau_i^{l-1}, i = 1, \ldots, N_l, l = 1, \ldots, L$, is extended to a larger region $\hat{\tau}_i^{l-1}$ so that

$$ch_i^{l-1} \leq \operatorname{dist}(\partial\hat{\tau}_i^{l-1} \cap \Omega, \partial\tau_i^{l-1} \cap \Omega) \leq Ch_i^{l-1}$$

aligning $\partial\hat{\tau}_i^{l-1}$ with the boundaries of level l triangles. For each l, we color the $\hat{\tau}_i^{l-1}$ by using J_l colors, such that all substructures of the same color are disjoint. Here J_l is a fixed constant depends only on the triangulations. On each level l, we group the extended subregions $\hat{\tau}_i^{j-1}$ by colors and obtain J_l sets of subregions. We denote by $\hat{\Omega}_i^l$ as the union of l-level extended subregions that share the same color $1 \leq i \leq J_l$. We denote $J = \max_l J_l$.

We define $V_i^l = \{v \in V^l |\ v(x) = 0,\ x \bar{\in} \hat{\Omega}_i^l\} \subset V^l$. Let $J_0 = 1$ and $V^0 = V_1^0 = V^H$. Thus, our finite element space $V^h = V^L$ can be represented as

$$V^L = V^0 + \sum_{l=1}^{L} V^l = V^0 + \sum_{l=1}^{L}\sum_{i=1}^{J_l} V_i^l \tag{11}$$

or

$$V^L = \sum_{l=1}^{L} V^l = \sum_{l=1}^{L}\sum_{i=1}^{J_l} V_i^l. \tag{12}$$

These two decompositions are referred to as the multilevel decomposition with and without a coarse mesh space, respectively. We note that the main difference between these two decompositions is that in (11) the coarsest mesh space is not decomposed into local subspaces. This works well if the degrees of freedoms involved in the coarse mesh problem are few. However, this is not always satisfied, especially for nonsymmetric problems where the coarse

mesh needs to be sufficiently fine. In the latter case, it is desirable to further decompose the coarse mesh problem and therefore (12) is sometimes more useful.

It is known that the decomposition with the coarse mesh (11) is uniformly bounded in the $\|\cdot\|_a$ norm, i.e., for any $v \in V^h$, there exist $v_i^l \in V_i^l$ such that

$$v = v^0 + \sum_{l=1}^{L} \sum_{i=1}^{J_l} v_i^l.$$

Moreover, there exists a constant $C > 0$, independent of the parameters h, H, and L, such that

$$\|v^0\|_a^2 + \sum_{l=1}^{L} \sum_{i=1}^{J_l} \|v_i^l\|_a^2 \le C\|v\|_a^2 \quad \forall v \in V^h.$$

We refer to [1], [14], [22], and [27] for the analysis and discussions of this bound.

To prove that the decomposition (11) is also uniformly bounded in the τ dependent norm $\|\cdot\|_{a_\tau}$, we need only to show that the same decomposition is uniformly bounded in the $L^2(\Omega)$ norm. Let us define Q^l as the usual L^2 projection

$$(Q^l v, \phi) = (v, \phi) \quad \forall v \in V^h \quad \forall \phi \in V^l$$

for $l = 0, \ldots, L$. For any given $v \in V^h$, let $v^0 = Q^0 v \in V^0$ and $v^l = Q^l v - Q^{l-1} v \in V^l$ for $l \ge 1$. It is easy to verify that

$$v = v^0 + v^1 + \cdots + v^l.$$

By using the fact that, for any $v \in V^h$ and $l \ge 1$,

$$\|(Q^l - Q^{l-1})v\|_{L^2(\Omega)}^2 = (v, (Q^l - Q^{l-1})v),$$

we establish the identity

$$\|Q^0 v\|_{L^2(\Omega)}^2 + \sum_{l=1}^{L} \|(Q^l - Q^{l-1})v\|_{L^2(\Omega)}^2 = \|v\|_{L^2(\Omega)}^2. \tag{13}$$

For each level $l \ge 1$, we define a partition of unity $\{\theta_i^l\}_{i=1}^{J_l}$ with $\theta_i^l \in H_0^1(\hat{\Omega}_i^l) \cap C_0^1(\hat{\Omega}_i^l \cap \Omega)$ such that $\sum_i \theta_i^l = 1$, $0 \le \theta_i^l \le 1$ and $|\nabla \theta_i^l| \le C/h_{l-1}$. Now, each $v^l = (Q^l - Q^{l-1})v$ can be further decomposed as

$$v^l = \sum_{i=1}^{J_l} v_i^l,$$

where $v_i^l = I_{h_l}(\theta_i^l v^l) \in V_i^l$, and where I_{h_l} is the piecewise linear interpolation operator from V^h to V^l. By using the second part of the proof of Lemma 4 of [4], we have that

$$\sum_{i=1}^{J_l} \|v_i^l\|_{L^2(\Omega)}^2 \le C\|v^l\|_{L^2(\Omega)}^2.$$

Therefore, by combining the above results, we obtain a decomposition

$$v = v^0 + \sum_{l=1}^{L} v^l = v^0 + \sum_{l=1}^{L} \sum_{i=1}^{J_l} v_i^l,$$

which is uniformly bounded in the $L^2(\Omega)$ norm, i.e.,

$$\|v^0\|^2_{L^2(\Omega)} + \sum_{l=1}^{L}\sum_{i=1}^{J_l} \|v_i^l\|^2_{L^2(\Omega)} \le C\|v\|^2_{L^2(\Omega)} \quad \forall v \in V^h,$$

where the constant $C > 0$ is independent of h, H, and L. As a consequence, we have proved the following lemma.

LEMMA 2.3. *For any $v \in V^h$, there exist $v_i^l \in V_i^l$, such that*

$$v = v^0 + \sum_{l=1}^{L}\sum_{i=1}^{J_l} v_i^l$$

and, moreover, there exists a constant C_0^{M}, independent of the parameters h, H, L, and τ, such that

$$\|v^0\|^2_{a_\tau} + \sum_{l=1}^{L}\sum_{i=1}^{J_l} \|v_i^l\|^2_{a_\tau} \le (C_0^{\mathrm{M}})^2\|v\|^2_{a_\tau} \quad \forall v \in V^h.$$

We now discuss the case when the coarse mesh space V^0 is dropped from the decomposition. In the analysis, we shall use the well-known approximation and boundedness properties of the operators $\{Q^l\}$, see e.g., [26], i.e., for any $v \in V^h$,

$$\|(Q^l - Q^{l-1})v\|^2_{L^2(\Omega)} \le Ch_l^2 a(v, v), \qquad l = 1, \ldots, L \tag{14}$$

and

$$\|Q^l v\|_a \le C\|v\|_a, \qquad l = 1, \ldots, L. \tag{15}$$

LEMMA 2.4. *For any $v \in V^h$, there exist $v_i^l \in V_i^l$, such that $v = \sum_{l=1}^{L}\sum_{i=1}^{J_l} v_i^l$ and*

$$\sum_{l=1}^{L}\sum_{i=1}^{J_l} \|v_i^l\|^2_{a_\tau} \le C\left(1 + \frac{\tau}{H^2}\right)\|v\|^2_{a_\tau} \equiv (C_\tau^{\mathrm{M}})^2\|v\|^2_{a_\tau} \quad \forall v \in V^h, \tag{16}$$

where $C > 0$ is independent of the parameters h, H, L, and τ.

Proof. For any given $v \in V^h$, we construct the multilevel overlapping decomposition as follows. Let $v^1 = Q^1 v$, $v^l = (Q^l - Q^{l-1})v$ for $l \ge 2$, and $v_i^l = I_{h_l}(\theta_i^l v^l)$ for $l = 1, \ldots, L$, $i = 1, \ldots, J_l$. These operators are linear, therefore

$$v = \sum_{l=1}^{L}\sum_{i=1}^{J_l} v_i^l.$$

We do the $H^1(\Omega)$ norm and $L^2(\Omega)$ norm estimates separately. For the $L^2(\Omega)$ estimate, by using the same arguments made for (13), we obtain

$$\sum_{l=1}^{L} \|v^l\|^2_{L^2(\Omega)} = \|v\|^2_{L^2(\Omega)}, \tag{17}$$

which, combined with the proof of Lemma 4 of [4], implies that

$$\sum_{l=1}^{L}\sum_{i=1}^{J_l} \|v_i^l\|^2_{L^2(\Omega)} \le C\|v\|^2_{L^2(\Omega)}. \tag{18}$$

We next examine the boundedness of the same decomposition described above in the $\|\cdot\|_a$ norm. We use the fact that

$$\sum_{l=1}^{L} \|v^l\|_a^2 \le C\|v\|_a^2 \quad \forall v \in V^h, \tag{19}$$

where the constant $C > 0$ is independent of the parameters h, H, and L. We refer to [1], [22], and [27] for the proofs and discussions.

Because of the estimate (6) of Dryja and Widlund [11], at each level $l \ge 2$, we have

$$|v_i^l|_{H_0^1(\hat{\Omega}_i^l)}^2 \le C\left(|v^l|_{H_0^1(\hat{\Omega}_i^l)}^2 + h_{l-1}^{-2}\|v^l\|_{L^2(\hat{\Omega}_i^l)}^2\right).$$

Here we used the fact that $\hat{\Omega}_i^l$ is the union of substructures of diameters on the order of h_{l-1}. Thus, by summing over $i = 1, \ldots, J_l$, and by using the approximation property (14), i.e., $\|v^l\|_{L^2(\Omega)}^2 \le Ch_{l-1}^2\|v^l\|_a^2$, we obtain

$$\sum_{i=1}^{J_l} \|v_i^l\|_a^2 \le C\left(\|v^l\|_a^2 + h_{l-1}^{-2}\|v^l\|_{L^2(\Omega)}^2\right) \le C\|v^l\|_a^2. \tag{20}$$

In the case $l = 1$, the desired estimate is known; see Lemma 8 of [4], i.e.,

$$\sum_{i=1}^{J_1} \|v_i^1\|_a^2 \le CH^{-2}\|v^1\|_a^2. \tag{21}$$

Therefore, the $\|\cdot\|_a$ estimate is accomplished by first adding (20), for $l = 2, \ldots, L$, and (21), and then using (19),

$$\sum_{l=1}^{L}\sum_{i=1}^{J_l} \|v_i^l\|_a^2 \le CH^{-2}\sum_{l=1}^{L} \|v^l\|_a^2 \le CH^{-2}\|v\|_a^2. \tag{22}$$

The proof of this lemma is completed by combining the results of the $L^2(\Omega)$ estimate (18) and the $H^1(\Omega)$ estimate (22). □

3. Schwarz algorithms and convergence rates analysis. In this section, we briefly describe the multiplicative Schwarz algorithms based on the various decompositions of V^h discussed in §2. The convergence rate of each algorithm is analyzed with the general Schwarz theory recently developed by Cai and Widlund [8].

3.1. Multiplicative Schwarz algorithms. Assuming that we have a set of triplets of $\{W_i, S_i, g_i \mid i = 1, \ldots, m\}$, where W_i are some subspaces of a normed linear space W, S_i some mappings from W to W_i, and $g_i \in W_i$, the multiplicative Schwarz algorithm can be described as follows.

ALGORITHM 3.1 (Multiplicative Schwarz $\{W_i, S_i, g_i\}$).
Given an initial guess $u^0 \in W$:
For $n = 0, 1, \ldots, n_{\max}$;
For $i = 1, \ldots, m$;
$u^{n+i/m} = u^{n+i-1/m} + (g_i - S_i u^{n+i-1/m})$.

It is not difficult to see that with the *error propagation* operator defined by

$$E = (I - S_m)\cdots(I - S_1)$$

and

$$g = (I - S_m)\cdots(I - S_2)g_1 + (I - S_M)\cdots(I - S_3)g_2 + \cdots + g_m,$$

then $u^{n+1} = Eu^n + g$. The convergence rate of the multiplicative Schwarz algorithm is thus determined by the spectral radus of the operator E. In an application, Algorithm 3.1 can also be used as a preconditioner for other conjugate gradient (CG)-type iterative methods, such as CG for symmetric positive definite problems, and generalized minimal residual (GMRES) [24], [15], or flexible GMRES (FGMRES) [23] for other cases. When it is used as a preconditioner, the actual system solved is the *transformed system* $(I - E)u = g$. We refer to [5], [6], [7], and [8] for discussion and comments on the numerical implementations.

In the rest of the paper, we take $W = V^h$ and W_i as one of the V_i or V_i^l defined in §2. Operators for subspaces V_i are defined as follows: For each $v \in V^h$, a unique $P_i v \in V_i$ is defined as the finite element solution of

$$d_\tau(P_i v, \phi) = d_\tau(v, \phi) \quad \forall \phi \in V_i.$$

Similarly, we can define P_i^l on V_i^l. We denote $g_i = P_i u_h^*$ and $g_i^l = P_i^l u_h^*$. We note that the g_i can be computed, without knowing the function u_h^* itself, by solving the finite element problems

$$d_\tau(g_i, \phi) = (f, \phi) \quad \forall \phi \in V_i. \tag{23}$$

Similarly, g_i^l can also be computed without knowing u_h^*.

ALGORITHM 3.2 (One-level multiplicative Schwarz). Run Algorithm 3.1 with an initial guess $u^0 \in V^h$ and $\{V_i, P_i, g_i \mid i = 1, \ldots, J\}$.

ALGORITHM 3.3 (Two-level multiplicative Schwarz). Run Algorithm 3.1 with an initial guess $u^0 \in V^h$ and $\{V_0, P_0, g_0, \text{ and } V_i, P_i, g_i \mid i = 1, \ldots, J\}$.

ALGORITHM 3.4 (Multilevel multiplicative Schwarz without a coarse mesh space). Run Algorithm 3.1 with an initial guess $u^0 \in V^h$ and

$$\{V_i^l, P_i^l, g_i^l \mid l = 1, \ldots, L,\ i = 1, \ldots, J_l\}.$$

ALGORITHM 3.5 (Multilevel multiplicative Schwarz with a coarse mesh space). Run Algorithm 3.1 with an initial guess $u^0 \in V^h$ and

$$\{V^0, P_0, g_0, \text{ and } V_i^l, P_i^l, g_i^l \mid l = 1, \ldots, L,\ i = 1, \ldots, J_l\}.$$

The convergence of these algorithms will be analyzed in §3.2. We note that the Algorithm 3.1 is a purely sequential algorithm; however, Algorithms 3.2–3.5 can be made highly parallel. This is due to the fact that each V_i (or V_i^l), except V_0 (or V^0), is the sum of several subspaces that are mutually orthogonal. Therefore the subproblem defined on V_i can be regarded as a set of independent sub-subproblems that can be solved in parallel. In Algorithms 3.2 and 3.4, the bottleneck step with V_0 (or V^0) is removed, as compared with their counterpart Algorithms 3.3 and 3.5. We show in the next subsection that the removal of the coarse mesh space does not degenerate by much the convergence rates for the class of parabolic problems under certain circumstances, although this is known to be not true for elliptic problems.

3.2. Convergence rates analysis. Let $E_{\text{A3.}i}$ be the error propagation operator for Algorithm $3.i$, $i = 1, \ldots, 5$. We estimate the norm of these operators by using a theorem of Cai and Widlund [8]. The proof of this theorem is technical; interested readers are referred to [8]

for details. To apply this theorem, we need only to verify certain properties of the subspaces related to the operator.

THEOREM 3.1 (Cai and Widlund). *Let W be a Hilbert space with inner product $v(\cdot,\cdot)$ and norm $\|\cdot\|_v = v(\cdot,\cdot)^{1/2}$. Suppose that W has a decomposition*

$$W = W_1 + \cdots + W_m, \tag{24}$$

where $W_1, \ldots, W_m$ are subspaces of W. Let the matrix $\mathcal{E} = \{\varepsilon_{ij}\}$ be defined by the strengthened Cauchy–Schwarz coefficients, where ε_{ij} is the smallest constant for which

$$|v(w_i, w_j)| \le \varepsilon_{ij}\|w_i\|_v\|w_j\|_v \quad \forall w_i \in W_i \quad \forall w_j \in W_j, \qquad i, j = 1, \ldots, m \tag{25}$$

holds. We assume that there are operators $T_i : W \to W_i$ that satisfy the following assumptions.

(i) *There exists a constant $\gamma > 0$ and parameters $\delta_i \ge 0$, such that $\sum_{i=1}^m \delta_i$ can be made sufficiently small, such that*

$$T_i + T_i^T - T_i^T T_i \ge \gamma T_i^T T_i - \delta_i I. \tag{26}$$

(ii) *There exists a constant $\bar{C}_0 > 0$, such that*

$$\sum_{i=1}^m T_i^T T_i \ge \bar{C}_0^{-2} I. \tag{27}$$

Here T_i^T is the adjoint operator of T_i given by $v(T_i^T u, v) = v(u, T_i v)$. Then, we have

$$\|(I - T_m)\cdots(I - T_1)\|_v \le \sqrt{1 - \bar{\gamma}\bar{C}_0^{-2}},$$

where

$$\bar{\gamma} = C\left(4(1+\gamma)^{-2}|\mathcal{E}|_{l^1}^2 + \left(\sum_{i=1}^m \delta_i\right)^2 + 1\right)^{-1}$$

and C is a positive constant.

In the remainder of this paper, we shall apply this theorem to the Hilbert space V^h equipped with the inner product $a_\tau(\cdot,\cdot)$, the decompositions (4), (5), (11), and (12) discussed in the previous section, and the operators P_i or P_i^l. Our main tasks include the verification of assumptions (i) and (ii) and the estimate of $|\mathcal{E}|_{l^1}$ for the four algorithms. The two assumptions required in Theorem 3.1 will be verified through the next two lemmas.

LEMMA 3.1. *For $i = 0, 1, \ldots, J$,*

$$P_i + P_i^T - P_i^T P_i \ge P_i^T P_i - \delta_i I \tag{28}$$

with $\delta_i = O(H)$, for $i \ge 1$, and $\delta_0 = CH^\alpha\sqrt{H^2/\tau + 1}$. Here $C > 0$ is independent of h, H, and τ. For $l = 1, \ldots, L$ and $i = 1, \ldots, J_l$,

$$P_i^l + (P_i^l)^T - (P_i^l)^T P_i^l \ge (P_i^l)^T P_i^l - \delta_i^l I \tag{29}$$

with $\delta_i^l = O(h_l)$.

The proof for P_0 can be obtained by applying Lemma 6 of [4] and the definition of P_0. A similar approach works for other mapping operators.

Remark 3.1. If the bilinear form $b(\cdot,\cdot)$ is selfadjoint and positive definite, then $\delta_i = 0$, for all $i = 0, \ldots, J$, and $\delta_i^l = 0$, for all $l = 1, \ldots, L$, $i = 1, \ldots, J_l$. $\gamma = 1.0$ in all cases.

Remark 3.2. For the one- and two-level methods, both summations

$$\sum_{i=1}^{J} \delta_i \le CJH \quad \text{and} \quad \delta_0 + \sum_{i=1}^{J} \delta_i \le C\left(H^{\alpha}\sqrt{\frac{H^2}{\tau}+1} + JH\right)$$

can made sufficiently small; therefore assumption (i) is verified. Note that the fact that J is independent of H is used here.

Remark 3.3. For the two multilevel algorithms, since h_l is proportional to Hr^l, the quantity

$$\sum_{l=1}^{L}\sum_{i=1}^{J_l} \delta_i^l \le C\frac{r}{1-r}\, JH$$

can be made sufficiently small if H is small. The extra member for the method with coarse mesh $H^{\alpha}\sqrt{H^2/\tau + 1}$ can also be made sufficiently small. Thus, assumption (i) is verified for the multilevel algorithms.

We next examine assumption (ii) for the four algorithms.

LEMMA 3.2. *The following four inequalities hold*:

$$\sum_{i=1}^{J} P_i^T P_i \ge C\left(1+\frac{\tau}{H^2}\right)^{-2} I = C_d^{-2}C_\tau^{-2} I, \tag{30}$$

$$P_0^T P_0 + \sum_{i=1}^{J} P_i^T P_i \ge CI = C_d^{-2}C_0^{-2} I, \tag{31}$$

$$\sum_{l=1}^{L}\sum_{i=1}^{J_l} (P_i^l)^T P_i^l \ge C\left(1+\frac{\tau}{H^2}\right)^{-2} I = C_d^{-2}(C_\tau^{\mathrm{M}})^{-2} I, \tag{32}$$

$$P_0^T P_0 + \sum_{l=1}^{L}\sum_{i=1}^{J_l} (P_i^l)^T P_i^l \ge CI = C_d^{-2}(C_0^{\mathrm{M}})^{-2} I, \tag{33}$$

where C_d *satisfies* $|d_\tau(u, v)| \le C_d \|u\|_{a_\tau} \|v\|_{a_\tau}$ *for any* $u, v \in V^h$.

Proof. The proofs for these four estimates are essentially the same, and we therefore only provide the proof of the second one. For any $v \in V^h$, by using the decomposition (4), the definition of mapping operators P_i, the Cauchy–Schwarz inequality, the uniformly boundedness of the decomposition (6) in the $\|\cdot\|_{a_\tau}$ norm, and the boundedness of $d_\tau(\cdot,\cdot)$, we have

$$\|v\|_{a_\tau}^2 \le d_\tau(v, v) = \sum_{i=0}^{J} d_\tau(v, v_i) = \sum_{i=0}^{J} d_\tau(P_i v, v_i)$$

$$\le C_d \sqrt{\sum_{i=0}^{J} \|P_i v\|_{a_\tau}^2}\sqrt{\sum_{i=0}^{J} \|v_i\|_{a_\tau}^2}$$

$$\le C_d C_0 \|v\|_{a_\tau} \sqrt{\sum_{i=0}^{J} \|P_i v\|_{a_\tau}^2}.$$

Therefore, removing the common term $\|v\|_{a_\tau}$ and squaring both sides, we obtain

$$C_d^{-2}C_0^{-2}\|v\|_{a_\tau}^2 \le \sum_{i=0}^{J} \|P_i v\|_{a_\tau}^2,$$

which is the desired proof. □

Remark 3.4. If $d_\tau(\cdot,\cdot)$ is symmetric positive definite, then $C_d = 1$.

We now discuss the bounds for $|\mathcal{E}|_{l^1}$. The cases for one- and two-level methods are simple. $|\mathcal{E}|_{l^1}$ is bounded by the maximum number of subregions that any given point in Ω belongs to. For example, for the box-decomposition of a rectangular region in the two-dimensional space, as in Fig. 1, the subregions can be colored such that $|\mathcal{E}|_{l^1} = 5$. In general, $|\mathcal{E}|_{l^1} \le J$ or $J+1$ for one- or two-level methods. In the multilevel cases, we need an estimate of Zhang [27],

$$a(v_i^{l_1}, v_j^{l_2}) \le C(r^d)^{|l_1-l_2|-1}\|v_i^{l_1}\|_a \|v_j^{l_2}\|_a \quad \text{for } l_1 \ne l_2 \quad \text{and} \quad \forall v_i^{l_1} \in V_i^{l_1}, \quad v_j^{l_2} \in V_j^{l_2}$$

and the following lemma.

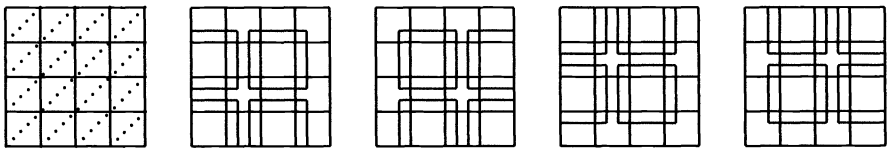


FIG. 1. *The coloring pattern of* 16 *fine grid overlapped subregions and a coarse grid region. Color* 0 *is for the global coarse grid. The extended subregions of the other colors are indicated by the dotted boundaries.*

LEMMA 3.3. *For* $l_1 \ne l_2$, *and any* $v_i^{l_1} \in V_i^{l_1}$, $v_j^{l_2} \in V_j^{l_2}$,

$$a_\tau(v_i^{l_1}, v_j^{l_2}) \le C(r^d)^{|l_1-l_2|-1}\|v_i^{l_1}\|_{a_\tau}\|v_j^{l_2}\|_{a_\tau},$$

where $C > 0$ *is independent of* h, H, L, *and* τ.

Proof. It suffices to show that, for $l_2 > l_1$,

$$(v_i^{l_1}, v_j^{l_2}) \le C(r^d)^{l_2-l_1-1}\|v_i^{l_1}\|_{L^2(\Omega)}\|v_j^{l_2}\|_{L^2(\Omega)}. \tag{34}$$

If the supporting regions $\hat{\Omega}_i^{l_1}$ and $\hat{\Omega}_i^{l_2}$ of $v_i^{l_1}$ and $v_j^{l_2}$ do not intersect, then (34) is trivial; otherwise, we have

$$(v_i^{l_1}, v_j^{l_2}) \le \|v_i^{l_1}\|_{L^2(\hat{\Omega}_i^{l_2})}\|v_j^{l_2}\|_{L^2(\Omega)}.$$

Let the triangle $K^{l_1} \in \mathcal{T}^{l_1}$ have vertices $y_1, \ldots, y_{d+1}$, and let $x_1, \ldots, x_k$ be all the nodal points of $\mathcal{T}^{l_2}$ in $\hat{\Omega}_i^{l_2}$. We can show that $k \le CJ_{l_2}(h_{l_2-1}/h_{l_2})^d$. Since $v_i^{l_1}$ is linear in $\hat{\Omega}_i^{l_2} \cap K^{l_1}$, we have

$$\|v_i^{l_1}\|_{L^2(\hat{\Omega}_i^{l_2}\cap K^{l_1})}^2 \le Ch_{l_2}^d \left(\sum_{1\le\sigma\le k, x_\sigma \in K^{l_1}} (v_i^{l_1}(x_\sigma))^2 \right)$$

$$\le Ck\, h_{l_2}^d \max_{x_\sigma \in K^{l_1}} \{|v_i^{l_1}(x_\sigma)|^2\}$$

$$\leq Ck\, h_{l_2}^d \sum_{\xi=1}^{d+1} (v_i^{l_1}(y_\xi))^2$$

$$\leq Ck \frac{h_{l_2}^d}{h_{l_1}^d} \|v_i^{l_1}\|^2_{L^2(K^{l_1})} \leq C \left(\frac{h_{l_2-1}}{h_{l_1}}\right)^d \|v_i^{l_1}\|^2_{L^2(K^{l_1})}.$$

By using the assumption that h_l is proportional to Hr^l, we obtain

$$\|v_i^{l_1}\|^2_{L^2(\hat{\Omega}_i^{l_2} \cap K^{l_1})} \leq C(r^d)^{l_2-l_1-1} \|v_i^{l_1}\|^2_{L^2(K^{l_1})}.$$

Summing over all $K^{l_1} \in \mathcal{T}^{l_1}$, we have

$$\|v_i^{l_1}\|^2_{L^2(\hat{\Omega}_i^{l_2})} \leq C(r^d)^{l_2-l_1-1} \|v_i^{l_1}\|^2_{L^2(\Omega)},$$

which proves the estimate (34). □

Applying Lemma 3.3 and using the fact that $1 + r^d + r^{2d} + \cdots \leq 1/(1-r^d)$, we see that $|\mathcal{E}|_{l^1}$ is uniformly bounded for both decompositions (11) and (12), independent of the number of levels L.

We now summerize the main convergence results for the four algorithms in the next theorem. The proof is a simple consequence of Theorem 3.1 and the lemmas of this section. Let $J = \max_l J_l$.

THEOREM 3.2. (a) *For Algorithm* 3.2, *there exists constants* $\tilde{c} > 0$ *and* $c = c(\tilde{c}) > 0$, *such that if* $\max\{H, H^\alpha \sqrt{H^2/\tau + 1}\} \leq \tilde{c}$, *then*

$$\|E_{\text{A3.2}}\|^2_{a_\tau} \leq 1 - \frac{c}{(1+J)^2 C_0^2}.$$

(b) *For Algorithm* 3.3, *there exist constants* $H_0 > 0$ *and* $c(H_0) > 0$, *such that if* $H \leq H_0$, *then*

$$\|E_{\text{A3.3}}\|^2_{a_\tau} \leq 1 - \frac{c}{(1+J)^2 C_\tau^2}.$$

Recall that $C_\tau^2 = C(1 + \tau/H^2)$.

(c) *For Algorithm* 3.4, *there exist constants* $H_0 > 0$ *and* $c(H_0) > 0$, *such that if* $H \leq H_0$, *then*

$$\|E_{\text{A3.4}}\|^2_{a_\tau} \leq 1 - \frac{c}{(1+J)^2 (C_\tau^{\text{M}})^2}.$$

Recall that $(C_\tau^{\text{M}})^2 = C(1 + \tau/H^2)$.

(d) *For Algorithm* 3.5, *there exists constants* $\tilde{c} > 0$ *and* $c = c(\tilde{c}) > 0$, *such that if* $\max\{H, H^\alpha \sqrt{H^2/\tau + 1}\} \leq \tilde{c}$, *then*

$$\|E_{\text{A3.5}}\|^2_{a_\tau} \leq 1 - \frac{c}{(1+\hat{c}^2)C} = 1 - \frac{c}{(1+\hat{c}^2)(C_0^{\text{M}})^2},$$

where $\hat{c} = J + H^\alpha \sqrt{1 + \frac{\tau}{H^2}} + JH$.

Here the constants c may be different in different occurrence.

Remark 3.5. As previously remarked, if the bilinear form $b(\cdot,\cdot)$ is symmetric positive definite, then the requirement that H be sufficiently small is not necessary, i.e., $H_0 = +\infty$.

Remark 3.6. In the nonsymmetric cases, when the coarse mesh size is $H \leq H_0$, the convergence is insured. Generally, the smaller H is, the faster the algorithm is.

Remark 3.7. These algorithms are often referred as algorithms with exact solvers, because all the subproblems defined in extended subregions, and the coarse mesh problem, are obtained by discretizing the original partial differential operator and then solved exactly by direct method. In practice, this is sometimes not necessary. A different differential operator can be used instead, and the linear systems need not be solved exactly; see discussions in [6], [7], and [8].

4. Applications to parabolic problems. In this section, we apply the algorithms developed in the previous sections to solve the following parabolic convection-diffusion problem: Find $u(x, t)$, such that

$$\begin{aligned} \tfrac{\partial u}{\partial t} + Lu &= f \qquad \text{in } \Omega \times [0, T], \\ u(x, t) &= 0 \qquad \text{on } \partial\Omega \times [0, T], \\ u(x, 0) &= u_0(x) \quad \text{in } \Omega, \end{aligned} \tag{35}$$

where $\Omega \subset R^2$. A weak formulation of (35) is: Find $u(x, t) \in H_0^1(\Omega)$, $u(x, 0) = u_0(x)$ in Ω, such that

$$\left(\frac{\partial u}{\partial t}, \phi\right) + b(u, \phi) = (f, \phi) \quad \forall \phi \in H_0^1(\Omega) \quad \forall t \in [0, T],$$

where the bilinear form $b(\cdot, \cdot)$ is the same as in (2). The existence and uniqueness of the solution of the weak parabolic convection-diffusion equation present no problems; see, e.g., [18]. Two time discretization schemes are considered, namely, a backward Euler scheme and a Crank–Nicolson scheme. The space variable is discretized by the piecewise linear finite element method. Let Δt_n be the nth timestep, M the number of steps, and $\sum_{n=1}^{M} \Delta t_n = T$. For the backward Euler–Galerkin scheme,

$$\left(\frac{u_h^n - u_h^{n-1}}{\Delta t_n}, \phi_h\right) + b(u_h^n, \phi_h) = (f, \phi_h) \quad \forall \phi_h \in V^h,$$

with $u_h^0(x, t) = u_0(x)$ and $n = 1, \ldots, M$. For the Crank–Nicolson–Galerkin scheme,

$$\left(\frac{u_h^n - u_h^{n-1}}{\Delta t_n}, \phi_h\right) + b\left(\frac{u_h^n + u_h^{n-1}}{2}, \phi_h\right) = (f, \phi_h) \quad \forall \phi_h \in V^h,$$

with $u_h^0(x, t) = u_0(x)$ and $n = 1, \ldots, M$. Both schemes lead to the following problem: For a given function $g_{n-1} \in H^{-1}(\Omega)$, find $w_h \in V^h$, such that

$$d_\tau(w_h, \phi_h) \equiv (w_h, \phi_h) + \tau b(w_h, \phi_h) = (g_{n-1}, \phi_h) \quad \forall \phi_h \in V^h, \tag{36}$$

where τ is the timestep parameter. The stability of both schemes is well understood; see, e.g., [18]. The algorithms discussed in the previous sections can thus be applied to the solution of (36) at each timestep. An obvious initial guess is the approximate solution obtained at the previous timestep. For the backward Euler–Galerkin scheme,

$$\begin{aligned} w_h &= u_h^n - u_h^{n-1}, \\ \tau &= \Delta t_n, \\ (g_{n-1}, \phi_h) &= \tau((f, \phi_h) - b(u_h^{n-1}, \phi_h)), \end{aligned}$$

and it is known that the truncation error is $O(\tau + h^2)$; therefore, it is reasonable to assume that τ is of order h^2 to maintain the balance of the time and space discretization errors. The factor τ/H^2 is thus $(h/H)^2$ bounded, and thus the algorithms without coarse mesh spaces are optimal. Similarly, for the Crank–Nicolson–Galerkin scheme,

$$\begin{aligned} w_h &= u_h^n - u_h^{n-1}, \\ \tau &= \tfrac{\Delta t_n}{2}, \\ (g_{n-1}, \phi_h) &= \tau(2(f, \phi_h) - b(u_h^{n-1}, \phi_h)), \end{aligned}$$

and since the truncation error is $O(\tau^2 + h^2)$, the factor τ/H^2 is approximately h/H^2. As long as this h/H^2 is kept reasonably small, the methods without coarse mesh spaces should also behave well.

In the rest of this section, we present some numerical results for model problems. We have only tested the one- and two-level methods; the numerical performance of the multilevel methods will be studied elsewhere. To specify our model problems, we give only the elliptic part of the parabolic operator. We consider the following linear second-order elliptic operator defined on $\Omega = [0, 1] \times [0, 1]$,

$$Lu = -\frac{\partial}{\partial x}\left(\xi \frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\eta \frac{\partial u}{\partial y}\right) + \alpha \frac{\partial u}{\partial x} + \beta \frac{\partial u}{\partial y} + \gamma u,$$

with the homogenous Dirichlet boundary condition. The right-hand side is chosen so that the exact solution is $u = e^{xy} \sin(\pi x) \sin(\pi y)$. The coefficients are specified as follows.

Example 1. $\xi = 1$, $\eta = 1$, and $\alpha = \beta = \gamma = 0$.

Example 2. $\xi = 1 + x^2 + y^2$, $\eta = e^{xy}$, $\alpha = 5(x + y)$, $\beta = 1/(1 + x + y)$, and $\gamma = 0$.

The domain Ω is partitioned with two uniform triangular grids that have sizes h and H, respectively. The actual values of h and H are given in Tables 1 and 2. The overlapping subdomains are colored by using four colors as in Fig. 1 and the coarse grid has color 0. The differential operator is discretized by the usual five-point center finite difference method at both the coarse and fine levels. We assume that the timestep has the form

$$\tau = h^{\epsilon},$$

where $\epsilon = 0.25, 0.5, 1.0, 1.25$, and 1.5. All algorithms are accelerated by the GMRES method without restarting. The subdomain problems, as well as the coarse mesh problem, are solved exactly by a direct method. We present the iteration counts for GMRES that are required to solve (36). The iteration is stopped when the $\|r_i\|_2/\|r_0\|_2 \leq 10^{-5}$, where r_i is the preconditioned residual at the ith iteration. The initial guess is always zero.

TABLE 1

Iteration counts for solving the problems given in Examples 1 *and* 2. *The two-level method is used with the fine mesh size uniformly* $h = \frac{1}{128}$ *and the overlap is* $\frac{1}{8}$ *of* H.

	Example 1					Example 2				
$H\backslash\epsilon$	0.25	0.5	1.0	1.25	1.5	0.25	0.5	1.0	1.25	1.5 10pt
$\frac{1}{4}$	5	5	4	4	3	5	5	5	4	3
$\frac{1}{8}$	4	4	4	4	4	4	4	4	4	4
$\frac{1}{16}$	3	3	4	4	4	4	4	4	4	4

For the two-level method, we test various coarse grid sizes and the iteration numbers are given in Table 1. As predicted in the theory, the numbers are independent of H and τ(or ϵ).

TABLE 2

Iteration counts for solving the problems given in Examples 1 *and* 2. *The one-level method is used with a uniform fine mesh* $h = \frac{1}{128}$ *and the overlap is* $\frac{1}{8}$ *of* H. *The approximate ratio* τ/H^2 *is given as the subscript of the iteration number for Example* 1. *The ratios are the same for Example* 2 *and are therefore omitted.*

	Example 1					Example 2				
$H\backslash\epsilon$	0.25	0.5	1.0	1.25	1.5	0.25	0.5	1.0	1.25	1.5
$\frac{1}{4}$	$9_{(4.76)}$	$9_{(1.41)}$	$6_{(0.13)}$	$4_{(0.04)}$	$3_{(0.01)}$	11	10	7	5	4
$\frac{1}{8}$	$17_{(19.03)}$	$16_{(5.66)}$	$10_{(0.50)}$	$6_{(0.15)}$	$4_{(0.04)}$	20	19	12	8	6
$\frac{1}{16}$	$33_{(76.11)}$	$30_{(22.63)}$	$17_{(2.00)}$	$10_{(0.59)}$	$7_{(0.18)}$	39	38	22	14	10

If the coarse solve is dropped as in the one-level method, the iteration numbers become more sensitive to the ratio τ/H^2, as seen in Table 2, especially for the nonsymmetric problem. We see that if the ratio τ/H^2 is relatively small, the results are acceptable as compared with the cases that use a coarse mesh solve.

Acknowledgment. I would like to thank Professor Olof Widlund for many valuable discussions and comments on this paper.

REFERENCES

[1] H. BRAMBLE AND J. E. PASCIAK, *New estimates for multilevel algorithms including the V-cycle*, Math. Comp., 60 (1993), pp. 447-471.

[2] J. H. BRAMBLE, J. E. PASCIAK, AND A. H. SCHATZ, *The construction of preconditioners for elliptic problems by substructuring*, II, Math. Comp., 49 (1987), pp. 1–16.

[3] J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for product iterative methods with applications to domain decomposition and multigrid*, Math. Comp., 57 (1991), pp. 1–22.

[4] X.-C. CAI, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–62.

[5] ———, *An optimal two-level overlapping domain decomposition method for elliptic problems in two and three dimensions*, SIAM J. Sci. Comput., 14 (1993), pp. 239–247.

[6] X.-C. CAI, W. D. GROPP, AND D. E. KEYES, *A comparison of some domain decomposition and ILU preconditioned iterative methods for nonsymmetric elliptic problems*, J. Numer. Linear Algebra Appl., (1993).

[7] X.-C. CAI AND O. B. WIDLUND, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 243–258.

[8] ———, *Multiplicative Schwarz algorithms for nonsymmetric and indefinite problems*, SIAM J. Numer. Anal., 30 (1993), pp. 936–952.

[9] C. DAWSON AND T. F. DUPONT, *Explicit/implicit, conservative, Galerkin domain decomposition procedures for parabolic problems*, Math. Comp., 58 (1992), pp. 21–34.

[10] M. DRYJA, *Substructuring methods for parabolic problems*, 4th Internat. Sympos. on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, Y. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[11] M. DRYJA AND O. B. WIDLUND, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Report 339, Dept. of Computer Science, Courant Institute of Mathematical Sciences, New York University, 1987.

[12] ———, *Some Domain Decomposition Algorithms for Elliptic Problems,* Iterative Methods for Large Linear Systems, L. Hayes and D. Kincaid, eds., Academic Press, San Diego, CA, 1989.

[13] ———, *Towards a unified theory of domain decomposition algorithms for elliptic problems*, in 3rd International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[14] ———, *Multilevel additive methods for elliptic finite element problems*, Parallel Algorithms for Partial Differential Equations, Proc. 6th GAMM-Seminar, Kiel, Jan. 19–21, W. Hackbusch, ed., Vieweg & Son, Braunschweig, Germany, 1991.

[15] S. C. EISENSTAT, H. C. ELMAN, AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric system of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[16] R. E. EWING, R. D. LAZAROV, J. E. PASCIAK, AND P. S. VASSILEVSKI, *Finite element methods for parabolic problems with timesteps variable in space*, Tech. Report 1989-05, Inst. for Scientific Computing, University of Wyoming, Laramie, 1989.

[17] P. GRISVARD, *Elliptic Problems in Nonsmooth Domains*, Pitman Publishing, Boston, MA, 1985.

[18] C. JOHNSON, *Numerical Solution of Partial Differential Equation by the Finite Element Method*, Cambridge University Press, Cambridge, 1987.

[19] Y. A. KUZNETSOV, *Domain decomposition methods for unsteady convection-diffusion problems*, in IXth Internat. Conf. on Comp. Methods in Appl. Science and Engineering, R. Glowinski and J. L. Lions, eds., Institut de Recherche d'Informatique, Paris, 1990, pp. 327–344.

[20] P. L. LIONS, *On the Schwarz Alternating Method.* I, Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[21] J. NEČAS, *Sur la coercivité des formes sesquilinéaires, elliptiques*, Rev. Roumaine Math., Pures Appl., 9 (1964), pp. 47–69.

[22] P. OSWALD, *On discrete morm estimates related to multilevel preconditioners in the finite element method*, in Proc. Internat. Conf. Theory of Functions, Varna 91, 1991.

[23] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, Tech. Report TR 91-279, Minnesota Supercomputer Inst., Univ. of Minnesota, Duluth, 1991.

[24] 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. 865–869.

[25] H. A. SCHWARZ, *Gesammelte Mathematische Abhandlungen*, Vol. 2, Springer-Verlag, Berlin, 1890, pp. 133–143. First published in Vierteljahrsschrift der Naturforschenden Gesellschaft, Zürich, 15 (1870), pp. 272–286.

[26] J. XU, *Theory of Multilevel Methods*, Ph.D. thesis, Cornell University, Ithaca, NY, May 1989.

[27] X. J. ZHANG, *Multilevel Schwarz methods*, Numer. Math., 63 (1992), pp. 521–539.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 604–620, May 1994

008

DOMAIN DECOMPOSITION ALGORITHMS WITH SMALL OVERLAP*

MAKSYMILIAN DRYJA† AND OLOF B. WIDLUND‡

Abstract. Numerical experiments have shown that two-level Schwarz methods often perform very well even if the overlap between neighboring subregions is quite small. This is true to an even greater extent for a related algorithm, due to Barry Smith, where a Schwarz algorithm is applied to the reduced linear system of equations that remains after the variables interior to the subregions have been eliminated. In this paper, a supporting theory is developed.

Key words. domain decomposition, elliptic finite element problems, preconditioned conjugate gradients, Schwarz methods

AMS subject classifications. 65F10, 65N30, 65N55

1. Introduction.

Over the last decade, considerable interest has developed in Schwarz methods and other domain decomposition methods for partial differential equations; see, e.g., the proceedings of five international symposia [17]–[19], [34], [35]. A general theory has evolved and a substantial number of new algorithms has been designed, analyzed and tested numerically. Among them are *two-level, additive Schwarz* methods first introduced in 1987; see Dryja and Widlund [25], [28]–[30], [55]. For related work see also Bjørstad, Moe, and Skogen [1]–[3], Cai [10]–[12], Mathew [40]–[42], Matsokin and Nepomnyaschikh [43], Nepomnyaschikh [44], Skogen [47], Smith [48]–[52], and Zhang [58], [59]. As shown in Dryja and Widlund [30], a number of other domain decomposition methods, in particular those of Bramble, Pasciak, and Schatz [5], [6], can also be derived and analyzed using the same framework. Recent efforts by Bramble et al. [7] and Xu [56] have extended the general framework making a systematic study of multiplicative Schwarz methods possible. The multiplicative algorithms are direct generalizations of the original alternating method discovered more than 120 years ago by H. A. Schwarz [46]. For other recent projects, that also use the Schwarz framework, see Dryja, Smith, and Widlund [27] and Dryja and Widlund [32], [33].

When a two-level method is used, the restrictions of the discrete elliptic problem to overlapping subregions, into which the given region has been decomposed, are solved exactly or approximately. These local solvers form an important part of a preconditioner for a conjugate gradient method. In addition, to enhance the convergence rate, the preconditioner includes a global problem of relatively modest dimension.

Generalizations to more than two levels have also been developed; see, e.g., Bramble, Pasciak, and Xu [8], Dryja and Widlund [31], and Zhang [58]–[60]; here the families of domain decomposition methods and multigrid algorithms merge. Recently there has also been a considerable interest in nonsymmetric and indefinite problems; cf., e.g., Bramble, Leyk, and Pasciak [4], Cai [10]–[12], Cai, Gropp, and Keyes [13], Cai and Widlund [14], [15], Cai and Xu [16], and Xu [57]. However, in this paper, we work exclusively with two-level methods for positive definite, symmetric problems.

*Received by the editors May 26, 1992; accepted for publication (in revised form) February 1, 1993.

†Department of Mathematics, Warsaw University, 2 Banach, 02-097 Warsaw, Poland (dryja@mimuw.edu.pl).
The work of this author was supported in part by National Science Foundation grant NSF-CCR-8903003, by Polish Scientific grant 211669101, and by the Center for Computational Sciences of the University of Kentucky at Lexington.

‡Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012 (widlund@cs.nyu.edu). This work was supported in part by National Science Foundation grant NSF-CCR-8903003 and in part by the U.S. Department of Energy contract DE-FG02-88ER25053 at the Courant Mathematics and Computing Laboratory.

The main result of our early study of two-level Schwarz methods shows that the condition number of the operator, which is relevant for the conjugate gradient iteration, is uniformly bounded if the overlap between neighboring subregions is sufficiently generous in proportion to the diameters of the subregions.

Our current work has been inspired very directly by several series of numerical experiments that indicate that the rate of convergence is quite satisfactory even for a small overlap, and that the running time of the programs is often the smallest when the overlap is at a minimum. The number of conjugate gradient iterations is typically higher in such a case, but this can be compensated for by the fact that the local problems are smaller and therefore cheaper to solve; see, in particular, Bjørstad, Moe, and Skogen [2], Bjørstad and Skogen [3], Cai [10], [11], Cai, Gropp, and Keyes [13], and Skogen [47]. If the local problems are themselves solved by an iterative method, then a smaller overlap will give better conditioned local problems and therefore a higher rate of convergence; see Skogen [47] for a detailed discussion of this effect. All this work also shows that these algorithms are relatively easy to implement. Recent experiments by Gropp and Smith [37] for problems of linear elasticity provide strong evidence that these methods can be quite effective even for very ill-conditioned problems. In this paper, we show that the condition number of the preconditioned operator for the algorithm, introduced in 1987 by Dryja and Widlund [28], is bounded from above by const.$(1 + (H/\delta))$. Here H measures the diameter of a subregion and δ the overlap between neighboring subregions. We note that H/δ is a measure of the aspect ratio of the subregion common to two overlapping neighboring subregions.

We then turn our attention to a very interesting method, introduced in 1989 by Barry Smith [48], [51]. It is known as the *vertex space* (or *Copper Mountain*) algorithm. Numerical experiments, for problems in the plane, have shown that this method converges quite rapidly even for problems, which were originally very ill conditioned, even if the overlap is very modest; see Smith [48]. For additional work on variants of this method, see Chan and Mathew [20], [21], and Chan, Mathew, and Shao [22].

When Smith's algorithm is used, the given large linear system of algebraic equations, resulting from a finite element discretization of an elliptic problem, is first reduced in size by eliminating all variables associated with the interiors of the nonoverlapping *substructures* $\{\Omega_i\}$ into which the region has been subdivided. The reduced problem is known as the *Schur complement* system and the remaining degrees of freedom are associated with the set $\{\partial\Omega_i\}$ of substructure boundaries that form the interface Γ between the substructures. The preconditioner of this domain decomposition method, classified as a *Schwarz method on the interface* in Dryja and Widlund [30], is constructed from a coarse mesh problem, with the substructures serving as elements, and a potentially large number of local problems. The latter correspond to an overlapping covering of Γ, with each subset corresponding to a set of adjacent interface variables.

Smith's main theoretical result, given in [48], [51], is quite similar to that for the original two-level Schwarz method; the condition number of this domain decomposition algorithm is uniformly bounded for a class of second-order elliptic problems provided that there is a relatively generous overlap between neighboring subregions that define the subdivision of the domain decomposition method. In this paper, we show that the condition number of the iteration operator grows only in proportion to $(1 + \log(H/\delta))^2$. We note that even for a minimal overlap of just one mesh width h, this bound is as strong as those for the well-known iterative substructuring methods considered by Bramble, Pasciak, and Schatz [5], [6], Dryja [24], Dryja, Proskurowski, and Widlund [26], Smith [49], and Widlund [54]; cf. also Dryja, Smith, and Widlund [27]. We also note that the successful iterative substructuring methods for problems in three dimensions require the use of more complicated coarse subspaces and

that therefore Smith's method, considered in this paper, seems to offer an advantage.

2. Some Schwarz methods for finite element problems. As usual, we write our continuous and finite element elliptic problems as: Find $u \in V$, such that

$$a(u, v) = f(v) \quad \forall\, v \in V,$$

and, find $u_h \in V^h$, such that

$$a(u_h, v_h) = f(v_h) \quad \forall\, v_h \in V^h, \tag{1}$$

respectively. We assume that the bilinear form $a(u, v)$ is selfadjoint and elliptic and that it is bounded in $V \times V$. In the case of Poisson's equation, the bilinear form is defined by

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v \, dx. \tag{2}$$

We assume that Ω is a Lipschitz region in R^n, $n = 2, 3$, and that its diameter is on the order of 1. (We will follow Nečas [45] when defining Lipschitz regions and Sobolev spaces on Ω.) The bilinear form $a(u, v)$ is directly related to the Sobolev space $H^1(\Omega)$ that is defined by the seminorm and norm

$$|u|^2_{H^1(\Omega)} = a(u, u) \quad \text{and} \quad \|u\|^2_{H^1(\Omega)} = |u|^2_{H^1(\Omega)} + \|u\|^2_{L^2(\Omega)},$$

respectively. When we are considering subregions $\Omega_i \subset \Omega$, of diameter H, we use a different relative weight, obtained by dilation,

$$\|u\|^2_{H^1(\Omega_i)} = |u|^2_{H^1(\Omega_i)} + \frac{1}{H^2}\|u\|^2_{L^2(\Omega_i)}.$$

Whenever appropriate, we tacitly assume that the elements of $H^1_0(\Omega_i)$, the subspace of $H^1(\Omega_i)$ with zero trace on the boundary $\partial\Omega_i$, are extended by zero to $\Omega \setminus \Omega_i$.

To avoid unnecessary complications, we confine our discussion to Poisson's equation, to homogeneous Dirichlet conditions, and to continuous, piecewise linear finite elements and a polygonal region Ω triangulated using triangles or tetrahedra. It is well known that the resulting space $V^h \subset H^1_0(\Omega)$, i.e., it is conforming.

For the problem considered in an 1870 paper by H.A. Schwarz [46], two overlapping subregions Ω'_1 and Ω'_2 are used; the union of the two is Ω. There are two sequential, fractional steps of the iteration in which the approximate solution of the elliptic equation on Ω is updated by solving the given problem restricted to the subregions one at a time. The most recent values of the solution are used as boundary values on the part of $\partial\Omega'_i$ that is not a part of $\partial\Omega$.

The finite element version of the algorithm can conveniently be described in terms of projections $P_i : V^h \to V^h_i = H^1_0(\Omega'_i) \bigcap V^h$, defined by

$$a(P_i v_h, \phi_h) = a(v_h, \phi_h) \quad \forall \phi_h \in V^h_i. \tag{3}$$

It is easy to show that the error propagation operator of this multiplicative Schwarz method is

$$(I - P_2)(I - P_1).$$

This algorithm can therefore be viewed as a simple iterative method for solving

$$(P_1 + P_2 - P_2 P_1)u_h = g_h,$$

with an appropriate right-hand side g_h.

This operator is a polynomial of degree two and therefore not ideal for parallel computing since two sequential steps are involved. This effect is further pronounced if more than two subspaces are used. Therefore, it is often advantageous to collect subregions, which do not intersect, into groups; the subspaces of each group can then be regarded as one. The number of subspaces is thus reduced, and the algorithm becomes easier to parallelize. Numerical experiments with multiplicative Schwarz methods have also shown that the convergence rate often is enhanced if such a strategy is pursued; this approach is similar to a red-black or multicolor ordering in the context of classical iterative methods. In the case of an additive Schwarz method, this ordering only serves as a device to facilitate the analysis.

In the additive form of the algorithm, we work with the simplest possible polynomial of the projections: The equation

$$Pu_h = (P_1 + P_2 + \cdots + P_N)u_h = g_h' \tag{4}$$

is solved by an iterative method. Here $P_i : V \to V_i$, and $V = V_1 + \cdots + V_N$. Since the operator P can be shown to be positive definite, symmetric with respect to $a(\cdot, \cdot)$, the iterative method of choice is the conjugate gradient method. Equation (4) must also have the same solution as (1), i.e., the correct right-hand side must be found. This can easily be arranged; see, e.g., [29], [30], [55]. Much of the work, in particular that which involves the individual projections, can be carried out in parallel.

2.1. The Dryja–Widlund algorithm. We now describe the special additive Schwarz method introduced in Dryja and Widlund [28]; see also Dryja [25] and Dryja and Widlund [29]. We start by introducing two triangulations of Ω into nonoverlapping triangular or tetrahedral substructures Ω_i and into triangular or tetrahedral elements. We obtain the elements by subdividing the substructures. We always assume that the two triangulations are shape regular, see, e.g., Ciarlet [23], and, to simplify our arguments, that the diameters of all the substructures are on the order of H. In this algorithm, we use overlapping subregions obtained by extending each substructure Ω_i to a larger region Ω_i'. The overlap is said to be generous if the distance between the boundaries $\partial\Omega_i$ and $\partial\Omega_i'$ is bounded from below by a fixed fraction of H. We always assume that $\partial\Omega_i'$ does not cut through any element. We carry out the same construction for the substructures that meet the boundary except that we cut off the part of Ω_i' that is outside of Ω.

We remark that other decompositions are also of interest. In particular, the analysis in §4 extends immediately to the case when no degrees of freedom are shared between neighboring subregions. In this case the distance between $\partial\Omega_i'$ and $\partial\Omega_j'$ is just h for neighboring subregions. This additive Schwarz method corresponds to a block Jacobi preconditioner augmented by a coarse solver.

For this Schwarz method, the finite element space is represented as the sum of N+1 subspaces

$$V^h = V_0^h + V_1^h + \cdots + V_N^h.$$

The first subspace V_0^h is equal to V^H, the space of continuous, piecewise linear functions on the coarse mesh defined by the substructures Ω_i. The other subspaces are related to the subdomains in the same way as in the original Schwarz algorithm, i.e., $V_i^h = H_0^1(\Omega_i') \bigcap V^h$.

It is often more economical to use approximate rather than exact solvers for the problems on the subspaces. The approximate solvers are described in the following terms: Let $b_i(u, v)$ be an inner product defined on $V_i^h \times V_i^h$ and assume that there exists a constant ω such that

$$a(u, u) \leq \omega b_i(u, u) \quad \forall u \in V_i^h. \tag{5}$$

In terms of matrices, this inequality becomes a one-sided bound of the stiffness matrix corresponding to $a(\cdot,\cdot)$ and V_i^h, in terms of the matrix corresponding to the bilinear form $b_i(u,u)$.

An operator $T_i : V^h \to V_i^h$, which replaces P_i, is now defined by

$$b_i(T_i u, \phi_h) = a(u, \phi_h) \quad \forall \phi_h \in V_i^h. \tag{6}$$

It is easy to show that the operator T_i is positive semidefinite and symmetric with respect to $a(\cdot,\cdot)$ and that the minimal constant ω in (5) is $\|T_i\|_a$. Additive and multiplicative Schwarz methods can now be defined straightforwardly in terms of polynomials of the operators T_i. We note that if exact solvers, and thus the projections P_i, are used, then $\omega = 1$.

2.2. The Smith algorithm. Smith's method has previously been described in Smith [48], [51]. Let K be the stiffness matrix given by the bilinear form of (2). In the first step of this and many other domain decomposition methods, the unknowns of the linear system of equations

$$Kx = b$$

that correspond to the interiors of the substructures are eliminated. We now describe this procedure in some detail.

Let $K^{(i)}$ be the stiffness matrix corresponding to the bilinear form $a_{\Omega_i}(u_h, v_h)$, which represents the contribution of the substructure Ω_i to the integral $a_\Omega(u_h, v_h) = a(u_h, v_h)$. Let x and y be the vectors of nodal values that correspond to the finite element functions u_h and v_h, respectively. Then the stiffness matrix K of the entire problem can be obtained by using the *method of subassembly* defined by the formula

$$x^T K y = \sum_i x^{(i)T} K^{(i)} y^{(i)}.$$

Here $x^{(i)}$ is the subvector of nodal parameters associated with $\overline{\Omega}_i$, the closure of Ω_i. We represent $K^{(i)}$ as

$$\begin{pmatrix} K_{II}^{(i)} & K_{IB}^{(i)} \\ K_{IB}^{(i)T} & K_{BB}^{(i)} \end{pmatrix}.$$

Here we have divided the subvector $x^{(i)}$ into two, $x_I^{(i)}$ and $x_B^{(i)}$, corresponding to the variables that are interior to the substructure and those that are shared with other substructures, i.e., those associated with the nodal points of $\partial\Omega_i$. Since the interior variables of Ω_i are coupled only to other variables of the same substructure, they can be eliminated locally and in parallel. The resulting reduced matrix is a Schur complement and is of the form

$$S^{(i)} = K_{BB}^{(i)} - K_{IB}^{(i)T} K_{II}^{(i)-1} K_{IB}^{(i)}. \tag{7}$$

From this follows that the Schur complement, corresponding to the global stiffness matrix K, is given by S where

$$x_B^T S y_B = \sum_i x_B^{(i)T} S^{(i)} y_B^{(i)}. \tag{8}$$

If the local problems are solved exactly, what remains is to find a sufficiently accurate approximation of the solution of the linear system

$$S x_B = g_B. \tag{9}$$

It is convenient to rewrite (9) in variational form. Let $s_i(u_h, v_h)$ and $s(u_h, v_h)$ denote the forms defined by (7) and (8), respectively, i.e.,

$$s_i(u_h, v_h) = x_B^{(i)} S^{(i)} y_B^{(i)} \quad \text{and} \quad s(u_h, v_h) = x_B^T S y_B.$$

Equation (9) can then be rewritten as

$$s(u_h, v_h) = (g, v_h)_{L^2(\Gamma)} \quad \forall v_h \in V^h(\Gamma). \tag{10}$$

Here $\Gamma = \bigcup \partial\Omega_i \setminus \partial\Omega$.

Problem (10) will be solved by an iterative method of additive Schwarz type. The most important difference between this algorithm and that of the previous subsection is that we are now working with the trace space $H^{1/2}(\Gamma)$ instead of $H^1(\Omega)$; see §3 for a definition of $H^{1/2}(\Gamma)$.

It is well known that

$$x_B^{(i)T} S^{(i)} x_B^{(i)} = \min_{x_I^{(i)}} x^{(i)T} K^{(i)} x^{(i)}. \tag{11}$$

Therefore, if u_h is the minimal, *the discrete harmonic* extension of the boundary data represented by x_B, then

$$x_B^{(i)T} S^{(i)} x_B^{(i)} = |u_h|^2_{H^1(\Omega_i)}.$$

Smith's algorithm can now be described in terms of a subspace decomposition. We use the same coarse space as in the previous subsection, i.e., V^H, but we restrict its values to Γ. In the case when the original problem is two dimensional, we introduce one subspace for each interior edge and one for each vertex of the substructures. An edge space is defined by setting all nodal values, except those associated with the interior of the edge in question, to zero. Similarly, a vertex space is obtained by setting to zero all values at the nodes on Γ, which are at a distance greater than δ. For many more details and a discussion of implementation details, see Smith [48], [51].

In the case when the original problem is three dimensional, we introduce one subspace for each interior face, edge, and vertex. The elements of a face subspace vanish at all nodes on Γ that do not belong to the interior of the face. Similarly, an edge space is supported in the strips of width δ, which belong to the faces, that have this edge in common. Finally, a vertex space is defined in terms of the nodes on Γ that are within a distance δ of the vertex. We also again use the standard coarse space V^H.

2.3. Basic theory. To estimate the rate of convergence of our special, or any other, additive Schwarz methods, we need upper and lower bounds for the spectrum of the operator relevant in the conjugate gradient iteration. A lower bound can be obtained by using the following lemma; see, e.g., Dryja and Widlund [29], [33] or Zhang [60].

LEMMA 2.1. *Let T_i be the operators defined in* (6) *and let $T = T_0 + T_1 + \cdots + T_N$. Then*

$$a(T^{-1}u, u) = \min_{u=\sum u_i} \sum b_i(u_i, u_i), \qquad u_i \in V_i.$$

Therefore, if a representation, $u = \sum u_i$, can be found, such that

$$\sum b_i(u_i, u_i) \le C_0^2 a(u, u) \quad \forall u \in V^h,$$

then

$$\lambda_{\min}(T) \ge C_0^{-2}.$$

An upper bound for the spectrum of T is often obtained in terms of strengthened Cauchy–Schwarz inequalities between the different subspaces. Note that we now exclude the index 0; the coarse subspace is treated separately.

DEFINITION 1. *The matrix $\mathcal{E} = \{\varepsilon_{ij}\}$ is the matrix of strengthened Cauchy–Schwarz constants, i.e., ε_{ij} is the smallest constant for which*

$$|a(v_i, v_j)| \leq \varepsilon_{ij} \|v_i\|_a \|v_j\|_a \quad \forall v_i \in V_i \quad \forall v_j \in V_j \qquad i, j \geq 1 \tag{12}$$

holds.

The following lemma is easy to prove; see Dryja and Widlund [33].

LEMMA 2.2. *Let $\rho(\mathcal{E})$ be the spectral radius of the matrix $\mathcal{E}$. Then, the operator T satisfies*

$$T \leq \omega(\rho(\mathcal{E}) + 1)I.$$

For the particular algorithms considered in this paper, it is very easy to show that there is a uniform upper bound. In fact, by collecting and merging local subspaces that belong to nonoverlapping subregions, the number of subspaces, and $\rho(\mathcal{E})$, can be made uniformly bounded; see §4 for an alternative argument.

By combining Lemmas 2.1 and 2.2, we obtain the following theorem.

THEOREM 2.3. *The condition number $\kappa(T)$ of the operator T of the additive Schwarz method satisfies*

$$\kappa(T) \equiv \lambda_{\max}(T)/\lambda_{\min}(T) \leq \omega(\rho(\mathcal{E}) + 1)C_0^2.$$

In the multiplicative case, we need to provide an upper bound for the spectral radius, or norm, of the error propagation operator

$$E_J = (I - T_J) \ldots (I - T_0). \tag{13}$$

The following theorem is a variant of a result of Bramble, Pasciak, Wang, and Xu [7]; see also Cai and Widlund [15], Xu [56], or Zhang [60]. Note that this bound is also given in terms of the same three parameters that appear in Theorem 2.3.

THEOREM 2.4. *In the symmetric, positive definite case*

$$\|E_J\|_a \leq \sqrt{1 - \frac{(2 - \omega)}{(2\omega^2\rho(\mathcal{E})^2 + 1)C_0^2}}\,.$$

We note that this formula is useless if $\omega \geq 2$; since $\|I - T_i\|_a > 1$ if $\|T_i\|_a > 2$, the assumption that $\omega < 2$ is most natural. If we wish to use a multiplicative algorithm and ω is too large, we can scale the bilinear forms $b_i(\cdot, \cdot)$ suitably.

In this paper, our results are only formulated for additive algorithms. The corresponding bounds for the multiplicative variants can easily be worked out as applications of Theorem 2.4.

3. Technical tools. In this section, we collect a number of technical tools that are used to prove our main results. Some of these tools are quite familiar to specialists of the field. Others, to our knowledge, have not previously been used in the analysis of domain decomposition; see Il'in [38] for some similar inequalities.

As before, $\Omega \subset R^n$, n =2 or 3, is a bounded, polygonal region and $\{\Omega_i\}$ a nonoverlapping decomposition of Ω into substructures. To simplify our considerations, we now assume that the substructures are squares or cubes; see, e.g., Nečas [45] where simple maps and partions of unity are used to derive bounds for Lipschitz regions from bounds for such special regions;

if we can handle a corner of a square or cube, then we can analyze the general polygonal case. Our estimates, given in the next two sections, are developed for one substructure at a time; our arguments can be modified to make them valid for any shape regular substructure with a boundary consisting of a finite number of smooth curves or surfaces.

As before, let $\Gamma = \bigcup \partial\Omega_i \setminus \partial\Omega$. Let $\Gamma_{\delta,i} \subset \Omega_i$ be the set of points that is within a distance δ of Γ.

LEMMA 3.1. *Let u be an arbitrary element of $H^1(\Omega_i)$. Then*

$$\|u\|^2_{L^2(\Gamma_{\delta,i})} \leq C\,\delta^2\big((1 + H/\delta)|u|^2_{H^1(\Omega_i)} + 1/(H\delta)\|u\|^2_{L^2(\Omega_i)}\big).$$

Proof. We first consider a square region $(0, H) \times (0, H)$ in detail; the extension of the proof to the case of three dimensions is straightforward. Since

$$u(x, 0) = u(x, y) - \int_0^y \frac{\partial u(x, \tau)}{\partial y} d\tau,$$

we find, by elementary arguments, that

$$H\int_0^H |u(x, 0)|^2 dx \leq 2\int_0^H\int_0^H |u(x, y)|^2 dxdy + H^2\int_0^H\int_0^H \left|\frac{\partial u}{\partial y}\right|^2 dxdy.$$

Therefore,

$$H\int_0^H |u(x, 0)|^2 dx \leq 2\|u\|^2_{L^2(\Omega_i)} + H^2|u|^2_{H^1(\Omega_i)}.$$

Now consider the integral over a narrow subregion next to one of the sides of the square. Using similar arguments, we obtain

$$\int_0^H\int_0^\delta |u(x, y)|^2 dxdy \leq \delta^2|u|^2_{H^1(\Omega_i)} + 2\delta\int_0^H |u(x, 0)|^2 dx. \tag{14}$$

By combining this and the previous inequality, we obtain

$$\int_0^H\int_0^\delta |u(x, y)|^2 dxdy \leq \delta^2|u|^2_{H^1(\Omega_i)} + 2\delta\left(\frac{2}{H}\|u\|^2_{L^2(\Omega_i)} + H|u|^2_{H^1(\Omega_i)}\right)$$

as required.

The modifications necessary for the case of an arbitrary, shape regular substructure and the extension of the proof to the case of three dimensions are routine. □

LEMMA 3.2. *Let u_h be a continuous, piecewise quadratic function defined on the finite element triangulation and let $I_h u_h \in V^h$ be its piecewise linear interpolant on the same mesh. Then there exists a constant C, independent of h and H, such that*

$$|I_h u_h|_{H^1(\Omega_i)} \leq C|u_h|_{H^1(\Omega_i)}.$$

The same type of bounds hold for the L_2 and $H^{1/2}_{00}$ norms and it can also be extended, with different constants, to the case of piecewise cubic functions, etc.

Proof. It is elementary to show that,

$$|I_h u_h|^2_{H^1(\Omega_i)} \leq 2\big(|I_h u_h - u_h|^2_{H^1(\Omega_i)} + |u_h|^2_{H^1(\Omega_i)}\big).$$

Consider the contribution to the first term on the right-hand side from an individual element K. We obtain

$$|I_h u_h - u_h|^2_{H^1(K)} \leq Ch^2 |u_h|^2_{H^2(K)} \leq C|u_h|^2_{H^1(K)}$$

by using a standard error bound and an elementary inverse inequality for quadratic polynomials. The bound in L_2 follows from the linear independence of the standard finite element basis for the space of quadratic polynomials, see Ciarlet [23]. The bounds for the $H^{1/2}_{00}$ norm, which is further discussed later in this section, can be obtained by interpolation in Sobolev spaces; cf., e.g., Lions and Magenes [39, pp. 98–99]. We use the K-method and a norm, which is equivalent to the one given below,

$$\|u\|^2_{H^{1/2}_{00}} = \|u\|^2_{L_2} + \int_0^\infty t^{-2} K(t, u)^2 dt,$$

where

$$K(t, u) = \inf_{u=u_0+u_1} (\|u_0\|^2_{L^2} + t^2 \|u_1\|^2_{H^1_0})^{1/2}.$$

The crucial observation is that

$$K(t, I_h u_h) \leq C K(t, u_h),$$

To prove this estimate, consider an arbitrary decomposition $u_h = u_0 + u_1$ of a piecewise quadratic u_h. Let Q be the L_2-projection onto the space of piecewise quadratic finite elements. It is well known that Q is bounded in H^1 as well as L_2; see, e.g., Bramble and Xu [9]. Using the H^1 and L_2 bounds of I_h derived in this proof, we find that

$$K(t, I_h u_h)^2 \leq \|I_h Q u_0\|^2_{L^2} + t^2 \|I_h Q u_1\|^2_{H^1_0} \leq C(\|u_0\|^2_{L^2} + t^2 \|u_1\|^2_{H^1_0}).$$

The proof now follows easily. □

We now turn to the other auxiliary results needed in the analysis of Smith's algorithm. We begin by writing down a standard expression for the norm of $H^{1/2}$; see Chapters 1.3.2 and 1.5 of Grisvard [36] for a detailed discussion.

Let $I \subset R^1$ be an open interval of diameter H. Then

$$\|u\|^2_{H^{1/2}(I)} = \int_I \int_I \frac{|u(s) - u(t)|^2}{|s - t|^2} ds dt + \frac{1}{H} \|u\|^2_{L_2(I)}. \tag{15}$$

The relative weight of the two terms is obtained, by dilation, from the norm defined on a region of diameter 1.

It is well known that the extension by zero of the elements of $H^{1/2}(I)$ does not define a continuous map into $H^{1/2}(R^1)$; see Lemma 1.3.2.6 of Grisvard [36] or Lions and Magenes [39]. The largest subspace for which this extension operator is continuous is $H^{1/2}_{00}(I)$, which is defined in terms of the norm obtained by replacing the last term of (15) by

$$\int_I \frac{|u(s)|^2}{d(s)} ds. \tag{16}$$

Here $d(s)$ is the distance to the nearest end point of I.

In the case of a subset Ψ of the boundary of a three-dimensional region, the formula (15) is valid after replacing $|s - t|^2$ by $|s - t|^3$ and I by Ψ. However, for our purposes, it is more

convenient to use an alternative formula; see Dryja [24] and Nečas [45, Lemma 5.3, Chap. 2]. In the special case of a square with side H, the seminorm is defined by

$$(17)\quad \int_0^H \int_0^H \frac{\|u(s_1,.)-u(t_1,.)\|^2_{L_2}}{|s_1-t_1|^2} ds_1 dt_1 + \int_0^H \int_0^H \frac{\|u(.,s_2)-u(.,t_2)\|^2_{L_2}}{|s_2-t_2|^2} ds_2 dt_2.$$

To obtain the norm for the subspace $H_{00}^{1/2}((0,H)^2)$, we add a weighted norm

$$(18)\quad \int_0^H \int_0^H \frac{|u(s_1,s_2)|^2}{d(s)} ds_1 ds_2$$

just as in (16).

In addition to the space V^h, we will also use a coarser space V^δ, defined on a mesh with mesh size δ, in our proofs. We now formulate results that have been used extensively in work of this kind; see Dryja [24] or Bramble and Xu [9]. The first inequality of the lemma is given as Lemma 1 in Dryja [24]. The second is part of the proof of Lemma 4 in the same paper.

LEMMA 3.3. *Let I be an interval of length H. Then*

$$\|u_\delta\|^2_{L_\infty(I)} \le C(1+\log(H/\delta))\|u_\delta\|^2_{H^{1/2}(I)} \quad \forall u_\delta \in V^\delta.$$

Let I be an edge of a face Ψ of diameter H of a cube. Then

$$\|u_\delta\|^2_{L^2(I)} \le C(1+\log(H/\delta))\|u_\delta\|^2_{H^{1/2}(\Psi)} \quad \forall u_\delta \in V^\delta.$$

The next result gives a bound that is similar to the second formula of Lemma 3.3. However, the bound holds for all of $H^{1/2}$.

LEMMA 3.4. *Let $\Psi = (0,H)^2$ and let $\Psi_\delta = (0,H)\times(0,\delta)$. Then*

$$\|u\|^2_{L_2(\Psi_\delta)} \le C\delta(1+\log(H/\delta))\|u\|^2_{H^{1/2}(\Psi)} \quad \forall u \in H^{1/2}(\Psi).$$

The same result holds if we replace Ψ and Ψ_δ by $(0,H)$ and $(0,\delta)$, respectively.

Proof. We only provide a proof for the first of the two cases; the proof in the other case is completely analogous. Let $Q_\delta : L_2(\Psi) \to V^\delta$, be the L_2- projection. It is well known that $\|Q_\delta\|_{H^1(\Psi)}$ is bounded; see, e.g., Bramble and Xu [9]. Since, trivially, this operator is also bounded in L_2, it follows that $\|Q_\delta\|_{H^{1/2}(\Psi)}$ is bounded. By a standard argument,

$$(19)\quad \|u-Q_\delta u\|^2_{L_2(\Psi)} \le \delta C|u|^2_{H^{1/2}(\Psi)}.$$

We now only need to show that

$$(20)\quad \|Q_\delta u\|^2_{L_2(\Psi_\delta)} \le C\delta(1+\log(H/\delta))\|Q_\delta u\|^2_{H^{1/2}(\Psi)}.$$

To prove (20), we use the bound (14) derived in the proof of Lemma 3.1. Thus

$$\|Q_\delta u\|^2_{L_2(\Psi_\delta)} \le \delta^2|Q_\delta u|^2_{H^1(\Psi)} + 2\delta\int_0^H |Q_\delta u(x,0)|^2 dx.$$

By using an inverse inequality, the first term can be replaced by $\delta|Q_\delta u|^2_{H^{1/2}(\Psi)}$. The second term is estimated using Lemma 3.3. □

The final lemma will be used to estimate the weighted L_2 term in the $H_{00}^{1/2}$ norm.

LEMMA 3.5. *Let $u \in H^{1/2}(0,H)$. Then there exists a constant C, such that*

$$\int_\delta^H \frac{|u(s)|^2}{s} ds \le C(1+\log(H/\delta))^2\|u\|^2_{H^{1/2}(0,H)}.$$

Similarly, let $u \in H^{1/2}((0, H)^2)$. *Then there exists a constant* C, *such that*

$$\int_0^H \left(\int_\delta^H \frac{|u(s,t)|^2}{s} ds \right) dt \leq C(1 + \log(H/\delta))^2 \|u\|^2_{H^{1/2}((0,H)^2)}.$$

Proof. We begin by considering the first inequality. Let $Q_\delta : L_2(0, H) \to V^\delta(0, H)$, be the L_2-projection onto the finite element space with mesh size δ. We write $u = (u - Q_\delta u) + Q_\delta u$ and estimate each term separately. We first note that by a standard estimate,

$$\|u - Q_\delta u\|^2_{L_2(0,H)} \leq C\delta |u|^2_{H^{1/2}(0,H)}.$$

The bound for the first term is therefore obtained by noting that $s \geq \delta$ over the interval of integration.

The other term can be estimated by using the first bound of Lemma 3.3, which results in one logarithmic factor, and the observation that

$$\int_\delta^H \frac{|Q_\delta u|^2}{s} ds \leq \|Q_\delta u\|^2_{L_\infty} \int_\delta^H \frac{ds}{s},$$

from which the second logarithmic factor arises.

The second inequality follows by considering $u(s, t)$, a function of two variables, and applying the first inequality with respect to s. We then integrate with respect to t, change the order of integration, and use formula (17) to complete the proof. □

4. Analysis of the Dryja–Widlund algorithm. We now use the set $\Gamma_{\delta,i}$, previously introduced, to characterize the extent of the overlap. We assume that all $x \in \Omega_i$, which belong to at least one additional overlapping subregion Ω'_j, lie in $\Gamma_{\delta,i}$.

THEOREM 4.1. *In the case when exact solvers are used for the subproblems, the condition number of the additive Schwarz method satisfies*

$$\kappa(P) \leq C(1 + H/\delta).$$

The constant is independent of the parameters H, h, *and* δ.

We note that for the case of two subregions, it is easy to show that this result is sharp. It is routine to modify Theorem 4.1 to cover cases where inexact solvers are used.

Proof. The proof is a refinement of a result first given in Dryja and Widlund [28]; cf. [29] for a better discussion. The proof is equally valid for two and three dimensions. We first show that a constant upper bound for the spectrum of P can be obtained without the use of Lemma 2.2.

We note that P_i is also an orthogonal projection of $H^1(\Omega'_i) \bigcap V^h$ onto V_i. Therefore,

$$a(P_i u_h, u_h) \leq a_{\Omega'_i}(u_h, u_h).$$

Since, by construction, there is an upper bound, N_c, on the number of subregions to which any $x \in \Omega$ can belong, we have

$$\sum_{i=1}^N a_{\Omega'_i}(u_h, u_h) \leq N_c\, a(u_h, u_h).$$

In addition, we use the fact that the norm of P_0 is equal to one and obtain

$$\lambda_{\max}(P) \leq (N_c + 1).$$

The lower bound is obtained by using Lemma 2.1. A natural choice of u_0 is the L_2-projection $Q_H u_h$ of u_h onto V^H. As previously pointed out, this projection is bounded in L_2 as well as H^1 and there exists a constant, independent of h and H, such that

$$\|u_h - Q_H u_h\|_{L_2} \leq C\, H \|u_h\|_a. \tag{21}$$

Let $w_h = u_h - Q_H u_h$ and let $u_i = I_h(\theta_i w_h)$, $i = 1, \ldots, N$. Here I_h is the interpolation operator onto the space V^h and the $\theta_i(x)$ define a partition of unity, i.e., $\sum_i \theta_i(x) \equiv 1$. These functions are chosen as nonnegative elements of V^h. It is easy to see that

$$u_h = Q_H u_h + \sum u_i.$$

In the interior part of Ω_i, which does not belong to $\Gamma_{\delta,i}$, $\theta_i \equiv 1$. This function must decrease to 0 over a distance on the order of δ. It is easy to construct a partition of unity with $0 \leq \theta_i \leq 1$ and such that

$$|\nabla \theta_i| \leq \frac{C}{\delta}.$$

To use Lemma 2.1, we first estimate $a(u_i, u_i)$ in terms of $a(w_h, w_h)$. We consider the contribution from one substructure at a time and note that, trivially,

$$a_{\Omega_i \setminus \Gamma_{\delta,i}}(u_i, u_i) = a_{\Omega_i \setminus \Gamma_{\delta,i}}(w_h, w_h).$$

Let K be an element in $\Gamma_{\delta,i}$. Then, using the definition of u_i,

$$a_K(u_i, u_i) \leq 2a_K(\bar{\theta}_i w_h, \bar{\theta}_i w_h) + 2a_K(I_h((\theta_i - \bar{\theta}_i)w_h), I_h((\theta_i - \bar{\theta}_i)w_h)),$$

where $\bar{\theta}_i$ is the average of θ_i over the element K. Using the fact that the diameter of K is on the order of h and the bound on $\nabla\theta_i$, we obtain, after adding over all the relevant elements,

$$a_{\Gamma_{\delta,i}}(u_i, u_i) \leq 2a_{\Omega_i}(w_h, w_h) + \frac{C}{\delta^2}\|w_h\|^2_{L^2(\Gamma_{\delta,i})}.$$

We also need to estimate $a_{\Gamma_{\delta,i}}(u_j, u_j)$ for the j that correspond to neighboring substructures. This presents no new difficulties.

To complete the proof, we need to estimate $\|w_h\|^2_{L^2(\Gamma_{\delta,i})}$. We note that each $x \in \Omega$ is covered only a finite number of times by the subregions. We apply Lemma 3.1 to the function w_h, sum over i and use inequality (21) to complete the estimate of the parameter C_0^2 of Lemma 2.1. □

5. Analysis of the Smith method. A description of the reduction of the original linear system to one for the degrees of freedom on Γ, and of the algorithm, have been given in §2. We will now work in the $H^{1/2}(\Gamma)$ norm. The fact that this is a weaker norm than H^1 is reflected in a stronger bound than that of the previous section; better bounds of the components in the different subspaces can be obtained. There are many similarities between the two cases. Much of the analysis is again carried out one substructure at a time.

To show that we can work with $|u_h|^2_{H^{1/2}(\partial\Omega_i)}$ instead of $x_B^{(i)T} S^{(i)} x_B^{(i)}$, we must show that these norms are equivalent. We use (11) and the standard trace theorem to bound $|u_h|^2_{H^{1/2}(\partial\Omega_i)}$ from above by $x_B^{(i)T} S^{(i)} x_B^{(i)}$. The proof of the reverse inequality requires an extension theorem for finite element spaces given in Widlund [53]; see further discussion in Smith [51].

THEOREM 5.1. *In the case when exact solvers are used for the subproblems, the condition number of the vertex space method satisfies*

$$\kappa(P) \leq C(1 + \log(H/\delta))^2.$$

The constant is independent of the parameters H, h, and δ.

Proof. As in the proof of Theorem 4.1, there is no difficulty in establishing a uniform upper bound on the spectrum of P.

We now turn to the lower bound in the case where the original problem is two dimensional and thus the interface is of dimension one. To use Lemma 2.1, we have to decompose functions defined on Γ.

We use the $L_2(\Omega)$ projection onto V^H of the discrete harmonic function u_h, introduced in §2.2, to define the component of the coarse space. We only use the values on Γ. In addition, we use a partition of unity to represent the local space components. In the study of the local spaces, it is sufficient to consider one substructure Ω_i at a time. The partition of unity is based on simple, piecewise linear functions. Let $0 < t < H$ represent one of the edges of the boundary of this substructure and let $\theta_e(t)$ be a piecewise linear function that vanishes for t outside $(0, H)$, grows linearly to 1 at $t = \delta$, is equal to 1 for $\delta \leq t \leq H - \delta$, and drops to zero linearly over the interval $(H - \delta, H)$. In the decomposition, we choose $I_h(\theta_e w_h)$ as the component corresponding to this edge. As in the previous section, $u_0 = Q_H u_h$ and $w_h = u_h - Q_H u_h$ is the error of the L_2- projection. It follows from Lemma 3.2 that it is sufficient to estimate $\|v_e\|_{H_{00}^{1/2}(0,H)}$, where $v_e(t) = \theta_e(t) w_h(t)$. We note that we cannot use the weaker norm of $H^{1/2}(0, H)$ here; we must estimate the $H^{1/2}(\partial\Omega_i)$ norm of v_e extended by zero to the rest of the boundary, i.e., $\|v_e\|_{H_{00}^{1/2}(0,H)}$.

We first consider

$$\int_0^H \int_0^H \frac{|v_e(s) - v_e(t)|^2}{|s-t|^2} ds dt, \tag{22}$$

and then the additional term (16), which completes the definition of the relevant norm. We divide the interval $[0, H]$ into three parts, $[0, \delta]$, $[\delta, H - \delta]$, and $[H - \delta, H]$, and take the tensor product of $[0, H]$ with itself. The double integral (22) is then split into the sum of nine. By symmetry, only six different cases need to be considered. The integral over $[\delta, H - \delta] \times [\delta, H - \delta]$ is completely harmless. We now consider the diagonal term corresponding to the set $[0, \delta] \times [0, \delta]$ and use the identity

$$v_e(s) - v_e(t) \equiv \frac{s w_h(s) - t w_h(t)}{\delta} = \frac{(s+t)(w_h(s) - w_h(t))}{2\delta} + \frac{(s-t)(w_h(s) + w_h(t))}{2\delta}.$$

The integral corresponding to the first term is estimated by $\|w_h\|_{H^{1/2}}$ after noting that, for the relevant values of s and t, $|s + t|/2\delta \leq 1$.

The integral, corresponding to the second term, can be estimated by

$$1/\delta^2 \int_0^\delta \int_0^\delta |w_h(t)|^2 ds dt = 1/\delta \|w_h\|^2_{L^2(0,\delta)},$$

which, in turn, can be estimated appropriately by using Lemma 3.4.

The third diagonal double integral is estimated in exactly the same way.

We next estimate the off-diagonal double integrals. We note that for $0 \leq t \leq \delta$ and $\delta \leq s \leq H - \delta$,

$$v_e(s) - v_e(t) = w_h(s) - \frac{t}{\delta} w_h(t) = (w_h(s) - w_h(t)) + \frac{(\delta - t)}{\delta} w_h(t).$$

The first term gives an integral that can be estimated straightforwardly in terms of $|w_h|^2_{H^{1/2}(0,H)}$. What remains is the integral

$$\int_0^\delta \left(\int_\delta^H \frac{(\delta - t)^2}{\delta^2 (t - s)^2} ds \right) |w_h(t)|^2 dt.$$

We integrate with respect to s and find that the inner integral is bounded by $1/\delta$ and the double integral by

$$1/\delta \|w_h\|^2_{L^2(0,\delta)}. \tag{23}$$

The estimates of the other integrals can be carried out quite similarly. To complete the estimate of the $\|v_e\|_{H_{00}^{1/2}(0,H)}$, we consider

$$\int_0^H \left(\frac{|v_e(s)|^2}{s} + \frac{|v_e(s)|^2}{H-s} \right) ds.$$

For $s \in (0, \delta)$, and $s \in (H - \delta, H)$, we obtain contributions that can be estimated by the expression given in (23). For the integral over $(\delta, H - \delta)$, we use Lemma 3.5.

We next turn to the space associated with one of the vertices of Ω_i. We now use $\theta_v(t) = (\delta - |t|)/\delta$ to complete the partition of unity, i.e., we use $I_h(\theta_v w_h) = I_h(v_v)$. It follows from Lemma 3.2 that we can again ignore the interpolation operator I_h. We need to estimate

$$\int_{-\delta}^{\delta} \int_{-\delta}^{\delta} \frac{|v_v(s) - v_v(t)|^2}{|s-t|^2} ds dt$$

and

$$\int_{-\delta}^{\delta} \left(\frac{|v_v(s)|^2}{\delta - s} + \frac{|v_v(s)|^2}{s + \delta} \right) ds.$$

Considering the double integral, we note that

$$\frac{(\delta - |s|)w_h(s) - (\delta - |t|)w_h(t)}{\delta(s-t)} = \frac{w_h(s) - w_h(t)}{s-t} \left(1 - \frac{|s| + |t|}{2\delta} \right) - \frac{w_h(s) + w_h(t)}{2\delta} \frac{|s| - |t|}{s-t}.$$

Since $|1 - \frac{|s|+|t|}{2\delta}| \leq 1$ and $|\frac{|s|-|t|}{s-t}| \leq 1$, for relevant values of s and t, the two contributions to the integral can be estimated in terms of

$$|w_h|^2_{H^{1/2}(0,H)} \quad \text{and} \quad 1/\delta \|w_h\|^2_{L^2(0,\delta)},$$

respectively. Arguments, quite similar to those given above, complete the proof for the case of two dimensions.

We now turn to problems in three dimensions, i.e., the case where the interface Γ is two dimensional. In addition to the coarse space, we use three types of local subspaces associated with faces, and neighborhoods of edges and vertices, respectively. The diameter of the point set associated with a vertex subspace is on the order of δ. Similarly, the edge spaces include the degrees of freedom on Γ that are within a distance δ of the edge in question. We again construct a partition of unity associated with these sets. As before these functions are continuous, piecewise linear functions and their gradients are bounded by C/δ. The proof proceeds as in the case of two dimensions. We give only a few details. We use $\theta_e(t_1)\theta_e(t_2)$ to construct the contribution to the decomposition related to a face. Similarly, we use $\theta_e(t_1)\theta_v(t_2)$ as the part of the partition of unity associated with an edge. Using our formulas for θ_e and θ_v, we then show that the partition of unity is completed by adding functions that differ from zero only in small neighborhoods of the vertices. The estimates necessary for the use of Lemma 2.1 and the completion of this proof are then carried out as in the two-dimensional case. □

REFERENCES

[1] P. E. BJØRSTAD, *Multiplicative and additive Schwarz methods: Convergence in the two domain case*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[2] P. E. BJØRSTAD, R. MOE, AND M. SKOGEN, *Parallel domain decomposition and iterative refinement algorithms*, in Parallel Algorithms for PDEs, Proceedings of the 6th GAMM-Seminar held in Kiel, Germany, Jan. 19–21, 1990, W. Hackbusch, ed., Vieweg & Son, Braunschweig, Wiesbaden, 1990.

[3] P. E. BJØRSTAD 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. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[4] 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.

[5] J. H. BRAMBLE, J. E. PASCIAK, AND A. H. SCHATZ, *The construction of preconditioners for elliptic problems by substructuring*, I, Math. Comp., 47 (1986), pp. 103–134.

[6] ———, *The construction of preconditioners for elliptic problems by substructuring*, IV, Math. Comp., 53 (1989), pp. 1–24.

[7] J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for product iterative methods with applications to domain decomposition*, Math. Comp., 57 (1991), pp. 1–21.

[8] J. H. BRAMBLE, J. E. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[9] J. H. BRAMBLE AND J. XU, *Some estimates for a weighted L^2 projection*, Math. Comp., 56 (1991), pp. 463–476. Ph.D. thesis compare [58].

[10] X.-C. CAI, *Some Domain Decomposition Algorithms for Nonselfadjoint Elliptic and Parabolic Partial Differential Equations*, Ph.D. thesis, Mathematics Dept., Courant Institute of Mathematical Sciences, New York University, Sept. 1989; Tech. Report 461, Dept. of Computer Science, Courant Institute of Mathematical Sciences, New York University, Sept. 1989.

[11] ———, *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. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[12] ———, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.

[13] X.-C. CAI, W. D. GROPP, AND D. E. 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. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[14] X.-C. CAI AND O. WIDLUND, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 243–258.

[15] ———, *Multiplicative Schwarz algorithms for some nonsymmetric and indefinite problems*, SIAM J. Numer. Anal., 30 (1993), pp. 936–952.

[16] X.-C. CAI AND J. XU, *A preconditioned GMRES method for nonsymmetric or indefinite problems*, Math. Comp., 59 (1992), pp. 311–319.

[17] T. CHAN, R. GLOWINSKI, J. PÉRIAUX, AND O. WIDLUND, EDS., *Domain Decomposition Methods*, in Second International Symposium on Domain Decomposition Methods, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[18] ———, EDS., *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[19] T. F. CHAN, D. E. KEYES, G. A. MEURANT, J. S. SCROGGS, AND R. G. VOIGT, EDS., *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[20] T. F. CHAN AND T. P. MATHEW, *An application of the probing technique to the vertex space method in domain decomposition*, in Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, Y. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[21] ———, *The interface probing technique in domain decomposition*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 212–238.

[22] T. F. CHAN, T. P. MATHEW, AND J.-P. SHAO, *Efficient variants of the vertex space domain decomposition algorithm*, Tech. Report CAM 92-07, Department of Mathematics, Univ. of California Los Angeles, Jan. 1992. SIAM J. Sci. Comput., 15 (1994).

[23] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978.

[24] M. DRYJA, *A method of domain decomposition for* 3-D *finite element problems*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[25] ———, *An additive Schwarz algorithm for two- and three-dimensional finite element elliptic problems*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[26] M. DRYJA, W. PROSKUROWSKI, AND O. WIDLUND, *A method of domain decomposition with crosspoints for elliptic finite element problems*, in Optimal Algorithms, B. Sendov, ed., Bulgarian Academy of Sciences, Sofia, Bulgaria, 1986, pp. 97–111.

[27] M. DRYJA, B. F. SMITH, AND O. B. WIDLUND, *Schwarz analysis of iterative substructuring algorithms for problems in three dimensions*, Tech. Report 638, Dept. of Computer Science, Courant Institute of Mathematical Sciences, New York University, May 1993.

[28] M. DRYJA AND O. B. WIDLUND, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Report 339; also, Ultracomputer Note 131, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, 1987.

[29] ———, *Some domain decomposition algorithms for elliptic problems*, in Iterative Methods for Large Linear Systems, L. Hayes and D. Kincaid, eds., Academic Press, San Diego, CA, 1989, pp. 273–291.

[30] ———, *Towards a unified theory of domain decomposition algorithms for elliptic problems*, in Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[31] ———, *Multilevel additive methods for elliptic finite element problems*, in Parallel Algorithms for Partial Differential Equations, W. Hackbusch, ed., Proceedings of the Sixth GAMM-Seminar, Kiel, Jan. 19–21, 1990, Vieweg & Son, Braunschweig, Germany, 1991.

[32] ———, *Additive Schwarz methods for elliptic finite element problems in three dimensions*, in Fifth Conference on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[33] ———, *Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems*, Tech. Report 626, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, March 1993.

[34] R. GLOWINSKI, G. H. GOLUB, G. A. MEURANT, AND J. PÉRIAUX, EDS., *Domain Decomposition Methods for Partial Differential Equations*, Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[35] R. GLOWINSKI, Y. A. KUZNETSOV, G. A. MEURANT, J. PÉRIAUX, AND O. WIDLUND, EDS., *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991.

[36] P. GRISVARD, *Elliptic problems in nonsmooth domains*, Pitman, Boston, 1985.

[37] W. D. GROPP AND B. F. SMITH, *Experiences with domain decomposition in three dimensions: Overlapping Schwarz methods*, Tech. Report, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 1992; Proceedings of the Sixth International Symposium on Domain Decomposition Methods, to appear.

[38] V. P. IL'IN, *The Properties of Some Classes of Differentiable Functions of Several Variables Defined in an n-dimensional Region*, Amer. Math. Soc., 81 (1969), pp. 91–256. Originally in Trudy Mat. Inst. Steklov., 66 (1962), pp. 227–363.

[39] J.-L. LIONS AND E. MAGENES, *Nonhomogeneous Boundary Value Problems and Applications*, Vol. I, Springer-Verlag, New York, Heidelberg, Berlin, 1972.

[40] T. P. MATHEW, *Domain Decomposition and Iterative Refinement Methods for Mixed Finite Element Discretisations of Elliptic Problems*, Ph.D. thesis, Mathematics Dept., Courant Institute of Mathematical Sciences, New York University, Sept. 1989; Tech. Report 463, Dept. of Computer Science, Courant Institute of Mathematical Sciences, New York, 1989.

[41] ———, *Schwarz alternating and iterative refinement methods for mixed formulations of elliptic problems, part* I: *Algorithms and numerical results*, Numer. Math., 65 (1993), pp. 445–468.

[42] ———, *Schwarz alternating, iterative refinement and Schur complement based methods for mixed formulations of elliptic problems, part* II: *Convergence theory*, Numer. Math., 65 (1993), pp. 469–492.

[43] A. M. MATSOKIN AND S. V. NEPOMNYASCHIKH, *A Schwarz alternating method in a subspace*, Soviet Math., 29 (1985), pp. 78–84.

[44] S. V. NEPOMNYASCHIKH, *Domain Decomposition and Schwarz Methods in a Subspace for the Approximate Solution of Elliptic Boundary Value Problems*, Ph.D. thesis, Computing Center of the Siberian Branch of the USSR Academy of Sciences, Novosibirsk, USSR, 1986.

[45] J. NEČAS, *Les méthodes directes en théorie des équations elliptiques*, Academia, Prague, 1967.

[46] H. A. SCHWARZ, *Gesammelte Mathematische Abhandlungen*, Vol. 2, Springer-Verlag, Berlin, 1890, pp. 133–143. First published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 (1870), pp. 272–286.

[47] M. D. SKOGEN, *Schwarz Methods and Parallelism*, Ph.D. thesis, Department of Informatics, University of Bergen, Norway, Feb. 1992.

[48] B. F. SMITH, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Ph.D. thesis, Mathematics Dept., Courant Institute of Mathematical Sciences, New York University, Sept. 1990; Tech. Report 517, Dept. of Computer Science, Courant Institute of Mathematical Sciences, New York University, 1990.

[49] ———, *A domain decomposition algorithm for elliptic problems in three dimensions*, Numer. Math., 60 (1991), pp. 219–234.

[50] ———, *An iterative substructuring algorithm for problems in three dimensions*, in Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[51] ———, *An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 364–378.

[52] ———, *A parallel implementation of an iterative substructuring algorithm for problems in three dimensions*, SIAM J. Sci. Comput., 14 (1993), pp. 406–423.

[53] O. B. WIDLUND, *An extension theorem for finite element spaces with three applications*, in Numerical Techniques in Continuum Mechanics, W. Hackbusch and K. Witsch, eds., Braunschweig/Wiesbaden, 1987, Notes on Numerical Fluid Mechanics, V. 16, Friedr. Vieweg und Sohn, pp. 110–122.

[54] ———, *Iterative substructuring methods: Algorithms and theory for elliptic problems in the plane*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[55] ———, *Some Schwarz methods for symmetric and nonsymmetric elliptic problems*, in Fifth Conference on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[56] J. XU, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

[57] ———, *A new class of iterative methods for nonselfadjoint or indefinite problems*, SIAM J. Numer. Anal., 29 (1992), pp: 303–319.

[58] X. ZHANG, *Studies in Domain Decomposition: Multilevel Methods and the Biharmonic Dirichlet Problem*, Ph.D. thesis, Mathematics Dept., Courant Institute of Mathematical Sciences, New York University, Sept. 1991; Tech. Report 584, Dept. of Computer Science, Courant Institute of Mathematical Sciences, New York University, 1991.

[59] ———, *Domain decomposition algorithms for the biharmonic Dirichlet problem*, in Fifth Conference on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[60] ———, *Multilevel Schwarz methods*, Numer. Math., 63 (1992), pp. 521–539.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 621–644, May 1994

009

MULTILEVEL SCHWARZ METHODS FOR THE BIHARMONIC DIRICHLET PROBLEM*

XUEJUN ZHANG†

Abstract. The author considers the solution of the algebraic system of equations that result from the finite element discretization of the biharmonic equation. Some multilevel algorithms are designed and analyzed using a Schwarz framework. Both additive and multiplicative variants of the algorithms are considered and condition number estimates for the additive algorithms and the energy norm estimates for the error propagation operator of the multiplicative algorithms are given. It is noted that for a proper ordering, the iterative operators of the multiplicative algorithms correspond to the error propagation operators of certain V-cycle multigrid methods.

Key words. Schwarz methods, domain decomposition, multilevel methods, multigrid, preconditioned iterative methods, biharmonic problem, plate problem

AMS subject classifications. 65F10, 65N30, 65N55

1. Introduction.

We consider the biharmonic Dirichlet problem on a polygonal region in R^2. The solution is approximated by conforming finite elements. We design and analyze some multilevel methods for solving the resulting linear system. The additive variants of these methods can be considered as generalizations of the hierarchical basis method [23] and the multilevel additive Schwarz (MAS) method [10] designed previously for second-order elliptic problems. We adopt a Schwarz framework to study these methods. When the additive algorithms are used, an equivalent/preconditioned equation is solved by an iterative method such as the conjugate gradient method. We also consider some multiplicative variants of these algorithms and give condition number estimates for the additive variants and energy norm estimates for the error propagation operator of the multiplicative variants.

In this paper, we confine ourselves to the bicubic element discretization. We remark that similar ideas can be used for the Argyris and Bell elements V_A^h and V_B^h; see Ciarlet [7] for properties of these elements. Let $V_l = V_B^{h_l}$ be the Bell elements with respect to a nested triangulation $\mathcal{T}^l$. The new difficulty is that $V_l \not\subset V_{l-1}$, i.e., the subspaces $\{V_l\}$ are not nested. This difficulty can be overcome by using some tricks proposed in [24] for two-level domain decomposition algorithms. We will not further discuss this issue in this paper.

This paper is organized as follows. In §2, we consider the biharmonic problem and its finite element approximation by bicubic elements. We introduce a Schwarz framework in §3. We review some space decomposition and projection techniques, which are used to estimate the rate of convergence of the algorithms. In §4, we study the hierarchical basis (HB) algorithms for the biharmonic problem using an additive Schwarz framework, and we establish that the condition number $\kappa(P_{\mathrm{HB}})$ of the linear system with respect to the normalized hierarchical basis grows like $O(h^{-2})$. We then derive a more efficient algorithm by using additional vertex spaces in the space decomposition. This modified HB (MHB) algorithm costs almost the same as the HB algorithm, but has a condition number that grows like $O(|\log h|^2)$. In §5, we study the MAS method developed in Zhang [24]. This is an additive Schwarz method using a multilevel nodal basis decomposition. Optimal convergence properties of the algorithm are established. In §6, we consider some multilevel multiplicative Schwarz (MMS) algorithms and show that the energy norm of the error propagation operator is bounded by a constant less than one, independent of the number of levels. The MMS algorithm can also be considered as

*Received by the editors June 1, 1992; accepted for publicaton (in revised form) April 30, 1993. This work was supported by National Science Foundation grant ASC-8958544.

†Department of Computer Sciences, University of Maryland, College Park, Maryland 20742 (xzhang@cs.umd.edu).

a certain V-cycle multigrid algorithm and the MAS algorithm can be considered as a parallel multigrid algorithm. The MAS algorithm [10], [26] is quite similar to the BPX algorithm [21], [6]. The MMS and MAS algorithms are also related to, but different from, the fast adaptive composite (FAC) and parallel FAC (AFAC) algorithms developed in [11], [15], [16] and the algorithm studied in [3]. Those papers focused on the partial mesh refinement. In §7, we present some numerical results to verify our theory.

This paper is based in part on Chapter 5 of the author's thesis [24]. We note that recently Oswald [19] established a similar optimal convergence result for the MAS algorithm using Besov space theory.

2. Finite elements for the biharmonic equation.

2.1. The biharmonic problem. We are interested in solving the following biharmonic Dirichlet problem in a plane region: Find $u \in H^2(\Omega)$ such that

$$\begin{aligned} \Delta^2 u &= f \quad \text{in} \quad \Omega, \\ u &= g_0 \quad \text{on} \quad \partial\Omega, \\ \frac{\partial u}{\partial n} &= g_1 \quad \text{on} \quad \partial\Omega. \end{aligned}$$

For simplicity, we assume that Ω is a polygonal region and that $g_0 = g_1 = 0$. We consider the weak formulation: Find $u \in H_0^2(\Omega)$ such that

$$a(u, v) = (f, v) \quad \forall v \in H_0^2(\Omega),$$

where f is a bounded linear functional on $H_0^2(\Omega)$ and $a(u, v)$ is a symmetric, continuous, H_0^2-elliptic bilinear form. Two examples of such bilinear forms are

$$a(u, v) = \int_\Omega \Delta u \Delta v \, dx$$

and

$$a(u, v) = \int_\Omega \left\{ \Delta u \Delta v + (1 - \sigma) \left(2 \frac{\partial^2 u}{\partial x_1 \partial x_2} \frac{\partial^2 v}{\partial x_1 \partial x_2} - \frac{\partial^2 u}{\partial x_1^2} \frac{\partial^2 v}{\partial x_2^2} - \frac{\partial^2 u}{\partial x_2^2} \frac{\partial^2 v}{\partial x_1^2} \right) \right\} dx,$$

where $0 < \sigma < \frac{1}{2}$ is *Poisson's coefficient* of the plate. The first example arises in *fluid dynamics*, and the second provides a variational formulation of the *clamped plate problem*. By the Lax–Milgram theorem, the boundness and ellipticity imply the existence and uniqueness of the solution; cf. [12]. The methods studied in this paper can also be used for other symmetric, positive definite fourth-order problems, e.g.,

$$\Delta^2 u - \lambda \Delta u + \mu u = f, \qquad \lambda, \mu \geq 0.$$

The analysis would be similar to the case of the biharmonic equation.

2.2. The bicubic elements. For the biharmonic equation, the finite elements are all relatively complicated. In this paper, we confine ourselves to a simple case, the bicubic elements, known as the Bogner–Fox–Schmit rectangle. We assume that Ω is a union of rectangles.

We triangulate the domain Ω into nonoverlapping rectangles τ_i called elements; cf. Fig. 1. Let $Q_3 = \text{span}\{x^i y^j, 0 \leq i, j \leq 3\}$. The finite element space $V^h = V_{Q_3}^h$ is defined (cf. Fig. 2) by

$$V^h = \{v : v \in C^1(\Omega) \quad \text{and} \quad v|_{\tau_i} \in Q_3\}.$$

The degrees of freedom of this bicubic element are given by

$$\left\{ p(a_i),\ \frac{\partial p}{\partial x_1}(a_i),\ \frac{\partial p}{\partial x_2}(a_i),\ \frac{\partial^2 p}{\partial x_1 \partial x_2}(a_i) \right\},$$

where a_i is a vertex of an element. The finite element solution $u_h \in V^h$ satisfies

$$a(u_h, \phi_h) = f(\phi_h) \quad \forall \phi_h \in V^h.$$

Let $\mathcal{D}_1 = \{(0,0), (1,0), (0,1)\}$ and $\mathcal{D}_2 = \{(0,0), (1,0), (0,1), (1,1)\}$. We use the multi-indices notation to represent the order of derivatives. The nodal basis functions $\phi_i^\alpha(x)$ are defined by

$$\frac{\partial^{|\beta|}}{\partial x^\beta} \phi_i^\alpha(x_j) = \delta_{i,j}\delta_{\alpha,\beta}, \qquad \alpha, \beta \in \mathcal{D}_2.$$

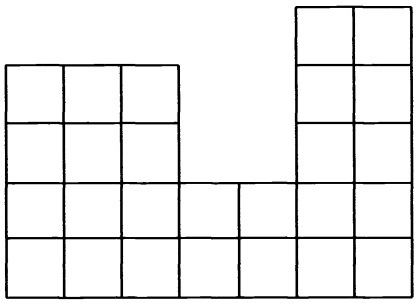


FIG. 1. *A uniform triangulation* $\mathcal{T}^1$, $h_x = h_y$.

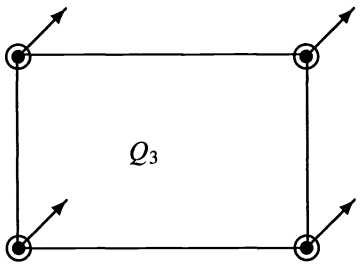


FIG. 2. *The bicubic elements.*

Representing u_h by the basis ϕ_i^α as

$$u_h = \sum_i \sum_{\alpha \in \mathcal{D}_2} x_i^\alpha \phi_i^\alpha,$$

we obtain a linear system

$$K_h x = b,$$

where $K_h = \{a(\phi_i^\alpha, \phi_j^\beta)\}$ and $b_i^\alpha = f(\phi_i^\alpha)$.

The stiffness matrix K_h is symmetric, positive definite. After a proper scaling, its condition number $\kappa(K_h)$ is on the order of h^{-4}. Since the linear system can be quite large, the condition number of K_h can also be very large, and solving the system can be very expensive.

2.3. Nested triangulations and subspaces. To construct and study multilevel algorithms, we need to define a sequence of nested rectangular triangulations $\{\mathcal{T}^l\}_{l=1}^L$. We start with a coarse triangulation $\mathcal{T}^1 = \{\tau_i^1\}_{i=1}^{N_1}$, where τ_i^1 represents an individual rectangle; cf. Fig. 1. The successively finer triangulations $\mathcal{T}^l = \{\tau_i^l\}_{i=1}^{N_l}$ are obtained by dividing the rectangles of the triangulation $\mathcal{T}^{l-1}$ into four rectangles (not necessarily of equal sizes). We denote by $h_x(\tau_i^l)$ and $h_y(\tau_i^l)$ the lengths of the edges of τ_i^l in the x and y directions, respectively, and we assume that all the triangulations are shape regular in the sense that $\min(h_x/h_y, h_y/h_x) \geq T_{\min}$, with $T_{\min} > 0$. Let $h_{li} = \text{diameter}(\tau_i^l)$, $h_l = \max_i h_{li}$, and $h = h_L$.

As a consequence of the shape regularity assumption, we know that there exist constants $C > 0$ and $0 < \gamma < 1$, such that if a rectangle τ_i^{l+k} of level $l+k$ lies in a rectangle τ_j^l of level l, then

$$\frac{\text{diam}(\tau_i^{l+k})}{\text{diam}(\tau_j^l)} \leq C\gamma^k.$$

For a uniform refinement, $\gamma = \frac{1}{2}$ and $C = 1$. We denote by $\mathcal{N}^l$ and $\mathcal{E}^l$ the sets of vertices and edges of $\mathcal{T}^l$ and by $\mathcal{E}^l(S)$ the edges of the subset S.

Let $V_l, l = 1, \ldots, L$, be the space of bicubic elements associated with the triangulation $\mathcal{T}^l$. The finite element problem can be rewritten in the new notation as: Find $u \in V_L$ such that

$$a(u, \phi) = f(\phi) \quad \forall \phi \in V_L. \tag{1}$$

We denote the nodal basis functions of V^l by $\phi_{li}^\alpha, i \in \mathcal{N}^l, \alpha \in \mathcal{D}_2$. Let $\Omega_i^l = \text{supp}\{\phi_{li}^\alpha\}$, $V_{li}^\alpha = \text{span}\{\phi_{li}^\alpha\}$ and $V_{li} = V_l \cap H_0^2(\Omega_i^l) = \sum_{\alpha \in \mathcal{D}_2} V_{li}^\alpha$. We note that for the bicubic element $\dim\{V_{li}\} = 4$.

3. A Schwarz framework. In this section, we review some tools for the study of Schwarz methods; see Dryja and Widlund [8], [9] and Matsokin and Nepomnyaschikh [14] for the additive methods and Bramble et al. [5] and Xu [22] for the multiplicative methods.

3.1. Additive Schwarz method. Consider the finite element problem: Find $u \in V$ such that

$$a(u, v) = (f, v) \quad \forall v \in V \quad \text{or} \quad Au = f. \tag{2}$$

Suppose that the finite element space V can be written as a sum of subspaces,

$$V = V_1 + V_2 + \cdots + V_N.$$

The projections $P_{V_i} : V \to V_i$ and $Q_{V_i} : V \to V_i$, are defined by

$$\begin{aligned} a(P_{V_i}u, \phi) &= a(u, \phi) \quad \forall \phi \in V_i, \\ (Q_{V_i}u, \phi) &= (u, \phi) \quad \forall \phi \in V_i. \end{aligned}$$

Let A_{V_i} be the restriction of A to the subspace V_i given by

$$(A_{V_i}u, v) = a(u, v) \quad \forall u, v \in V_i.$$

Based on the above space decomposition and projections, we define a preconditioner

$$B_{as}^{-1} = \sum_i A_{V_i}^{-1} Q_{V_i}$$

and an additive Schwarz operator

$$P_{as} = B_{as}^{-1}A = \sum_i P_{V_i}.$$

ALGORITHM 3.1 (AS ALGORITHM). Find the solution u to (2) by solving iteratively the additive Schwarz equation

$$P_{as}u_h = (P_{V_1} + P_{V_2} + \cdots + P_{V_N})u_h = f_{as} \equiv B_{as}^{-1}f. \tag{3}$$

We note that $f_{as} = B_{as}^{-1}f = \sum_i f_i$, where $f_i = A_{V_i}^{-1}Q_{V_i}f$ are the solutions for

$$a(f_i, \phi_h) = a(P_{V_i}u, \phi_h) = (f, \phi_h) \quad \forall \phi_h \in V_i. \tag{4}$$

It is easy to see that for f_i given by (4), the additive Schwarz equation (3), and the original finite element equation (2) are equivalent. The crucial issue is the condition number of the iteration operator P_{as}. From the definitions of B_{as} and P_{as}, we have $a(P_{as}^{-1}u, u) = (B_{as}u, v)$. The following lemma characterizes the extreme eigenvalues of P_{as}.

LEMMA 3.1. *Let V be a Hilbert space, V_i be subspaces of V and $V = \sum V_i$. Let P_{V_i} be the projections from V onto V_i, and $P = \sum_i P_{V_i}$. Then*

$$\lambda_{\min}(P) = \min_u \frac{a(u,u)}{a(P^{-1}u, u)} = \min_u \frac{a(u,u)}{\min\limits_{\sum u_i = u} \sum_i a(u_i, u_i)}, \qquad u_i \in V_i$$

and

$$\lambda_{\max}(P) = \max_u \frac{a(u,u)}{a(P^{-1}u, u)} = \max_u \frac{a(u,u)}{\min\limits_{\sum u_i = u} \sum_i a(u_i, u_i)}, \qquad u_i \in V_i.$$

Proof. See Lemma 3.1 in [26] for the proof. □

Remark. 3.1. If we can find constant C_1 and C_2 such that there exists a decomposition of $u = \sum_i u_i$ satisfying

$$C_1 \sum_i a(u_i, u_i) \le a(u,u) \quad \forall u \in V,$$

and for all $u \in V$ and for any decomposition of $u = \sum_i u_i$, we have

$$a(u,u) \le C_2 \sum_i a(u_i, u_i).$$

Then it follows from Lemma 3.1 that $\lambda_{\min}(P) \ge C_1$ and $\lambda_{\max}(P) \le C_2$. The first part is known as the Lions lemma; cf. Dryja and Widlund [8]–[10] and Nepomnyaschikh [17]. Similar results for the case $V = \oplus V_i$ can be found in Mandel and McCormick [13], where the "min" in the denominators is dropped, since the decomposition of a function is unique.

For $u, v \in V$, let $\cos(u, v) = a(u, v)/\|u\|_a\|v\|_a$. For V_1, V_2, two nontrivial subspaces of V, we define the cosine of the angle between V_1 and V_2 by

$$\cos(V_1, V_2) \stackrel{\text{def}}{=} \sup_{u_1 \in V_1, u_2 \in V_2} \cos(u_1, u_2). \tag{5}$$

Let $\theta_{ij} = \cos(V_i, V_j)$, $\Theta = \{\theta_{ij}\}$, and let $u = \sum_i u_i$, $u_i \in V_i$ be any decomposition of u. Then

$$a(u,u) = \sum_{i,j} a(u_i, u_j) \le \sum_{i,j} \theta_{ij}\|u_i\|_a\|u_j\|_a \le \|\Theta\|_2 \sum_i a(u_i, u_i).$$

Thus, $\lambda_{\max}(P) \le \rho(\Theta) = \|\Theta\|_2$.

3.2. Multiplicative Schwarz method. Algorithm 3.1 can be considered as a Jacobi method or simultaneous subspace correction. We can of course also define a Gauss–Seidel method or sequential subspace correction algorithm.

Consider solving (2) by the following procedure

ALGORITHM 3.2 (MS ALGORITHM). Let u^{old} be the current approximation. Compute

$$u_0 = u^{\text{old}}, \quad u_i = u_{i-1} + A_{V_i}^{-1} Q_{V_i}(f - Au_{i-1}), \quad i = 1, \ldots, N, \quad u^{\text{new}} = u_N.$$

It is easy to see that the error propagation operator is

$$E = (I - P_{V_N}) \cdots (I - P_{V_2}) \cdot (I - P_{V_1}).$$

In the case $V_i = \text{span}\{\phi_i\}$, $A_{V_i} = a(\phi_i, \phi_i)$, where ϕ_i is a nodal basis function, Algorithm 3.2 is the classical Gauss–Seidel algorithm.

In general, computing $A_{V_i}^{-1}$ (solving $A_{V_i} y = z$) is relatively expensive, and we can replace $A_{V_i}^{-1}$ by a preconditioner or a smoother R_i. Let $T_i = R_i Q_{V_i} A$. Then the corresponding error propagation operator is

$$E = (I - T_N) \cdots (I - T_2) \cdot (I - T_1).$$

The convergence rate is determined by $\|E\|_a$, the energy norm of E. We will derive bounds for $\|E\|_a$ using the properties of T_i and subspace V_i.

Let T_i, $i = 1, \ldots, N$ be symmetric, semipositive definite operators with respect to $a(\cdot, \cdot)$, and let $T = \sum_{i=1}^{N} T_i$. Let $E_0 = I$, $E_i = (I - T_i)E_{i-1}$, and $E = E_N$.

Let $\Theta_T = \{\theta_T^{ij}\}$, where $\theta_T^{ii} = 1$ and θ_T^{ij}, $i \neq j$ are given by

$$\theta_T^{i,j} = \sup_{u,v} \frac{a(T_i u, T_j v)}{a(T_i u, u)^{1/2} a(T_j v, v)^{1/2}}. \tag{6}$$

We note that when T_i are projections, definitions (6) and (5) are identical.

THEOREM 3.1. *Assume* $\|T_i\|_a \leq \omega < 2$. *Then*

$$1 - \frac{2\lambda_{\min}(T)}{2 - \omega + \lambda_{\min}(T)} \leq \|E\|_a \leq \sqrt{1 - (2 - \omega)\frac{\lambda_{\min}(T)}{\|\Theta_T\|_2^2}}.$$

The second inequality is a slightly modified version of a theorem given in Xu [22]. See [26] for a proof of Theorem 3.1.

Theorem 3.1 can be made more precise if $\dim(V_i) = 1$ and $T_i = P_{V_i}$ for all i. Let $\phi_i \in V_i$, $|\phi_i|_a = 1$, then $\cos(V_i, V_j) = |a(\phi_i, \phi_j)|$. Let $K = \{a(\phi_i, \phi_j)\}$. Then, K is a semidefinite matrix and $\Theta_T = \{\cos(V_i, V_j)\} = |K|$. Since, usually, $\{\phi_i\}$ are linearly dependent, the "stiffness matrix" K is usually singular and $\text{rank}(K) = \dim(V)$. Let $K = D - L - L^t$, where D is the diagonal of K and $-L$ and $-L^t$ are the strictly lower and upper triangular parts of K. Then, the $\|\Theta_T\|_2$ in Theorem 3.1 can be replaced by $\|(D - L)\|_2$; it is easy to see that $\|D - L\|_2 \leq \|\Theta_T\|_2$. Theorem 3.1 also provides an energy norm bound for the Gauss–Seidel method.

4. The HB methods.

4.1. Some interpolation operators. In this subsection, we state and prove some properties of certain interpolation operators defined on V_L.

Let $\Pi^l = \Pi^{h_l}$ be the standard nodal value interpolation operator defined by

$$\Pi^l u(x) = \sum_{i \in \mathcal{N}^l} \sum_{\alpha \in \mathcal{D}_2} D^\alpha u(x_i) \phi_{li}^\alpha(x). \tag{7}$$

LEMMA 4.1. *We have the following estimate for* Π^l:

$$|\Pi^l u - u|_{H^s} \le C h_l^{2-s} \left(\frac{h_l}{h_L}\right) |u|_{H^2} \le C 2^{l(s-2)} 2^{L-l} |u|_{H^2}, \quad u \in V_L, \quad s = 0, 1, 2.$$

Proof. This is Lemma 4.2.3 in [24]. □

Let $\tilde{\Pi}^l = \tilde{\Pi}^{h_l}$ be an interpolation operator defined by

$$\tilde{\Pi}^l u(x) = \sum_{i \in \mathcal{N}^l} \sum_{\alpha \in \mathcal{D}_1} D^\alpha u(x_i) \phi_{li}^\alpha(x).$$

We note that the mixed second derivative vanishes at level-ℓ grid points, that is, $D^{(1,1)} \tilde{\Pi}^l u(x) = 0$ for $x \in \mathcal{N}^l$. We note also that $\tilde{\Pi}^l p = p$ for p linear.

LEMMA 4.2. *Let* $\tilde{\Pi}^l$ *be the interpolation operator defined above. Then,*

$$|\tilde{\Pi}^l u - u|_{H^s} \le C h_l^{2-s} \left(1 + \log \frac{h_l}{h_L}\right)^{1/2} |u|_{H^2} \le C 2^{l(s-2)} (L-l)^{1/2} |u|_{H^2}, \quad u \in V_L, \quad s = 0, 1, 2.$$

Proof. This is Lemma 4.2.4 in [24]. We only sketch a proof here. Let $\tau \in \mathcal{T}^l$ be an element of level l and x_i, $1 \le i \le 4$, be its vertices. We have

$$\tilde{\Pi}^l u = \sum_{i=1}^{4} \sum_{|\alpha| \le 1} D^\alpha u(x_i) \phi_{li}^\alpha(x) \quad \forall x \in \tau.$$

We want to estimate $|u_h - \tilde{\Pi}^l u_h|_{H^s(\tau)}$. Note that $\tilde{\Pi}^l(p) = p$ for p linear. We can assume without loss of generality that

$$\{D^\alpha u\}_\tau \equiv \frac{\int_\tau D^\alpha u dx}{|\tau|} = 0, \ |\alpha| = 0, 1.$$

By the triangle inequality, we have

$$|\tilde{\Pi}^l u|_{H^s(\tau)} \le \sum_{i=1}^{4} \sum_{|\alpha| \le 1} |D^\alpha u(x_i)| |\phi_{li}^\alpha(x)|_{H^s(\tau)} \le \sum_{i=1}^{4} \sum_{|\alpha| \le 1} |u|_{W^{|\alpha|,\infty}(\tau)} |\phi_{li}^\alpha|_{H^s(\tau)}.$$

Using the fact that the basis functions are uniform to order 2, i.e.,

$$|\phi_{li}^\alpha(x)|_{W^{s,\infty}(\tau)} \le C h_l^{|\alpha|-s} \quad \text{and} \quad |\phi_{li}^\alpha(x)|_{H^s(\tau)} \le C h_l^{|\alpha|+1-s},$$

we get

$$|\tilde{\Pi}^l u|_{H^s(\tau)} \le C \left\{h_l^{1-s} |u|_{L^\infty(\tau)} + h_l^{2-s} |u|_{W^{1,\infty}(\tau)}\right\}.$$

Since $\{D^\alpha u\}_\tau = 0$, $|\alpha| = 0, 1$, we find that

$$|u|_{L^\infty(\tau)} \le C h_l |u|_{H^2(\tau)},$$
$$|u|_{W^{1,\infty}(\tau)} \le C \left(1 + \log\left(\frac{h_l}{h}\right)\right)^{1/2} |u|_{H^2(\tau)}.$$

The first inequality follows by the embedding $L^\infty \hookrightarrow H^2$ and a scaling argument. The second inequality follows from an application of Lemma 3 in the Appendix of Yserentant [23] to ∇u.

Using these inequalities in the estimate for $|\tilde{\Pi}^l u|_{H^s(\tau)}$, we obtain

$$|\tilde{\Pi}^l u|_{H^s(\tau)} \le C\left\{h_l^{2-s}|u|_{H^2(\tau)} + h_l^{2-s}\left(1+\log\left(\frac{h_l}{h_L}\right)\right)^{1/2}|u|_{H^2(\tau)}\right\}$$
$$\le Ch_l^{2-s}\left(1+\log\left(\frac{h_l}{h_L}\right)\right)^{1/2}|u|_{H^2(\tau)}.$$

By the triangle inequality and the vanishing mean values property of u, we have

$$|u-\tilde{\Pi}^l u|_{H^s(\tau)} \le |u|_{H^s(\tau)} + |\tilde{\Pi}^l u|_{H^s(\tau)} \le Ch_l^{2-s}\left(1+\log\left(\frac{h_l}{h_L}\right)\right)^{1/2}|u|_{H^2(\tau)}.$$

The lemma follows by summing over all $\tau \in \mathcal{T}^l$. □

It can be shown that the estimates in Lemmas 4.1 and 4.2 cannot be improved.

4.2. The HB method. We now study the HB method for the biharmonic problem. Let $\tilde{\mathcal{N}}^1 = \mathcal{N}^1$ and $\tilde{\mathcal{N}}^l = \mathcal{N}^l \setminus \mathcal{N}^{l-1}$ for $2 \le l \le L$, where $\mathcal{N}^l$ is the set of vertices of $\mathcal{T}^l$ as defined in §2.3. Let K_{HB} be the stiffness matrix with respect to the HB $\{\phi_{li}^\alpha\}_{\alpha\in\mathcal{D}_2, i\in\tilde{\mathcal{N}}^l, l\le L}$. To simplify the presentation, we assume that $h_1 = O(1)$, i.e., the coarse triangulation $\mathcal{T}^1$ is coarse enough. Therefore, the coarse stiffness matrix $K_1 = \{a(\phi_{1i}^\alpha, \phi_{1j}^\beta)\}$ can be replaced by D_1, the diagonal of K_1, and $\kappa(D_1^{-1}K_1)$ is small. In the general case, we need to solve the coarse problem exactly and the algorithms and analysis can be easily modified.

If we represent a function $u \in V_L$ by this HB as

$$u = \sum_{l=1}^{L}\sum_{i\in\tilde{\mathcal{N}}^l}\sum_{\alpha\in\mathcal{D}_2} x_{li}^\alpha \phi_{li}^\alpha,$$

then the extreme eigenvalues of K_{HB} can be estimated as follows:

$$\lambda_{\min}(K_{\mathrm{HB}}) = \min_x \frac{(K_{\mathrm{HB}}x, x)}{\|x\|_2^2} = \min \frac{|u|_{H^2}^2}{\|x\|_2^2},$$
$$\lambda_{\max}(K_{\mathrm{HB}}) = \max_x \frac{(K_{\mathrm{HB}}x, x)}{\|x\|_2^2} = \max \frac{|u|_{H^2}^2}{\|x\|_2^2}.$$

Note that $|\phi^{(0,0)}|_{H^2}^2 = O(h^{-2})$ and $|\phi^{(1,1)}|_{H^2}^2 = O(h^2)$. The unbalance between $\phi^{(1,1)}$ and $\phi^{(0,0)}$ implies that $\kappa(K_{\mathrm{HB}}) \ge ch^{-4}$. This suggests that we should use the normalized basis function (with respect to the energy norm), or equivalently a diagonal scaling technique. The diagonal scaling is inherent in the additive Schwarz method; therefore one does not need to keep track of scaling for different degrees of freedom. We will show that the simple diagonal scaling reduces the condition number to $O(h^{-2})$.

Let $\Pi^0 = 0$. By definition (7), we have

$$(\Pi^l - \Pi^{l-1})u = \sum_{i\in\tilde{\mathcal{N}}^l}\sum_{\alpha\in\mathcal{D}_2} x_{li}^\alpha \phi_{li}^\alpha.$$

It is easy to show that

$$|(\Pi^l - \Pi^{l-1})u|_{H^2}^2 \approx C\sum_{i\in\tilde{\mathcal{N}}^l}\sum_{\alpha\in\mathcal{D}_2}|x_{li}^\alpha|^2|\phi_{li}^\alpha|_{H^2}^2 \approx C\sum_{i\in\tilde{\mathcal{N}}^l}\sum_{\alpha\in\mathcal{D}_2}|x_{li}^\alpha|^2 h_l^{2|\alpha|-2}.$$

If we scale the basis functions such that $|\phi_{li}^{\alpha}|_{H^2}^2 = O(1)$, then

$$|(\Pi^l - \Pi^{l-1})u|_{H^2}^2 \approx \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} |x_{li}^{\alpha}|^2.$$

Thus,

$$\lambda_{\min}(K_{\mathrm{HB}}) = \min_x \frac{(K_{\mathrm{HB}}x, x)}{\|x\|_2^2} \approx \min_u \frac{|u|_{H^2}^2}{\sum_l |(\Pi^l - \Pi^{l-1})u|_{H^2}^2},$$

$$\lambda_{\max}(K_{\mathrm{HB}}) = \max_x \frac{(K_{\mathrm{HB}}x, x)}{\|x\|_2^2} \approx \max_u \frac{|u|_{H^2}^2}{\sum_l |(\Pi^l - \Pi^{l-1})u|_{H^2}^2}.$$

We will not discuss this direction further. Instead, we will use the additive Schwarz techniques to study this method. Based on the HB decomposition

$$V_L = V_1 + \sum_{l=2}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} V_{li}^{\alpha},$$

we define a preconditioner

$$B_{\mathrm{HB}}^{-1} = A_{V_1}^{-1} Q_{V_1} + \sum_{l=2}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha} A_{V_{li}^{\alpha}}^{-1} Q_{V_{li}^{\alpha}}$$

and an additive Schwarz operator

$$P_{\mathrm{HB}} = B_{\mathrm{HB}}^{-1} A = P_{V_1} + \sum_{l=2}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} P_{V_{li}^{\alpha}}.$$

ALGORITHM 4.1 (HB). Find the solution u of the finite element problem (1) by solving iteratively the additive Schwarz equation

$$P_{\mathrm{HB}} u = f_{\mathrm{HB}} \stackrel{\mathrm{def}}{=} B_{\mathrm{HB}}^{-1} f.$$

We note that $f_{\mathrm{HB}} = B_{\mathrm{HB}}^{-1} f = \sum_l \sum_i \sum_{\alpha} f_{li}^{\alpha}$ and $f_{li}^{\alpha} = A_{V_{li}^{\alpha}}^{-1} Q_{V_{li}^{\alpha}} f$ are the solutions of

$$a(f_{li}^{\alpha}, \phi) = (f, \phi) \quad \forall \phi \in V_{li}^{\alpha}.$$

Since $\dim(V_{li}^{\alpha}) = 1$, it is easy to see that

$$P_{V_{li}^{\alpha}} v = \frac{a(v, \phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha} \quad \text{and} \quad f_{li}^{\alpha} = \frac{f(\phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha}.$$

If we replace P_{V_1} by $\sum_{i \in \mathcal{N}^1} \sum_{\alpha} P_{V_{1,i}^{\alpha}}$, then the above additive Schwarz equation can be written down explicitly as

$$\sum_{l=1}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} \frac{a(u, \phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha} = \sum_{l=1}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} \frac{f(\phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha}.$$

The corresponding linear system is

$$D^{-1} K_{\mathrm{HB}} x = D^{-1} b,$$

where $D = \operatorname{diag}(K_1, D_2, \ldots, D_L)$ or $D = \operatorname{diag}(D_1, D_2, \ldots, D_L)$, K_l is the stiffness matrix associated with $\{\phi_{li}^{\alpha}\}_{i \in \tilde{\mathcal{N}}^l}$, and $D_l = \operatorname{diag}(K_l)$.

LEMMA 4.3. *There exist constants C_1 and C_2 such that*

$$C_1 h^{-2} a(u,u) \le a(P_{\mathrm{HB}} u, u) \le C_2 a(u,u) \quad \forall u \in V^L.$$

The estimate cannot be improved, i.e., $c_1 h^{-2} \le \kappa(D^{-1} K_{\mathrm{HB}}) \le c_2 h^{-2}$.

Proof. Let $u_l = (\Pi^l - \Pi^{l-1})u$; it follows from Lemma 4.1 that

$$|u_l|_{H^2}^2 = |(\Pi^l - \Pi^{l-1})u|_{H^2(\Omega)}^2 \le C 4^{L-l} |u|_{H^2(\Omega)}^2.$$

It is easy to see that u_l can be represented uniquely as

$$u_l = \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} u_{li}^{\alpha}, \qquad u_{li}^{\alpha} \in V_{li}^{\alpha}.$$

By the inverse inequality and Lemma 4.1, we obtain

$$\begin{aligned} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} |u_{li}^{\alpha}|_{H^2}^2 &\le C \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} h_l^{-4} |u_{li}^{\alpha}|_{L^2}^2 \le C h_l^{-4} |u_l|_{L^2}^2 \\ &\le C h_l^{-4} C h_l^4 \left(\frac{h_l}{h_L}\right)^2 |u|_{H^2}^2 \le C 4^{(L-l)} |u|_{H^2}^2. \end{aligned}$$

Summing over l gives

$$u_1 + \sum_{l=2}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} |u_{li}^{\alpha}|_{H^2}^2 \le \sum_{l} C 4^{L-l} |u|_{H^2}^2 \le C 4^L |u|_{H^2}^2.$$

The lower bound follows from Lemma 3.1.

The upper bound, $a(P_{\mathrm{HB}}, u, u) \le C$, can be established by using certain strengthened Cauchy–Schwarz inequality. We will not give a proof here, since it follows easily from Theorem 5.1 and Remark 5.2 of the next section.

To show that the estimate is sharp, let $u = \phi_{Lj}^{(1,1)}$, $j \in \mathcal{N}^1$. The HB decomposition of u is given by

$$u = \sum_{l} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} u_{li}^{\alpha}.$$

We have $u_{1j}^{(1,1)} = \phi_{1j}^{(1,1)}$, and note that $|\phi_{1j}^{(1,1)}|_{H^2}^2 = O(h_1^2)$ and $|u|_{H^2}^2 = |\phi_{Lj}^{(1,1)}|_{H^2}^2 = O(h_L^2)$. Thus

$$\sum_{l=1}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} |u_{li}|_{H^2}^2 \ge |\phi_{1j}^{(1,1)}|_{H^2}^2 \ge C \left(\frac{h_1}{h_L}\right)^2 |u|_{H^2}^2 \ge C 4^L |u|_{H^2}^2.$$

It follows from Lemma 3.1 that $\lambda_{\min}(P_{\mathrm{HB}}) \le C h^{-2}$. □

Remark 4.1. In [20], Oswald studied the biharmonic problem approximated by composite finite elements consisting of piecewise quadratic Powell–Sabin elements and piecewise cubic elements of Clough–Tocher type. Using Besov space theory, Oswald established that the condition number of hierarchical multilevel methods grows like CL^2, which is significantly better than the case of bicubic element. This is because the second derivatives of the finite element function are not degrees of freedoms of the two composite finite elements considered by Oswald. In general, for symmetric elliptic problems of $2m$th order, if the degrees of freedom of the finite elements consists of function values and derivatives of order less than or equal to $m-1$, then, $\kappa(K_{\mathrm{HB}}) \le CL^2$.

4.3. A MHB method. To get better algorithms, we need to improve $\lambda_{\min}(P)$, the minimum eigenvalue of the iteration operator. If we use additional subspaces in the space decomposition, we have more freedom to choose the decomposition of a function $u \in V_L$ and by Lemma 3.1 we have a larger $\lambda_{\min}(P)$. Our approach is to add some additional vertex subspaces in the space decomposition. Using the space decomposition

$$V^L = V_1 + \sum_{l=2}^{L} \sum_{i \in \mathcal{N}^l} \sum_{\alpha \in \mathcal{D}_2} V_{li}^{\alpha} + \sum_{i \in \mathcal{N}^L} V_{Li}^{(1,1)},$$

we define a preconditioner

$$B_{\mathrm{MHB}}^{-1} = B_{\mathrm{HB}}^{-1} + \sum_{i \in \mathcal{N}^L} A_{V_{Li}^{(1,1)}}^{-1} Q_{V_{Li}^{(1,1)}}$$

and an additive Schwarz iteration operator

$$P_{\mathrm{MHB}} = B_{\mathrm{MHB}}^{-1} A = P_{\mathrm{HB}} + \sum_{i \in \mathcal{N}^L} P_{V_{Li}^{(1,1)}}.$$

ALGORITHM 4.2 (MHB). Find the solution u of the finite element equation (1) by solving the following additive Schwarz equation:

$$P_{\mathrm{MHB}} u = f_{\mathrm{MHB}},$$

where

$$f_{\mathrm{MHB}} = B_{\mathrm{MHB}}^{-1} f = f_{\mathrm{HB}} + \sum_{i \in \mathcal{N}^L} A_{V_{Li}^{(1,1)}}^{-1} Q_{V_{Li}^{(1,1)}} f = f_{\mathrm{HB}} + \sum_{i \in \mathcal{N}^L} \frac{f(\phi_{Li}^{(1,1)})}{a(\phi_{Li}^{(1,1)}, \phi_{Li}^{(1,1)})} \phi_{Li}^{\alpha}.$$

This algorithm costs a little more than the HB algorithm per iteration, since we need to compute the extra terms

$$P_{V_{Li}^{(1,1)}} u = \frac{a(u, \phi_{Li}^{(1,1)})}{a(\phi_{Li}^{(1,1)}, \phi_{Li}^{(1,1)})} \phi_{Li}^{(1,1)}, \qquad i = 1, \ldots, N_L$$

in each iteration. This requires N operations, where $N = N_L = \dim \mathcal{N}^L$. Compared with the cost for one multiplication Kx, which is about $144N$, this cost is small (less than 1% of the computation).

The following lemma provides a condition number estimates of P_{MHB}.

LEMMA 4.4. *There exist constants C_1 and C_2 such that*

$$C_1 |\log h|^{-2} a(u, u) \le a(P_{\mathrm{MHB}} u, u) \le C_2 a(u, u) \quad \forall u \in V^L.$$

Thus the condition number of the MHB *algorithm is $O(|\log h|^2) = O(L^2)$.*

Proof. We apply Lemma 3.1 to establish the lower bound. Let

$$u^{(1,1)} = \sum_{i \in \mathcal{N}^L} u_{xy}(x_i) \phi_{Li}^{(1,1)}.$$

It is easy to see that $u^{(1,1)}$ has the same mixed second derivatives as u, but has zero function values and first derivatives at the grid points $x \in \mathcal{N}^L$. Let $\Omega_i^L = \mathrm{supp}\{\phi_{Li}^{\alpha}\}$. Using the discrete norm estimates, $|u_{xy}(x_i)|^2 h^2 \le C|u|^2_{H^2(\Omega_i^L)}$, we obtain

$$\sum_{i \in \mathcal{N}^L} |u_{xy}(x_i) \phi_{Li}^{(1,1)}|^2_{H^2(\Omega)} \le \sum_{i \in \mathcal{N}^L} |u_{xy}(x_i)|^2 |\phi_{Li}^{(1,1)}|^2_{H^2(\Omega)} \le C \sum_{i \in \mathcal{N}^L} |u_{xy}(x_i)|^2 h^2$$

$$\le C \sum_{i \in \mathcal{N}^L} |u|^2_{H^2(\Omega_i^L)} \le C|u|^2_{H^2(\Omega)}.$$

Let $u_l = \tilde{\Pi}^l u - \tilde{\Pi}^{l-1} u$. Then,

$$u = u^{(1,1)} + \tilde{\Pi}^L u = \sum_{i \in \mathcal{N}^L} u_{xy}(x_i)\phi_{Li}^{(1,1)} + \sum_{l=1}^{L} u_l.$$

It is easy to check that

$$D^\alpha u_l(x_i) = 0, \qquad x_i \in \mathcal{N}^{l-1}, \quad \alpha \in \mathcal{D}_2.$$

Thus u_l can be uniquely represented as

$$u_l = \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} u_{li}^\alpha, \qquad u_{li}^\alpha \in V_{li}^\alpha.$$

By the inverse inequality and Lemma 4.2, we obtain

$$\sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} |u_{li}^\alpha|_{H^2}^2 \le C \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} h_l^{-4} |u_{li}^\alpha|_{L^2}^2 \le C h_l^{-4} |u_l|_{L^2}^2 \le C(L-l)|u|_{H^2}^2.$$

Combining the above two estimates, we get

$$\begin{aligned} &\sum_{i \in \mathcal{N}^L} |u_{xy}(x_i)\phi_{Li}^{(1,1)}|_{H^2}^2 + \sum_{l=1}^{L} \sum_{i \in \tilde{\mathcal{N}}^l} \sum_{\alpha \in \mathcal{D}_2} |u_{li}^\alpha|_{H^2}^2 \\ &\le \quad C|u|_{H^2}^2 + C \sum_{l=1}^{L} (L-l)|u|_{H^2}^2 \le CL^2 |u|_{H^2}^2, \qquad u_{li}^\alpha \in V_{li}^\alpha \end{aligned}$$

The lower bound follows from Lemma 3.1.

The upper bound, $a(P_{\mathrm{MHB}} u, u) \le Ca(u, u)$, follows from Theorem 5.1 and Remark 5.2 in the next section. □

5. MAS methods.

5.1. Algorithm. MAS methods for the biharmonic problem were designed and studied in [25] and [24]. They are generalizations of the two level domain decomposition methods also considered in [24]. In matrix form, they correspond to *multilevel block diagonal scaling*.

Let $\Omega_i^l = \mathrm{supp}\{\phi_{li}^\alpha\}$. The MAS method can be derived from the multilevel overlapping domain decomposition $\Omega = \cup_i \Omega_i^l$, using domain decomposition language; cf. [24]. In this paper, we work directly with the multilevel space decomposition.

Let $V_{li}^\alpha = \mathrm{span}\{\phi_{li}^\alpha\}$ and let $V_{li} = V_l \cap H_0^2(\Omega_i^l) = \sum_\alpha V_{li}^\alpha$ be defined as in §4. For uniform triangulations, $V_{li}^\alpha \perp V_{li}^\beta$, for $\alpha \ne \beta$, thus $V_{li} = \sum_\alpha V_{li}^\alpha$ is an orthogonal decomposition and $P_{V_{li}} u = \sum_\alpha P_{V_{li}^\alpha}$. The 4-by-4 local stiffness matrices $K_{li} = \{a(\phi_{li}^\alpha, \phi_{li}^\beta)\}$ are diagonal.

Based on the MAS space decomposition

$$V^h = V_1 + \sum_{l=2}^{L} \sum_{i \in \mathcal{N}^l} V_{li} \qquad \left(V_1 + \sum_{l=2}^{L} \sum_{i \in \mathcal{N}^l} \sum_{\alpha \in \mathcal{D}_2} V_{li}^\alpha \right),$$

we defined a preconditioner B_{MAS}^{-1}

$$B_{\mathrm{MAS}}^{-1} = A_{V_1}^{-1} Q_{V_1}^{-1} + \sum_{\ell \ge 2} \sum_{i \in \mathcal{N}^l} A_{V_{li}}^{-1} Q_{V_{li}} \qquad \left(A_{V_1}^{-1} Q_{V_1} + \sum_{\ell \ge 2} \sum_{i \in \mathcal{N}^l} \sum_{\alpha \in \mathcal{D}_2} A_{V_{li}^\alpha}^{-1} Q_{V_{li}^\alpha} \right)$$

and an additive Schwarz operator P_{MAS}

$$P_{\mathrm{MAS}} = B_{\mathrm{MAS}}^{-1} A = P_{V_1} + \sum_{\ell \geq 2} \sum_{i \in \mathcal{N}^l} P_{V_{li}} \qquad \left(P_{V_1} + \sum_{\ell \geq 2} \sum_{i \in \mathcal{N}^l} \sum_{\alpha \in \mathcal{D}_2} P_{V_{li}^{\alpha}} \right).$$

ALGORITHM 5.1 (MAS). Find the solution u_h of the finite element equation (1) by solving iteratively the equivalent equation:

$$P_{\mathrm{MAS}} u_h = f_{\mathrm{MAS}} \stackrel{\mathrm{def}}{=} B_{\mathrm{MAS}}^{-1} f. \tag{8}$$

Note that $f_{\mathrm{MAS}} = B_{\mathrm{MAS}}^{-1} f = \sum_l \sum_i f_{li}$, where $f_{li} = A_{V_{li}}^{-1} Q_{V_{li}} f$ are the solutions for the finite element problems

$$a(f_{li}, \phi_h) = a(P_{V_{li}} u, \phi_h) = f(\phi_h) \quad \forall \phi_h \in V_{li}. \tag{9}$$

It is easy to see that for f_{li} given by (9), equations (1) and (8) are equivalent. To find u_h, we first compute the right-hand side f_{MAS} by solving (9), and we then use the conjugate gradient (CG) method to solve the system (8). In each iteration, we need to compute $P_{V_{li}} v_h$ for a given $v_h \in V^h$ by solving the equation

$$a(P_{V_{li}} v_h, \phi_h) = a(v_h, \phi_h) \quad \forall \phi_h \in V_{li}.$$

This is a finite element equation on Ω_i^l with mesh size h_l, and $\dim(V_{li}) = 4$. Recall that for uniform triangulations, $V_{li}^{\alpha} \perp V_{li}^{\beta}$, for $\alpha \neq \beta$, and the local stiffness matrices $\{a(\phi_{li}^{\alpha}, \phi_{li}^{\beta})\}$ are diagonal, we have

$$P_{V_{li}} u = \sum_{\alpha} P_{V_{li}^{\alpha}} u = \sum_{\alpha} \frac{a(u, \phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha}, \qquad f_{li} = \sum_{\alpha} \frac{f(\phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha}.$$

If we assume that $h_1 = O(1)$, and we can replace P_{V_1} by $\sum_{i,\alpha} P_{V_{1i}^{\alpha}}$, then Algorithm 5.1 can be written explicitly as

$$\sum_{l=1}^{L} \sum_{i \in \mathcal{N}_l} \sum_{\alpha} \frac{a(u, \phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha} = \sum_{l} \sum_{i \in \mathcal{N}_l} \sum_{\alpha} \frac{f(\phi_{li}^{\alpha})}{a(\phi_{li}^{\alpha}, \phi_{li}^{\alpha})} \phi_{li}^{\alpha}.$$

K_l is the stiffness matrix associated with V_l and D_l is the diagonal (block) part of K_l. Let Π^l be the prolongation operator (matrix representation of the interpolation operator from V_l to V_L), and let Π_l^t be the adjoint of Π_l. Equation (8) can then be written as a preconditioned linear system

$$B_L^{-1} K_L x = B_L^{-1} b,$$

where

$$B_L^{-1} = \Pi_1 K_1^{-1} \Pi_1^t + \Pi_2 D_2^{-1} \Pi_2^t + \cdots + \Pi_{L-1} D_{L-1}^{-1} \Pi_{L-1}^t + D_L^{-1}.$$

K_1^{-1} can be replaced by any good preconditioner B_1 of K_1; in particular, if $h_1 = O(1)$, then K_1 can be replaced by D_1.

Remark 5.1. For domains with complicated geometry, the use of the diagonal D_1^{-1} to replace K_1^{-1} is not recommended, since $\kappa(D_1^{-1} K_1)$ can be large.

For elements with relatively larger aspect ratio, the 4-by-4 local stiffness matrix $K_{li} = \{a(\phi_{li}^{\alpha}, \phi_{li}^{\beta})\}$ or $\mathrm{diag}(K_{li})^{-1} K_{li}$ may have large condition numbers. Therefore, we should not replace V_{li} by the sum of V_{li}^{α} in the space decomposition, and the local 4-by-4 problem $K_{li} x = b$ should be solved exactly. Replacing K_{li} by its diagonal can hurt the overall performance of the algorithm.

5.2. Condition number estimate. It is easy to establish that the MAS operator P_{MAS} satisfies

$$C_1 L^{-1} a(u,u) \le a(P_{\text{MAS}}u, u) \le C_2 L a(u,u). \tag{10}$$

Thus, $\kappa(P_{\text{MAS}}) \le CL^2$. For second-order problems, $\kappa(P_{\text{MAS}})$ can be bounded uniformly with respect to the number of levels; cf. [18], [26]. We now eliminate L from (10).

We first establish a "strengthened Cauchy–Schwarz" inequality.

LEMMA 5.1. *For $u_l \in V_l$ and $u_k \in V_k$, let*

$$u_l = \sum_{i\in\mathcal{N}^l}\sum_{\alpha} u_{li}^{\alpha}, \quad u_{li}^{\alpha} \in V_{li}^{\alpha}, \quad \text{and} \quad u_k = \sum_{j\in\mathcal{N}^k}\sum_{\alpha} u_{kj}^{\alpha}, \quad u_{kj}^{\alpha} \in V_{kj}^{\alpha}.$$

Then

$$\begin{aligned} a(u_l,u_k) &\le C\sqrt{\gamma}^{|k-l|}(h_l^{-2}\|u_l\|_{L^2})(h_k^{-2}\|u_k\|_{L^2}) \\ &\le C_{cs}\sqrt{\gamma}^{|k-l|}\left(\sum_{i\in\mathcal{N}^l}\sum_{\alpha}|u_{li}^{\alpha}|_{H^2}^2\right)^{1/2}\left(\sum_{i\in\mathcal{N}^k}\sum_{\alpha}|u_{ki}^{\alpha}|_{H^2}^2\right)^{1/2}. \end{aligned}$$

Proof. We assume $k > l$. For each element τ^l, we decompose u_k as (cf. Fig. 3)

$$u_k = u_{\text{out}} + u_B + u_{\text{in}} \overset{\text{def}}{=} \sum_{j\notin\mathcal{N}^k(\overline{\tau}^l)}\sum_{\alpha\in\mathcal{D}_2} u_{kj}^{\alpha} + \sum_{j\in\mathcal{N}^k(\partial\tau^l)}\sum_{\alpha\in\mathcal{D}_2} u_{kj}^{\alpha} + \sum_{j\in\mathcal{N}^k(\tau^l)}\sum_{\alpha\in\mathcal{D}_2} u_{kj}^{\alpha}.$$

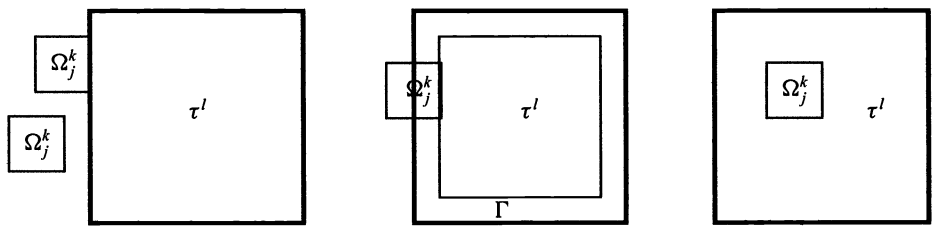


FIG. 3. *Three possible relations between Ω_j^k and τ^l.*

It is clear that, $a_{\tau^l}(u_l, u_{\text{out}}) = 0$, thus

$$a_{\tau^l}(u_l,u_k) = a_{\tau^l}(u_l,u_{in}) + a_{\tau^l}(u_l,u_B).$$

We estimate each term separately. Let $\Gamma = \tau^l \cap (\cup_{i\in\mathcal{N}^k(\partial\tau_k^l)}\Omega_i^k)$ be the union of smaller elements $\tau^k \in \mathcal{T}^k$ that touch the boundary of the larger element τ^l. We first estimate $a_{\tau^l}(u_l,u_B)$. Since $\text{supp}(u_B) = \Gamma$, we have

$$|a_{\tau^l}(u_l,u_B)| = |a_\Gamma(u_l,u_B)| \le |u_l|_{H^2(\Gamma)}|u_B|_{H^2(\Gamma)}.$$

Using inverse inequality, $|u_l|_{W^{2,\infty}(\tau^l)} \le Ch_l^{-1}|u_l|_{H^2(\tau^l)} \le Ch_l^{-3}\|u_l\|_{L^2(\tau^l)}$, we obtain

$$\begin{aligned} |u_l|_{H^2(\Gamma)} &\le |\Gamma|^{1/2}|u_l|_{W^{2,\infty}(\Gamma)} \le |\Gamma|^{1/2}|u_l|_{W^{2,\infty}(\tau^l)} \\ &\le C|\Gamma|^{1/2}h_l^{-3}\|u_l\|_{L^2(\tau^l)} \le C\sqrt{\gamma}^{k-l}h_l^{-2}\|u_l\|_{L^2(\tau^l)} \end{aligned}$$

and

$$|u_B|_{H^2(\Gamma)} \le Ch_k^{-2}\|u_B\|_{L^2(\Gamma)} \le Ch_k^{-2}\|u_k\|_{L^2(\tau^l)}.$$

Combining the above inequalities, we obtain

$$|a_{\tau^l}(u_l, u_B)| \le C\sqrt{\gamma}^{k-l}h_l^{-2}\|u_l\|_{L^2(\tau^l)}\, h_k^{-2}\|u_k\|_{L^2(\tau^l)}. \tag{11}$$

We now estimate $a_{\tau^l}(u_l, u_{in})$, integrate by parts, then use Schwarz inequality,

$$|a_{\tau^l}(u_l, u_{in})| = |(\Delta^2 u_l, u_{in})_{L_2(\tau^l)}| \le \|\Delta^2 u_l\|_{L^2(\tau^l)}\|u_{in}\|_{L^2(\tau^l)}.$$

It follows from the inverse inequality $|u_l|_{H^4(\tau^l)} \le h_l^{-4}\|u_l\|_{L^2(\tau^l)}$ that

$$\|\Delta^2 u_l\|_{L^2(\tau^l)} \le C|u_l|_{H^4(\tau^l)} \le Ch_l^{-4}\|u_l\|_{L^2(\tau^l)}.$$

Note that $\|u_{in}\|_{L^2(\tau^l)} \le C\|u_k\|_{L^2(\tau^l)}$, thus,

$$\begin{aligned} |a_{\tau^l}(u_l, u_{in})| &\le Ch_l^{-4}\|u_l\|_{L^2(\tau^l)}\|u_k\|_{L^2(\tau^l)} \\ &\le C\gamma^{2(k-l)}(h_l^{-2}\|u_l\|_{L^2(\tau^l)})\left(h_k^{-2}\|u_k\|_{L^2(\tau^l)}\right). \end{aligned} \tag{12}$$

Combining estimates (11) and (12), we obtain

$$|a_{\tau^l}(u_l, u_k)| \le C\sqrt{\gamma}^{k-l}\left(h_l^{-2}\|u_l\|_{L^2(\tau^l)}\right)\left(h_k^{-2}\|u_k\|_{L^2(\tau^l)}\right).$$

Summing over τ^l and using the Cauchy inequality, we obtain

$$\begin{aligned} a(u_l, u_k) &\le C\sqrt{\gamma}^{k-l}(h_l^{-2}\|u_l\|_{L^2(\Omega)})(h_k^{-2}\|u_k\|_{L^2(\Omega)}) \\ &\le C\sqrt{\gamma}^{k-l}h_l^{-2}\left(\sum_{i\in\mathcal{N}^l}\sum_{\alpha\in\mathcal{D}_2}\|u_{li}^\alpha\|_{L^2(\Omega)}^2\right)^{1/2} h_k^{-2}\left(\sum_{j\in\mathcal{N}^k}\sum_{\alpha\in\mathcal{D}_2}\|u_{kj}^\alpha\|_{L^2(\Omega)}^2\right)^{1/2}. \end{aligned}$$

The second inequality in the lemma follows from the fact that $\|u_{li}^\alpha\|_{L^2} \le h_l^2|u_{li}|_{H^2}$. □

The L^2 projection $Q_{V_h} : H^2(\Omega) \to V_h$ is defined by

$$(Q_{V_h}u, \phi)_{L^2(\Omega)} = (u, \phi)_{L^2(\Omega)} \quad \forall \phi \in V_h.$$

LEMMA 5.2. *We have the following estimates for* Q_{V_h}:

$$\begin{aligned} |Q_{V_h}u - u|_{H^s(\Omega)} &\le Ch^{2-s}|u|_{H^2(\Omega)}, \qquad s = 0, 1, 2, \\ |Q_{V_h}u|_{H^\sigma(\Omega)} &\le C|u|_{H^\sigma(\Omega)}, \qquad 0 \le \sigma \le 2. \end{aligned}$$

Proof. We construct a quasi-interpolation operator $\tilde{\Pi}_{V_h} : H_0^2(\Omega) \to V_h$ by

$$\tilde{\Pi}_{V_h}u = \sum_i u(x_i)\phi_i^{(0,0)} + \sum_i\sum_{\alpha\in\mathcal{D}_1}\{D^\alpha u\}_{\Omega_i}\phi_i^\alpha.$$

Here $\Omega_i = \operatorname{supp}\{\phi_i\}$ and $\{D^\alpha u\}_{\Omega_i} = (\int_{\Omega_i} D^\alpha u/|\Omega_i|)$. It can be shown, cf. Theorems 4.1.1 and 4.1.2 in [24], that

$$|\tilde{\Pi}_{V_h}u - u|_{H^s(\Omega)} \le Ch^{2-s}|u|_{H^2(\Omega)}. \tag{13}$$

By an inverse inequality and a property of Q_{V_h}, it follows that

$$
\begin{aligned}
|(Q_{V_h} - \tilde{\Pi}_{V_h})u|_{H^s(\Omega)} &\le Ch^{-s}\|(Q_{V_h} - \tilde{\Pi}_{V_h})u\|_{L^2(\Omega)} \\
&\le Ch^{-s}\|Q_{V_h}u - u\|_{L^2(\Omega)} + Ch^{-s}\|\tilde{\Pi}_{V_h}u - u\|_{L^2(\Omega)} \\
&\le 2Ch^{-s}\|\tilde{\Pi}_{V_h}u - u\|_{L^2(\Omega)} \le Ch^{2-s}|u|_{H^2(\Omega)}.
\end{aligned} \tag{14}
$$

The first part of the lemma follows from the inequalities (13), (14), and the triangle inequality. Taking $s = 2$ in the first inequality of the lemma and using the triangle inequality, we obtain

$$|Q_{V_h}u|_{H^2(\Omega)} \le |Q_{V_h}u - u|_{H^2(\Omega)} + |u|_{H^2(\Omega)} \le C|u|_{H^2(\Omega)}.$$

Note that $\|Q_{V_h}u\|_{L^2(\Omega)} \le \|u\|_{L^2(\Omega)}$. The second part of the lemma now follows from interpolation in Sobolev spaces. □

We will make the following regularity assumption.

Assumption 5.1. There exists a constant $\sigma > 0$ such that

$$|u|_{H^{2+\sigma}(\Omega)} \le C|\Delta^2 u|_{H^{-2+\sigma}(\Omega)}.$$

It is known that the assumption holds with $\sigma = 1$ for convex region. Blum and Rannacher [2] have shown that the assumption holds with $\sigma = 2$ for convex polygons with interior angles smaller than 126.28°.

Using the Aubin–Nitsche trick, it follows from Assumption 5.1 that the H^2-projection $P_{V_k} : H^2(\Omega) \to V_k$ satisfies

$$|(I - P_{V_k})u|_{H^{2-\sigma}(\Omega)} \le Ch_k^{\sigma}|(I - P_{V_k})u|_{H^2(\Omega)}. \tag{15}$$

LEMMA 5.3. *Assume that Assumption* 5.1 *holds. Let $Q_{V_l} : V_L \to V_l$ be the L^2 projection and let $u_k = (P_{V_k} - P_{V_{k-1}})u$. Then $Q_{V_l}u_k = 0$ for $l \ge k$ and*

$$|Q_{V_l}u_k|_{H^2(\Omega)} \le C\left(\frac{h_k}{h_l}\right)^{\sigma}|u_k|_{H^2(\Omega)} \le C_Q\gamma^{\sigma(k-l)}|u_k|_{H^2(\Omega)}, \qquad l \le k.$$

Proof. By an inverse inequality, the boundness of Q_{V_l} in the $H^{2-\sigma}$–norm, we obtain,

$$|Q_{V_l}u_k|_{H^2(\Omega)} \le Ch_l^{-\sigma}|Q_{V_l}u_k|_{H^{2-\sigma}(\Omega)} \le Ch_l^{-\sigma}|u_k|_{H^{2-\sigma}(\Omega)}.$$

The lemma follows by applying inequality (15) to the above inequality. □

We can now prove the following lemma.

LEMMA 5.4. *Assume that Assumption* 5.1 *holds. Then,*

$$\sum_{l=1}^{L}|(Q_{V_l} - Q_{V_{l-1}})u|^2_{H^2(\Omega)} \le C|u|^2_{H^2(\Omega)} \quad \forall u \in V_L.$$

The constant C is independent of L.

Proof. Let $u_k = (P_{V_k} - P_{V_{k-1}})u$. Then $u = \sum u_k$. Lemma 5.3 implies that $(Q_{V_l} - Q_{V_{l-1}})u_k = 0$, $k < l$ and

$$\sum_{l=1}^{L}|(Q_{V_l} - Q_{V_{l-1}})u|^2_{H^2} = \sum_{l=1}^{L}\left|(Q_{V_l} - Q_{V_{l-1}})\sum_{k=1}^{L}u_k\right|^2_{H^2}$$

$$
\begin{aligned}
&= \sum_{l=1}^{L} \left| \sum_{k=l}^{L} (Q_{V_l} - Q_{V_{l-1}}) u_k \right|_{H^2}^2 \\
&\le \sum_{l=1}^{L} \left\{ \sum_{k=l}^{L} |(Q_{V_l} - Q_{V_{l-1}}) u_k|_{H^2} \right\}^2 \\
&\le C_Q^2 \sum_{l=1}^{L} \left\{ \sum_{k=l}^{L} \gamma^{\sigma(k-l)} |u_k|_{H^2} \right\}^2 \\
&\le C_Q^2 C(\gamma, L) \sum_{k=1}^{L} |u_k|_{H^2}^2 = C_Q^2 C(\gamma, L) |u|_{H^2}^2 \le C_Q^2 C(\gamma) |u|_{H^2}^2.
\end{aligned}
$$

Here $C(\gamma, L) = (1 - \gamma^{\sigma L})^2/(1 - \gamma^{\sigma})^2$ and $C(\gamma) = 1/(1 - \gamma^{\sigma})^2$. □

THEOREM 5.1. *Assume that Lemma* 5.4 *holds. Then the* MAS *operator* P_{MAS} *satisfies*

$$C_1 a(u_h, u_h) \le a(P_{\text{MAS}} u_h, u_h) \le C_2 a(u_h, u_h) \quad \forall u_h \in V^h.$$

The constants are independent of $\{h_l\}$ *and* L.

Proof. Upper bound: Let $u \in V_L$ and let $\{u_{li}^{\alpha}\}$ be a decomposition of u, i.e.,

$$\sum_{l=1}^{L} \sum_{i \in \mathcal{N}^l} \sum_{\alpha} u_{li}^{\alpha} = u, \qquad u_{li}^{\alpha} \in V_{li}^{\alpha}.$$

Let $u_l = \sum_{i \in \mathcal{N}^l} \sum_{\alpha \in \mathcal{D}_2} u_{li}^{\alpha} \in V_{li}^{\alpha}$. It follows from Lemma 5.1 that

$$
\begin{aligned}
a(u, u) &= \sum_{l,k=1}^{L} a(u_l, u_k) \\
&\le C_{cs} \sum_{l,k=1}^{L} \sqrt{\gamma}^{|k-l|} \left(\sum_{i \in \mathcal{N}^l} \sum_{\alpha} |u_{li}^{\alpha}|_{H^2}^2 \right)^{1/2} \left(\sum_{j \in \mathcal{N}^k} \sum_{\alpha} |u_{kj}^{\alpha}|_{H^2}^2 \right)^{1/2} \\
&\le C_{cs} \frac{1 + \sqrt{\gamma}}{1 - \sqrt{\gamma}} \sum_{l=1}^{L} \sum_{i \in \mathcal{N}^l} \sum_{\alpha} |u_{li}^{\alpha}|_{H^2}^2.
\end{aligned}
$$

The upper bound follows from Lemma 3.1 and Remark 3.1.

Lower bound: Let $u_l = (Q_{V_l} - Q_{V_{l-1}})u$ and we further decompose u_l as

$$u_l = \sum_{i \in \mathcal{N}^l} \sum_{\alpha} u_{li}^{\alpha}, \qquad u_{li}^{\alpha} \in V_{li}^{\alpha}.$$

Using the inverse inequality $|u_{li}^{\alpha}|_{H^2(\Omega)}^2 \le C h_l^{-4} \|u_{li}^{\alpha}\|_{L^2(\Omega)}^2$, and Lemma 5.2, we obtain

$$\sum_{i \in \mathcal{N}^l} \sum_{\alpha} |u_{li}^{\alpha}|_{H^2(\Omega)}^2 \le C h_l^{-4} \sum_{i \in \mathcal{N}^l} \sum_{\alpha} \|u_{li}^{\alpha}\|_{L^2}^2 \le C h_l^{-4} \|u_l\|_{L^2}^2 \le C |u_l|_{H^2(\Omega)}^2.$$

Summing over l and using Lemma 5.4, we obtain

$$\sum_{l=1}^{L} \sum_{i \in \mathcal{N}^l} \sum_{\alpha} |u_{li}^{\alpha}|_{H^2(\Omega)}^2 \le C \sum_{l=1}^{L} |u_l|_{H^2(\Omega)}^2 \le C C(\gamma) |u|_{H^2(\Omega)}^2.$$

The lower bound now follows from Lemma 3.1. □

Remark 5.2. It is clear that

$$a(P_{\mathrm{HB}}u, u) \le a(P_{\mathrm{MAS}}u, u) \quad \text{and} \quad a(P_{\mathrm{MHB}}u, u) \le Ca(P_{\mathrm{MAS}}u, u).$$

Therefore, we have also obtained upper bounds for P_{HB} and P_{MHB}.

Remark 5.3. Recently, Oswald [19] also proved a similar result for the MAS algorithm. A discrete norm $|u|_O$ is defined by

$$|u|_O^2 = \inf_{\sum_l u_l = u} \sum_{l=1}^{L} h_l^{-4} \|u_l\|_{L^2}^2.$$

Oswald's approach is to use the Besov space theory to establish the following norm equivalency:

$$C_1|u|_O^2 \le |u|_{H^2(\Omega)}^2 \le C_2|u|_O^2. \tag{16}$$

Our proof for the lower bound follows an idea of Bramble and Pasciak [4]; see also Xu [22]. A more elementary approach for the lower bound is used in Zhang [26]: We first establish the result for domains with H^4-regularity. In the general case, we extend Ω to a larger convex domain $\tilde{\Omega}$, on which H^4-regularity holds. We then construct a decomposition of u with respect to $V^l(\Omega)$ from a decomposition of u with respect to $V^l(\tilde{\Omega})$. See [26] for a detailed discussion of the second-order case.

6. Multiplicative variants. In this section, we discuss some multiplicative variant of the multilevel Schwarz methods. They represent certain V-cycle multigrid methods.

Let $K^{ll} = \{a(\phi_{li}^{\alpha}, \phi_{lj}^{\beta})\}_{\alpha,\beta\in\mathcal{D}_2, i,j\in\mathcal{N}^l}$ be the stiffness matrix associated with triangulation $\mathcal{T}^l$. Let $K^{ll} = (D_l - L_l - L_l^t)$, where D_l, $-L_l$ are the diagonal and lower triangular parts of K^{ll}, respectively. Consider the problem

$$Au = f \quad \text{or} \quad K^{LL}x = b.$$

ALGORITHM 6.1 (MMS/MG-JACOBI). Let $u^{\mathrm{old}} \in V_L$ or x^{old} be the current approximation to the solution. Compute the new approximation by the following:

(1) Set $u_0 = u^{\mathrm{old}}$.

(2) For $l = 1, 2, \ldots, L$ compute

$$u_l = u_{l-1} + \eta(\textstyle\sum_{i,\alpha} A_{V_{li}^{\alpha}}^{-1} Q_{V_{li}^{\alpha}})(f - Au_{l-1}) = u_{l-1} + \eta(\textstyle\sum_{i,\alpha} P_{V_{li}^{\alpha}})(u - u_l),$$

where η is a damping factor such that $\|T_l\|_a \equiv \|\eta \sum_{i,\alpha} P_{V_{li}}^{\alpha}\|_a \le \omega < 2$.

(3) Set $u^{\mathrm{new}} = u_L$.

We order (α, i), $i \in \mathcal{N}^l$, $\alpha \in \mathcal{D}_2$, and represent it by one index i, $i = 1, \ldots, 4N_l$. Let $e_{li} = u - u_{li}$, $r_{li} = f - Au_{li} = Ae_{li}$.

ALGORITHM 6.2 (MMS/MG-SOR). Let $u^{\mathrm{old}} \in V_L$ be the current approximation.

(1) Set $u_{1,0} = u^{\mathrm{old}}$.

(2) For $\ell = 1, 2, \ldots, L$, compute

$u_{l,0} = u_{l-1,4N_{l-1}}$: the approximation of previous level (level ℓ-1).

For $i = 1, \ldots, 4N_l$ compute u_{li} by

$u_{li} = u_{l,i-1} + \omega A_{V_{li}}^{-1} Q_{V_{li}}(f - Au_{l,i-1}) = u_{l,i-1} + \omega P_{V_{li}} e_{l,i-1}$.

(3) Set $u^{\mathrm{new}} = u_{L,4N_L}$.

The case $\omega = 1$ corresponds to a Gauss–Seidel smoother and $1 < \omega < 2$ corresponds to a successive overrelaxation (SOR) smoother.

In matrix form, the algorithms can be written as

$$\begin{aligned} x_0 &= x^{\text{old}}, \\ x_l &= x_{l-1} + \Pi_l M_l^{-1} \Pi_l^t (b - K^{LL} x_{l-1}), \quad l = 1, \ldots, L, \\ x^{\text{new}} &= x_L, \end{aligned}$$

where $M_l^{-1} = \eta D_l^{-1}$ for weighted Jacobi smoother and $M_l^{-1} = (\frac{1}{\omega} D_l - L_l)^{-1}$ for a Gauss–Seidel/SOR smoother.

The corresponding error propagation operators are

$$E_J = \prod_{l=1}^{L} (I - T_l) \stackrel{\text{def}}{=} \prod_{l=1}^{L} \left(I - \eta \sum_{i \in N^l} \sum_{\alpha \in \mathcal{D}_2} P_{V_{li}^\alpha} \right), \tag{17}$$

$$E_\omega = \prod_{l=1}^{L} \prod_{i \in \mathcal{N}^l} \prod_{\alpha \in \mathcal{D}_2} (I - \omega P_{V_{li}^\alpha}). \tag{18}$$

The product in the above expressions can be arranged in any order; different orders correspond to different schemes. The operators E_J and E_ω correspond to the error propagation operators of V-cycle multigrid methods using damped Jacobi and Gauss–Seidel/SOR as smoothers, respectively.

Let ϕ_{li}^α be normalized basis functions, i.e., $|\phi_{li}^\alpha|_a = 1$. Then $\cos(V_{li}^\alpha, V_{kj}^\beta) = |a(\phi_{li}^\alpha, \phi_{kj}^\beta)|$. Let $K = \{a(\phi_{li}^\alpha, \phi_{kj}^\beta)\}$ and $\Theta = \{\cos(V_{li}^\alpha, V_{kj}^\beta)\}$. Then $\Theta = |K|$. The following lemma provides an estimate for $\|K\|_2$ and $\|\Theta\|_2$.

LEMMA 6.1. *There exists a constant $C > 0$ independent of L and h_l such that*

$$\|K\|_2 \le \|\Theta\|_2 \le C.$$

Proof. For simplicity, we assume that the triangulations are quasi uniform, thus $\gamma = \frac{1}{2}$. Let $\Omega_i^l = \text{supp}(\phi_{li}^\alpha) = \cup_{\tau^l \ni x_i^l} \tau^l$. We assume $l \le k$. There are three cases (cf. Fig 4).

(1) If $\Omega_i^l \cap \Omega_j^k = \emptyset$, then $\cos(V_{li}^\alpha, V_{kj}^\beta) = |a(\phi_{li}^\alpha, \phi_{kj}^\beta)| = 0$.

(2) If there exists an element $\tau^l \subset \Omega_i^l$ such that $\Omega_j^k \subset \tau^l \subset \Omega_i^l$, then

$$\begin{aligned} \cos(V_{li}^\alpha, V_{kj}^\beta) &= |a(\phi_{li}^\alpha, \phi_{kj}^\beta)| = |a_{\Omega_j^k}(\phi_{li}^\alpha, \phi_{kj}^\beta)| = |(\Delta^2 \phi_{li}^\alpha, \phi_{kj}^\beta)_{L^2(\Omega_j^k)}| \\ &\le C |\Delta^2 \phi_{li}^\alpha|_{L^2(\Omega_j^k)} |\phi_{kj}^\beta|_{L^2(\Omega_j^k)} \\ &\le C h_l^{-2} |\phi_{li}^\alpha|_{H^2(\Omega_j^k)} h_k^2 |\phi_{kj}^\beta|_{H^2(\Omega_j^k)} \\ &\le C 4^{-(k-l)} |\phi_{li}^\alpha|_{H^2(\Omega_j^k)} |\phi_{kj}^\beta|_{H^2(\Omega_j^k)} = C 4^{-(k-l)}. \end{aligned}$$

For fixed i, the number of Ω_j^k's that belong to this category is bounded by the number of Ω_j^k that intersect Ω_i^l. This number is bounded by

$$O\left(\frac{mes(\Omega_i^l)}{mes(\Omega_j^k)} \right) = O\left(\frac{h_l^2}{h_k^2} \right) = O(4^{k-l}).$$

(3) If i, j belong to neither the first case nor the second, then Ω_j^k must intersect some of the edges in $\mathcal{E}^l(\Omega_i^l)$, where $\mathcal{E}^l(\Omega_i^l)$ is the set of level-ℓ edges of Ω_i^l as defined in §2.3. For fixed

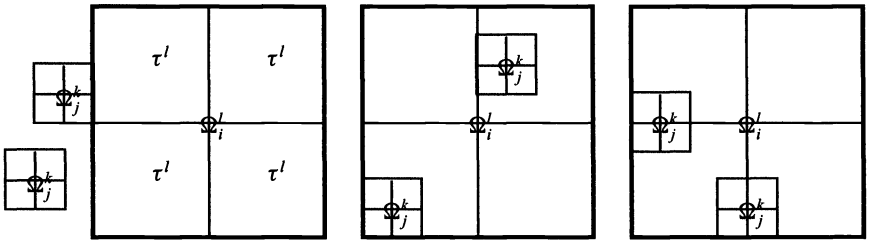


FIG. 4. *Three possible relations between* Ω_j^k *and* Ω_i^l.

i, the number of Ω_j^k that intersects some edges in $\mathcal{E}^l(\Omega_i^l)$ is bounded by $O(h_l/h_k) = O(2^{k-l})$. Note that $a(\phi_{kj}^\beta, \phi_{kj}^\beta) = 1$, and we have

$$\begin{aligned}
\cos(V_{li}^\alpha, V_{kj}^\beta) &= |a(\phi_{li}^\alpha, \phi_{kj}^\beta)| = |a_{\Omega_i^l \cap \Omega_j^k}(\phi_{li}^\alpha, \phi_{kj}^\beta)| \\
&\le |a_{\Omega_i^l \cap \Omega_j^k}(\phi_{li}^\alpha, \phi_{li}^\alpha)|^{1/2} |a(\phi_{kj}^\beta, \phi_{kj}^\beta)|^{1/2} \\
&\le mes(\Omega_i^l \cap \Omega_j^k)^{1/2} |\phi_{li}^\alpha|_{W^{2,\infty}(\Omega_i^l \cap \Omega_j^k)} \le Ch_k |\phi_{li}^\alpha|_{W^{2,\infty}(\Omega_i^l)} \\
&\le Ch_k h_l^{-1} |\phi_{li}^\alpha|_{H^2(\Omega_i^l)} = Ch_k h_l^{-1} \le C2^{-(k-l)}.
\end{aligned}$$

We partition $\Theta = \{\cos(V_{li}^\alpha, V_{kj}^\beta)\}_{i \in \mathcal{N}^l, j \in \mathcal{N}^k, \alpha, \beta \in \mathcal{D}_2, 1 \le l,k \le L}$ into L-by-L blocks,

$$\Theta = \{\Theta^{lk}\}_{1 \le l,k \le L},$$

where Θ^{lk} are $4N_l$-by-$4N_k$ matrices whose elements are given by

$$\Theta^{lk} = \{\Theta^{lk}(i, j)\}_{i \in \mathcal{N}^l, j \in \mathcal{N}^k, \alpha, \beta \in \mathcal{D}_2} = \{\cos(V_{li}^\alpha, V_{kj}^\beta)\}_{i \in \mathcal{N}^l, j \in \mathcal{N}^k, \alpha, \beta \in \mathcal{D}_2}.$$

We claim that

$$\|\Theta^{lk}\|_2 \le C\sqrt{2}^{(-|l-k|)}. \tag{19}$$

To prove this, we assume $l \le k$ and note that for fixed j, the number of Ω_i^l that intersect Ω_j^k is bounded uniformly. Thus the number of nonzeros in each column is bounded uniformly, i.e.;

$$\sum_{i \in \mathcal{N}^l} \sum_{\alpha} \operatorname{sign}(|a(\phi_{li}^\alpha, \phi_{kj}^\beta)|) \le N_0,$$

where N_0 depends only on the coarse triangulation $\mathcal{T}^1$. By a simple norm analysis, we can show that

$$\|\Theta^{lk}\|_2^2 \le N_0 \max_{i,\alpha} \|\mathrm{row}_{i,\alpha}(\Theta^{lk})\|_2^2.$$

We see that $\|\mathrm{row}_{i,\alpha}(\Theta^{lk})\|_2^2$ can be estimated as

$$\begin{aligned}
\|\mathrm{row}_{i,\alpha}(\Theta^{lk})\|_2^2 &= \sum_{j \in \mathcal{N}^k} \sum_{\beta} a(\phi_{li}^\alpha, \phi_{kj}^\beta)^2 \\
&= \left(\sum_{j \in \text{case 1}} + \sum_{j \in \text{case 2}} + \sum_{j \in \text{case 3}} \right) \sum_{\beta} a(\phi_{li}^\alpha, \phi_{kj}^\beta)^2 \\
&\le 0 + C4^{k-l} 4^{-2(k-l)} + C2^{k-l} 2^{-2(k-l)} \le C2^{-(k-l)}.
\end{aligned}$$

Thus

$$\|\Theta^{lk}\|_2 \le \sqrt{N_0}\max_{i,\alpha}\|\mathrm{row}_{i,\alpha}(\Theta^{lk})\|_2 \le C\sqrt{2}^{-(k-l)}.$$

If we replace Θ^{lk} by $\|\Theta^{lk}\|_2$, we get an L-by-L matrix $\tilde{\Theta} = \{\|\Theta^{lk}\|_2\}_{1\le lk\le L}$. The second part of the lemma follows from

$$\|\Theta\|_2 \le \|\tilde{\Theta}\|_2 \le \|\tilde{\Theta}\|_1 \le C\frac{\sqrt{2}+1}{\sqrt{2}-1}.$$

The first part of the lemma is trivial. □

The following theorem provides energy norm estimates for E_J and E_ω.

THEOREM 6.1. *There exist constants $\delta_J < 1$ and $\delta_\omega < 1$ such that*

$$\|E_J\|_a \le \delta_J, \quad \|E_\omega\|_a \le \delta_\omega, \quad 0 < \omega < 2.$$

The constants are independent of number of levels and the order of the product.

Proof. We first estimate $E_\omega = \prod_l \prod_i \prod_\alpha (I - \omega P_{V_{li}^\alpha})$. In this case, $T_i = \omega P_{V_{ij}^\alpha}$ for some subspace V_{ij}^α. Thus $T = \sum T_i = \omega P_{\mathrm{MAS}}$ and $\Theta_T = \omega\Theta = \omega|K|$. By Theorem 5.1, we know that $\lambda_{\min}(T) = \omega\lambda_{\min}(P_{\mathrm{MAS}})$ is uniformly bounded from below. On the other hand, it follows from Lemma 6.1 that $\|\Theta_T\|_2^2$ is uniformly bounded above. The estimate for $\|E_\omega\|_a$ now follows from Theorem 3.1.

We now estimate $E_J = \prod_l (I - \eta\sum_{i,\alpha} P_{V_{li}^\alpha})$. In this case, $T_l = \eta\sum_{i,\alpha} P_{V_{li}^\alpha}$ and $T = \sum T_l = \eta P_{\mathrm{MAS}}$. Thus $\lambda_{\min}(T) = \eta\lambda_{\min}(P_{\mathrm{MAS}})$ is uniformly bounded from below. To estimate $\|\Theta_T\|_2$, we note that

$$\begin{aligned}
a(T_l u, T_k v) &= \eta^2 \sum_{i,\alpha}\sum_{j,\beta} a(P_{V_{li}^\alpha} u, P_{V_{kj}^\beta} v)\\
&\le \eta^2 \sum_{i,\alpha}\sum_{j,\beta} \cos(V_{li}^\alpha, V_{kj}^\beta)|P_{V_{li}^\alpha} u|_a |P_{V_{kj}^\beta} v|_a\\
&\le \eta\|\Theta^{l,k}\|_2 \left(\eta\sum_{i,\alpha}|P_{V_{li}^\alpha} u|_a^2\right)^{1/2}\left(\eta\sum_{j,\beta}|P_{V_{kj}^\beta} v|_a^2\right)^{1/2}\\
&= \eta\|\Theta^{l,k}\|_2 a(T_l u, u)^{1/2} a(T_k v, v)^{1/2}.
\end{aligned}$$

By the definition of $\theta_T^{l,k}$ (cf. (6)), we have

$$\theta_T^{l,k} \le \eta\|\Theta^{l,k}\|_2 \le C\sqrt{2}^{(-|k-l|)}.$$

Therefore,

$$\|\Theta_T\|_2 \le \|\Theta_T\|_1 = \max_l \sum_k \theta_T^{l,k} \le C\frac{\sqrt{2}+1}{\sqrt{2}-1}.$$

The estimate for $\|E_J\|_a$ follows from Theorem 3.1. □

Theorem 6.1 implies that the V-cycle multigrid method, using damped Jacobi, Gauss–Seidel or SOR as smoother, has a rate of convergence that is independent of the number of levels and the order of product.

Similar techniques can be used to establish the convergence rate for the multiplicative HB algorithms (HB multigrid). We can show that the energy norms of the error propagation operators of the multiplicative variants of Algorithms 4.1 and 4.2 are bounded by $1 - Ch^2$ and $1 - C|\log h|^{-2}$, respectively. See [1] for more details about the multiplicative variants of Algorithm 4.1 for second-order elliptic problems.

7. Numerical experiments. In this section, we report on some numerical experiments with additive multilevel methods for the biharmonic problem. For comparison, we have also carried out experiments for the CG method and the Jacobi CG, a preconditioned CG method using a diagonal matrix as a preconditioner. These experiments were carried out for the biharmonic equation on a unit square with homogeneous Dirichlet boundary conditions

$$\begin{aligned} -\Delta^2 u &= f(x) \quad \text{in} \quad \Omega, \\ u = \frac{\partial u}{\partial n} &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{20}$$

We divide the domain Ω into 2×2 square elements τ_i^1, $i = 1, \ldots, 4$, and obtain a triangulation $\mathcal{T}^1 = \{\tau_i^1\}$. We then divide each τ_i^1 into 2×2 squares to obtain the triangulation $\mathcal{T}^2 = \{\tau_i^2\}$, etc. The length of an edge of τ_i^l is denoted by h_l, where $h_l = (\frac{1}{2})^l$. In all our experiments, we use the bicubic element. Let P_{NB}, P_{HB}, P_{MHB}, and P_{MAS} denote the linear operator (or matrix) obtained from the conventional nodal basis, the HB, the MHB, and the MAS methods, respectively. We use the preconditioned CG method to solve the corresponding linear systems.

In Tables 1–5, we present numerical results for these methods. The first column contains the number of levels, L, of the multilevel methods, the second column contains the total degrees of freedom N $(= 4(2^L - 1)^2)$. In the third, fourth, and fifth columns, we give the minimum eigenvalues, the maximum eigenvalues, and the condition numbers of the corresponding algorithms. These numbers are computed by the Lanczos methods (as a by-product of the CG computation). The sixth column gives the number of iterations required to decrease the l_2-norm of the residual of the preconditioned system by a factor of $\epsilon = 10^{-6}$.

TABLE 1
CG method using bicubic elements.

Level	N	$\lambda_{\min}(P)$	$\lambda_{\max}(P)$	$\kappa(P)$	Iter.
1	4	0.447E+0	0.472E+2	0.106E+3	3
2	36	0.276E+0	0.798E+2	0.289E+3	24
3	196	0.208E+0	0.894E+2	0.428E+3	89
4	900	0.180E–1	0.918E+2	0.510E+4	203
5	3844	0.120E–2	0.925E+2	0.767E+5	518
6	15876	0.764E–4	0.926E+2	0.121E+7	1749
7	64516	0.457E–5	0.927E+2	0.203E+8	6473

TABLE 2
Jacobi conjugate gradient method using bicubic elements.

Level	N	$\lambda_{\min}(P)$	$\lambda_{\max}(P)$	$\kappa(P)$	Iter.
1	4	0.100E+1	0.100E+1	0.100E+1	1
2	36	0.853E–1	0.176E+1	0.206E+2	14
3	196	0.631E–2	0.191E+1	0.303E+3	41
4	900	0.412E–3	0.195E+1	0.473E+4	116
5	3844	0.260E–4	0.196E+1	0.753E+5	405
6	15876	0.162E–5	0.197E+1	0.121E+7	1154
7	64516	0.971E–7	0.197E+1	0.202E+8	3313

We have shown that the condition number $\kappa(P_{NB}) = O(h^{-4}) = O(2^{-4L}) = O(16^{-L})$; this is clearly verified by Table 1. For $L = 1$, we have only one interior vertex and thus four degrees of freedom. The matrix is diagonal and has only three distinct eigenvalues and the CG algorithm converges in three steps. Note that the basis functions are not well scaled, $|\phi_{l,i}^{\alpha}|_a \neq \text{const}$. By normalizing the basis function or by using a diagonal scaling technique, the condition number can be improved. This is shown in Table 2.

TABLE 3
HB method using bicubic elements.

Level	N	$\lambda_{\min}(P)$	$\lambda_{\max}(P)$	$\kappa(P)$	Iter.
1	4	0.100E+1	0.100E+1	0.100E+1	1
2	36	0.757E−1	0.166E+1	0.220E+2	14
3	196	0.177E−1	0.204E+1	0.115E+3	39
4	900	0.428E−2	0.241E+1	0.562E+3	80
5	3844	0.106E−2	0.271E+1	0.256E+4	165
6	15876	0.263E−3	0.295E+1	0.112E+5	302
7	64516	0.657E−4	0.314E+1	0.478E+5	604
8	260100	0.164E−4	0.330E+1	0.201E+6	1113

TABLE 4
MHB method using bicubic elements.

Level	N	$\lambda_{\min}(P)$	$\lambda_{\max}(P)$	$\kappa(P)$	Iter.
1	4	0.100E+1	0.200E+1	0.200E+1	2
2	36	0.273E+0	0.230E+1	0.843E+1	14
3	196	0.167E+0	0.246E+1	0.147E+2	23
4	900	0.110E+0	0.253E+1	0.229E+2	30
5	3844	0.781E−1	0.274E+1	0.350E+2	37
6	15876	0.580E−1	0.296E+1	0.510E+2	44
7	64516	0.452E−1	0.315E+1	0.696E+2	50
8	260100	0.362E−1	0.330E+1	0.912E+2	55

TABLE 5
MAS method using bicubic elements.

Level	N	$\lambda_{\min}(P)$	$\lambda_{\max}(P)$	$\kappa(P)$	Iter.
1	4	0.100E+1	0.100E+1	0.100E+1	1
2	36	0.588E+0	0.188E+1	0.320E+1	10
3	196	0.471E+0	0.262E+1	0.557E+1	14
4	900	0.444E+0	0.327E+1	0.736E+1	17
5	3844	0.438E+0	0.379E+1	0.867E+1	18
6	15876	0.437E+0	0.422E+1	0.967E+1	19
7	64516	0.436E+0	0.458E+1	0.105E+2	20
8	260100	0.436E+0	0.486E+1	0.112E+2	20

In Table 3, we report on the results for the HB method. The theory suggest that the condition number $\kappa(P_{\mathrm{HB}}) = O(h^{-2}) = O(4^{-L})$. We see that the $\kappa(l+1)/\kappa(l) \approx 4$.

In Table 4, we report on the results for the MHB method. The condition number of P_{MHB} grows slowly as predicted by the theory.

Finally, in Table 5, we report on the results for the MAS method. The condition number of P_{MAS} is bounded (grows very slowly) as predicted by the theory. One can see that the smallest eigenvalues decrease very slowly and quickly approach a limiting value. This confirms our theory. As remarked in [23], the largest eigenvalues, although bounded in theory, increase as fast as they are allowed in the proof. In fact, for the first eight levels, the largest eigenvalues, thus the condition numbers, behave like $C\sqrt{L}$.

Acknowledgement. The author is indebted to Professor Olof B. Widlund for reading the manuscript and for providing valuable comments.

REFERENCES

[1] R. E. BANK, T. F. DUPONT AND H. YSERENTANT, *The hierarchical basis multigrid method*, Numer. Math., 52 (1988), pp. 427–458.

[2] H. BLUM AND R. RANNACHER, *On the boundary value problem of the biharmonic operator on domains with angular corners*, Math. Method Appl. Sci., (1980), pp. 556–581.

[3] 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.

[4] J. H. BRAMBLE AND J. E. PASCIAK, *New estimates for multilevel algorithms including the V-cycle*, Tech. report, Cornell University, Ithaca, NY, Aug. 1991.

[5] J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for product iterative methods with applications to domain decomposition*, Math. Comp., 57 (1991), pp. 1–21.

[6] J. H. BRAMBLE, J. E. PASCIAK, AND J. XU, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[7] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978.

[8] M. DRYJA AND O. B. WIDLUND, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. report 339; also, Ultracomputer Note 131, Dept. Computer Science, Courant Institute, New York University, 1987.

[9] ———, *Some domain decomposition algorithms for elliptic problems*, in Iterative Methods for Large Linear Systems, L. Hayes and D. Kincaid, eds., San Diego, CA, 1989, Academic Press, New York, pp. 273–291; Proc. Conf. Iterative Methods for Large Linear Systems, Austin, TX, Oct. 19–21, 1988, to celebrate the 65th birthday of David M. Young, Jr.

[10] ———, *Multilevel additive methods for elliptic finite element problems*, in Parallel Algorithms for Partial Differential Equations, Proc. 6th GAMM-Seminar, Kiel, Jan. 19–21, 1990, W. Hackbusch, ed., Vieweg & Son, Braunschweig, Germany, 1991.

[11] L. HART AND S. MCCORMICK, *Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors: Computational analysis*, Parallel Comput., 12 (1989), pp. 131–144.

[12] P. LAX AND A. MILGRAM, *Contributions to the Theory of Partial Differential Equations*, Princeton University Press, Princeton, NJ, Annals of Math. Studies 33, 1954, pp. 167–190.

[13] J. MANDEL AND S. MCCORMICK, *Iterative solution of elliptic equations with refinement: The two-level case*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[14] A. M. MATSOKIN AND S. V. NEPOMNYASCHIKH, *A Schwarz alternating method in a subspace*, Soviet Math., 29 (1985), pp. 78–84.

[15] S. MCCORMICK, *Fast adaptive composite grid* (FAC) *methods*, in Defect Correction Methods: Theory and Applications, K. Böhmer and H. J. Stetter, eds., Computing Supplementum 5, Springer-Verlag, Wien, 1984, pp. 115–121.

[16] S. MCCORMICK AND J. THOMAS, *The fast adaptive composite grid* (FAC) *method for elliptic equations*, Math. Comp., 46 (1986), pp. 439–456.

[17] S. V. NEPOMNYASCHIKH, *On the application of the method of bordering for elliptic mixed boundary value problems and on the difference norms of* $W_2^{1/2}(S)$, Tech. report 106, Computing Center of the Siberian Branch of the USSR Academy of Sciences, Novosibirsk, 1984. (In Russian.)

[18] P. OSWALD, *On discrete norm estimates related to multilevel preconditioners in the finite element method*, in Proc. Internat. Conf. Theory of Functions, Varna 91, 1991, Varna, Bulgaria, 1992.

[19] ———, *Preconditioners for discretizations of the biharmonic equation by rectangular finite elements*, Tech. report, Friedrich Schiller Universität, Jena, Germany, 1991.

[20] P. OSWALD, *Hierarchical conforming finite element methods for the biharmonic equation*, SIAM J. Numer. Anal., 29 (1992), pp. 1610–1625.

[21] J. XU, *Theory of Multilevel Methods*, Ph.D. thesis, Cornell University, Ithaca, NY, May 1989.

[22] ———, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

[23] H. YSERENTANT, *On the multi-level splitting of finite element spaces*, Numer. Math., 49 (1986), pp. 379–412.

[24] X. ZHANG, *Studies in Domain Decomposition: Multilevel Methods and the Biharmonic Dirichlet Problem*, Ph.D. thesis, Courant Institute, New York University, Sept. 1991.

[25] ———, *Domain decomposition algorithms for the biharmonic Dirichlet problem*, in 5th Conference on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. A. Meurant, J. S. Scroggs, and R. G. Voigt, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA 1992.

[26] ———, *Multilevel Schwarz methods*, Numer. Math., 63 (1992), pp. 521–539.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 645–667, May 1994

010

MULTILEVEL ALGORITHMS FOR CONSTRAINED COMPACT FIXED POINT PROBLEMS *

C. T. KELLEY† AND E. W. SACHS‡

Abstract. In this paper the authors extend the multilevel algorithm of Atkinson and Brakhage for compact fixed point problems and the projected Newton method of Bertsekas to create a fast multilevel algorithm for parabolic boundary control problems having bound constraints on the control. Results are extended from finite dimension on constraint identification. This approach permits both adaptive integration in time and inexact evaluation of the cost functional.

Key words. projected Newton method, constraint identification, collective compactness, parabolic optimal control problems

AMS subject classifications. 45G10, 47H17, 49K20, 49M15, 65H10, 65K10

1. Introduction.

In this paper, an expanded version of [15], we consider fast algorithms for solution of nonlinear equations that can be expressed in the form

$$u(t) = \mathcal{P}(\mathcal{K}(u))(t), \tag{1.1}$$

where $\mathcal{K}$ is a completely continuous map from $L^\infty(\Omega)$ to $C(\Omega)$ for some bounded $\Omega \subset R^n$, and $\mathcal{P}$ is the map on $C(\Omega)$ given by

$$\mathcal{P}(u)(t) = \begin{cases} u_{\min}(t) & \text{if } u(t) \le u_{\min}(t), \\ u(t) & \text{if } u_{\min}(t) \le u(t) \le u_{\max}(t), \\ u_{\max}(t) & \text{if } u(t) \ge u_{\max}(t), \end{cases} \tag{1.2}$$

for given $u_{\min}$ and $u_{\max}$ in $C(\Omega)$. The particular algorithm we consider is a generalization and synthesis of the Atkinson–Brakhage multilevel algorithm for compact fixed-point problems [3], [6], and the projected Newton method of Bertsekas [5] for bound constrained minimization problems.

A paradigm for problems of the form (1.1) is the Urysohn equation

$$\mathcal{K}(u)(t) = \int_\Omega k(t, s, u(s))\, ds. \tag{1.3}$$

Maps that are not easily expressible in this way, however, are the real target. In particular, we wish to develop an algorithm general enough to be applicable to boundary control problems for partial differential equations. The algorithms and assumptions in this paper provide fast local convergence for problems with continuous controls in one space dimension.

Our methods differ from previous multilevel approaches for such problems [10] in that the smoothing requirements on the approximate Fréchet derivatives at the various levels are relaxed, the results on identification of active indices from [5] can be extended, and the algorithm is a direct approximation of the projected Newton method and therefore quasi-Newton methods can be used to accelerate the convergence.

*Received by the editors May 6, 1992; accepted for publication (in revised form) December 21, 1992.

†Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, Box 8205, Raleigh, North Carolina 27695-8205 (Tim_Kelley@ncsu.edu). The research of this author was supported by Air Force Office of Scientific Research grant AFOSR-FQ8671-9101094, National Science Foundation grants DMS-9024622 and INT-8800560, North Atlantic Treaty Organization grant CRG 920067, and an allocation of computing resources from the North Carolina Supercomputing Center.

‡Universität Trier, FB IV – Mathematik, Postfach 3825, 5500 Trier, Federal Republic of Germany (sachs@msun4.uni-trier.de). The research of this author was supported by the Volkswagen-Stiftung.

We are motivated by constrained parabolic optimal control problems in one space dimension. One such problem, a constrained version of the problem considered in [16], is to minimize

$$f(u) = \frac{1}{2}\int_0^1 (y(u;T,x) - z(x))^2\,dx + \frac{\alpha}{2}\int_0^T u^2(t)\,dt, \tag{1.4}$$

where $\alpha > 0$ is given and $y(t,x) = y(u;t,x)$ is the solution to the nonlinear parabolic problem

$$\begin{aligned} &y_t(t,x) = y_{xx}(t,x), \quad 0 < x < 1, \quad 0 < t < T, \\ &y(0,x) = y_0(x), \quad 0 < x < 1, \\ &y_x(t,0) = 0, \quad y_x(t,1) = g(y(t,1)) + u(t), \quad 0 < t < T. \end{aligned} \tag{1.5}$$

In this problem u is allowed to vary over the set

$$\mathcal{U} = \{u \in L^\infty([0,T]) \mid u_{\min}(t) \le u(t) \le u_{\max}(t), \text{ for a.e. } t \in [0,T]\}, \tag{1.6}$$

and the nonlinear function g is assumed to satisfy

$$g \in C^2(R), \qquad g', g'' \in L^\infty(R). \tag{1.7}$$

Such problems arise in metallurgy, for example [21].

The gradient of f in $L^2([0,T])$ is

$$(\nabla f(u))(t) = \alpha u(t) + d(t,1), \tag{1.8}$$

where $d(t,x)$ is the solution of the adjoint problem

$$\begin{aligned} &-d_t(t,x) = d_{xx}(t,x), \quad 0 < x < 1, \quad 0 < t < T, \\ &d(T,x) = y(T,x) - z(x), \quad 0 < x < 1, \\ &d_x(t,0) = 0, \quad d_x(t,1) = g'(y(t,1))d(t,1), \quad 0 < t < T. \end{aligned} \tag{1.9}$$

We let $\mathcal{K}$ be the map that takes u into $-d(t,1)/\alpha$. It is known [20] that $\mathcal{K}$ is completely continuous from $L^\infty([0,T])$ to $C([0,T])$, (and hence a completely continuous map on $C([0,T])$), and in fact is a continuous map from $L^p([0,T])$ to $C([0,T])$ for $p > 2$. Standard techniques in optimization [4] imply that a necessary condition for u^* to be a solution of the problem given by (1.4), (1.5), and (1.6), is that

$$u^* = \mathcal{P}(\mathcal{K}(u^*)).$$

In [16] we considered the unconstrained problem with $\mathcal{U}$ replaced by $C([0,T])$ and used DASSL [7] to perform the integration in time. The use of such an adaptive time integrator required weaker smoothing assumptions than that used in [10] on the maps that take discrete versions of u into those of d. Under these relaxed smoothing assumptions we showed how the Atkinson–Brakhage algorithm could be implemented with appropriate finite difference gradients to obtain fast convergence.

The purpose of this paper is to merge the work in [16] with the ideas in [5] to design a fast algorithm for constrained optimal control problems of the type described above. We extend the projected Newton method to the abstract setting of constrained compact fixed point problems. We generalize the convergence results and the results on identification of the intervals on which $u(t)$ attains its bounds. This latter result is an extension of some of the results in [17].

Once the analysis is complete we can apply the ideas in [16] directly to the problem given by (1.4)–(1.6) and produce a fast algorithm. The algorithm here differs from that proposed in [16] in that numerical Jacobians are not computed on coarse grids. Instead, generalized minimal residual (GMRES) [19] iteration is used to solve the coarse mesh linearized problems needed by the Atkinson–Brakhage iteration and a projected form of the Newton-GMRES iteration [8] is used to solve the coarse mesh problem itself. This modification, suggested for the first time in [12], makes the analysis of the algorithm proposed here simpler than the one from [16]. We conclude the paper with a report on some numerical results for constrained optimal control problems.

In the remainder of this section we briefly describe the projected Newton iteration of Bertsekas and our proposed algorithm for (1.1). We do not discuss the line search used in [5] to ensure global convergence since the focus of this paper is fast algorithms for local convergence. We take the position that the method from [5] is sufficient to obtain convergence from distant initial iterates on coarse meshes and that the solution so obtained can be interpolated to provide a good initial iterate on finer meshes.

The problem considered in [5] is to minimize a function f defined on a box in R^N

$$\mathcal{U}_N = \{u \in R^N \mid u^i_{\min} \le u^i \le u^i_{\max},\ 1 \le i \le N\}.$$

Here u^i denotes the ith component of the vector $u \in R^N$ and $u_{\min}, u_{\max} \in R^N$ are given. If f is differentiable there is a solution u^* and each solution satisfies the necessary condition

$$u = \mathcal{P}(u - \nabla f(u)) \tag{1.10}$$

where, similarly to (1.2) for the continuous problem,

$$(\mathcal{P}u)^i = \begin{cases} u^i_{\min} & \text{if } u^i \le u^i_{\min}, \\ u^i & \text{if } u^i_{\min} \le u^i \le u^i_{\max}, \\ u^i_{\max} & \text{if } u^i \ge u^i_{\max}. \end{cases}$$

The second-order sufficient condition for optimality of u^* is that (1.10) holds and that the matrix H given by

$$H = P_A + P_I \nabla^2 f(u^*) P_I$$

be positive definite. Here P_A is the projection

$$(P_A u)^i = \begin{cases} u^i & \text{if } (u^*)^i = u^i_{\max} \text{ and } (\nabla f(u^*))^i < 0, \\ u^i & \text{if } (u^*)^i = u^i_{\min} \text{ and } (\nabla f(u^*))^i > 0, \\ 0 & \text{otherwise,} \end{cases}$$

and $P_I = I - P_A$.

The iteration proposed in [5] is of the form

$$u_{n+1} = \mathcal{P}(u_n - \alpha_n H_n^{-1} \nabla f(u_n)), \tag{1.11}$$

where

$$H_n = P_{A,n} + P_{I,n} \nabla^2 f(u_n) P_{I,n},$$

α_n is selected by an Armijo type rule, and $P_{A,n}$ and $P_{I,n}$ are approximations to P_A and P_I, respectively. The construction of $P_{A,n}$ and $P_{I,n}$ in [5] ensures that in the iteration $P_{A,n} = P_A$

and $\alpha_n = 1$ for n sufficiently large. After the active set has been identified (when $P_{A,n} = P_A$) the iteration reduces to Newton's method on the inactive set and therefore local quadratic convergence holds. Crucial to the analysis in [5] is identification of the active set after finitely many iterations. This allows one to reduce the analysis of the limiting behavior of the iteration to that for Newton's method.

For problems with a continuum of constraints, such as the ones under consideration in this paper, identification of the active set after finitely many iterations is unlikely. To use the ideas of [5] one must change the algorithm to take this into consideration and change the analysis as well. In §2 we introduce notation and discuss the assumptions needed to make the estimate

$$\|u - u^*\|_X \leq K\|u - \mathcal{P}(\mathcal{K}(u))\|_X$$

for u sufficiently near u^* and some constant K. This estimate is trivial in the finite dimensional case if f is sufficiently smooth, u is an element of the sequence of iterates, and the active set has already been identified. In the infinite dimensional case considered here attention must be paid to the size of sets in which activity of the constraints is unclear. In §3 we show how the algorithm in [5] can be extented to take the continuum of constraints into account and prove a local convergence result. The results in these sections require only a modest smoothing assumption. We show how the Atkinson–Brakhage algorithm can be used to create a fast algorithm and present some numerical results in §4. The design of the fast algorithm requires a stronger compactness assumption than the results in the earlier sections.

2. The basic estimate. In this section we state our assumptions on the nonlinearity and show that for continuous u the error in the solution is proportional to the size of the nonlinear residual in a neighborhood of the solution. We work in the space of continuous functions on Ω, $C = C(\Omega)$ and also on $X = L^\infty(\Omega)$, where $\Omega \subset R^d$. We assume that $\mathcal{K}$ is a smooth map on X and seek to solve the constrained compact fixed point problem

$$\mathcal{F}(u) = u - \mathcal{P}(\mathcal{K}(u)), \tag{2.1}$$

where $\mathcal{P}$ is given by (1.2). We note that since $\mathcal{K}$ maps L^p to $C = C(\Omega)$, the point evaluation implicit in $\mathcal{P}$ is defined and therefore $\mathcal{F}$ is a well-defined map on X. The spaces X and C are both given the sup norm, which we denote by $\|\cdot\|_X$, and C is a closed subspace of X.

In keeping with the application to optimal control we will denote points in Ω by t. For $t \in \Omega$ we let U be given by

$$U(t) = \{u \in X \mid u_{\min}(t) \leq u(t) \leq u_{\max}(t)\},$$

where $u_{\min}, u_{\max} \in C$, and define the two point set

$$\partial U(t) = \{u_{\min}(t), u_{\max}(t)\}.$$

If $S \subset \Omega$ we let $S^c = \Omega \setminus S$ be the complement of S in Ω. If S and T are subsets of Ω we denote the symmetric difference by

$$S \Delta T = (S \cup T) \setminus (S \cap T).$$

Throughout this paper χ_S will denote the characteristic function of the set S and μ will denote Lebesgue measure on R^d. We will let $\mathcal{L}(X, Y)$ be the Banach space of bounded linear maps from a Banach space X to a Banach space Y and let $\mathcal{COM}(X)$ be the space of compact linear maps on X.

2.1. Differentiability, smoothing, and nonsingularity assumptions. We make the following differentiability assumption on $\mathcal{K}$.

Assumption 2.1. There is a solution $u^* \in C$ to (2.1) and a neighborhood $\mathcal{N} \subset X$ of u^* in which the $\mathcal{K}$ is Lipschitz continuously Fréchet differentiable in X and in C.

For future use we let γ be the Lipschitz constant for the map $\mathcal{K}$ and γ_1 the Lipschitz constant for the maps $\mathcal{K}'$.

Our smoothing assumptions are given in the following statements.

Assumption 2.2. There is $p \in [1, \infty)$ such that the family of maps $\{\mathcal{K}'(u)\}$, where $u \in \mathcal{N}$ is a uniformly bounded subset of $\mathcal{L}(L^p(\Omega), C)$. Let

$$\sup_{u \in \mathcal{N}} \|\mathcal{K}'(u)\|_{\mathcal{L}(L^p(\Omega), C)} = M_K. \tag{2.2}$$

An immediate consequence of these assumptions is stated in Proposition 2.1.

PROPOSITION 2.1. *For all measurable $S, T \subset \Omega$ and $u \in \mathcal{N}$*

$$\|\mathcal{K}'(u)(\chi_S - \chi_T)\|_C \leq M_K \mu(S \Delta T)^{1/p}.$$

We define active and inactive sets for u^* by

$$A^* = \{t \mid u^*(t) \in \partial U(t)\} \quad \text{and} \quad I^* = \{t \mid u^*(t) \in \operatorname{int}(U)(t)\}. \tag{2.3}$$

In (2.3) $\operatorname{int}(U)(t)$ is defined to be the interval

$$\operatorname{int}(U)(t) = (u_{\min}(t), u_{\max}(t)).$$

For a measurable subset S of Ω define

$$\mathcal{K}_S(u) = \chi_S \mathcal{K}(u^* + \chi_S(u - u^*))$$

and

$$\mathcal{G}_S(u) = u - \mathcal{K}_S(u).$$

Clearly, we have the following proposition.

PROPOSITION 2.2. *For all measurable $I \subset \Omega$ $\mathcal{G}_I$ is Lipschitz continuously Fréchet differentiable as a map on X and L^p. If I is closed, $\mathcal{G}_I$ is Lipschitz continuously Fréchet differentiable as a map on $C(I)$. Moreover there is M_1, independent of I, such that*

$$\|\mathcal{G}_I'(u)\|_{\mathcal{L}(C(I))}, \text{ (if } I \text{ is closed)}, \|\mathcal{G}_I'(u)\|_{\mathcal{L}(X)}, \|\mathcal{G}_I'(u)\|_{\mathcal{L}(L^p)} \leq M_1 \tag{2.4}$$

for all $u \in \mathcal{N}$.

In Proposition 2.2, the action of $\mathcal{G}_I'(u)$ on $w \in C(I)$ is understood to be given by extension of w to zero on $[0, T] \setminus I$, application of $\mathcal{G}_I'(u)$ to that extension, and restriction to I.

The next assumption is needed to make the extension of the projected Newton method described in [5] converge quadratically.

Assumption 2.3. $\mathcal{G}_{I^*}' = I - \chi_{I^*}\mathcal{K}'(u^*)\chi_{I^*}$ is a nonsingular map on X and on L^p. There are η_1 and M_2 such that for every $u \in \mathcal{N}$ and every measurable $I \subset \Omega$ such that $\|u - u^*\|_X < \eta_1$ and $\mu(I \Delta I^*) < \eta_1$ the map $\mathcal{G}_I'(u)$ is a nonsingular map on C, on X, and on L^p, and

$$\|\mathcal{G}_I'(u)^{-1}\|_{\mathcal{L}(C)}, \|\mathcal{G}_I'(u)^{-1}\|_{\mathcal{L}(X)}, \|\mathcal{G}_I'(u)^{-1}\|_{\mathcal{L}(L^p)} \leq M_2. \tag{2.5}$$

The most simple analog of the iteration (1.11) from [5] is

$$u_{n+1} = \mathcal{P}(u_n - A_n \mathcal{F}(u_n)), \tag{2.6}$$

where A_n is an approximation of $\mathcal{G}_{I^*}'(u^*)^{-1}$. Construction of an extension of (1.11) that has good L^∞ convergence properties requires a more complex iteration than that given in (2.6).

Fundamental to the local convergence analysis of any Newton-like algorithm is a bound of the error in terms of the size of $\mathcal{F}$. The remainder of this section is devoted to such an estimate.

2.2. Assumptions and results on sets A^* and I^*. The next two lemmas are simple consequences of the Lipschitz continuity of $\mathcal{K}$ and the convexity of U. Before stating them we recall some more notation. If $S \subset R^k$ for some k and $t \in R^k$, we denote the distance from t to S by

$$\text{dist}(t, S) = \inf\{s \in S \mid \|t - s\|_{R^k}\}.$$

Note that

$$\mathcal{G}(u) = \mathcal{G}_\Omega(u) = u - \mathcal{K}(u).$$

For $u \in C$ define

$$\Omega_\delta(u) = \{t \mid |\mathcal{G}(u)(t)| \geq \delta\}, \quad \text{and} \quad \Omega_\delta^* = \Omega_\delta(u^*) = \{t \mid |\mathcal{G}(u^*)(t)| \geq \delta\}. \tag{2.7}$$

We require the following trivial lemma.

LEMMA 2.3. *$\Omega_\delta^* \subset A^*$ for all $\delta > 0$.*

Proof. Since $u^*(t) = \mathcal{P}(\mathcal{K}(u^*))(t)$ either $\mathcal{G}(u^*)(t) = 0$ or $u^*(t) = \mathcal{P}(\mathcal{K}(u^*))(t) \neq (\mathcal{K}(u^*))(t)$. Hence if $t \in \Omega_\delta^*$ then $u^*(t)$ is the image under $\mathcal{P}$ of a point outside of U and therefore $u^*(t) \in \partial U$. This means that $t \in A^*$ as asserted. □

Recall that we denote by γ the Lipschitz constant for $\mathcal{K}$ in the set $\mathcal{N}$; therefore, $\bar{\gamma} = 1 + \gamma$ is the Lipschitz constant for $\mathcal{G}$ in $\mathcal{N}$. We have the following lemma.

LEMMA 2.4. *Assume that Assumption* 2.1 *holds. There is $\sigma_0 > 0$ such that if $u \in C$ satisfies*

$$\|u - u^*\|_X < \sigma \leq \sigma_0, \tag{2.8}$$

and $\delta > \bar{\gamma}\sigma$, then

$$\Omega_\delta(u) \subset \Omega_{\delta - \bar{\gamma}\sigma}^* \subset A^*.$$

Proof. Let $\sigma_0 < \delta/\bar{\gamma}$ be small enough so that that (2.8) implies that $u \in \mathcal{N}$. Note that for $t \in \Omega_\delta(u)$,

$$|\mathcal{G}(u^*)(t)| \geq |\mathcal{G}(u)(t)| - \bar{\gamma}\sigma \geq \delta - \bar{\gamma}\sigma > 0.$$

Hence $t \in \Omega_{\delta - \bar{\gamma}\sigma}^* \subset A^*$ and the proof is complete. □

Assumption 2.4. There is $\nu \in (0, 1)$ such that $u_{\max}(t) \geq u_{\min}(t) + \nu$ for all $t \in \Omega$. A^* is the closure of a finite union of disjoint open components. There is $c_0 > 0$ such that for all $\delta > 0$ the sets

$$E_\delta = \{t \in R^d \mid \text{dist}(t, \partial A^*) < \delta\}$$

are uniformly bounded in measure by

$$\mu(E_\delta) \leq c_0 \delta^d. \tag{2.9}$$

On each component of A^* either $u = u_{\max}$ or $u = u_{\min}$.

Moreover, there is c_1 such that

$$\begin{aligned} &|\mathcal{G}(u^*)(t)| \geq c_1 \text{dist}(t, \partial A^*) \quad \text{for all } t \in A^* \quad \text{and} \\ &\text{dist}(t, A^*) \leq c_1^{-1} \text{dist}(u^*(t), \partial U(t)) \quad \text{for all } t \in I^*. \end{aligned} \tag{2.10}$$

Note that (2.9) and the assertion that on each component of A^*, either $u = u_{\max}$ or $u = u_{\min}$ follow from the previous parts of the assumption if $d = 1$. For $d > 1$ they may be viewed as regularity conditions on the boundary of A^*.

LEMMA 2.5. *Let Assumptions* 2.1 *and* 2.4 *hold. Then there are* c_2 *and* δ_1 *such that if* $\delta \le \delta_1$, *then* $\Omega_\delta^* \subset A^*$ *and*

$$\mu(A^* \setminus \Omega_\delta^*) \le c_2 \delta^d. \tag{2.11}$$

Proof. Let

$$\delta_1 = \min(\delta_0, \delta_0/c_1),$$

where δ_0 and c_1 are from Assumption 2.4. By Lemma 2.3 $\Omega_\delta^* \subset A^*$. If $t \in A^* \setminus \Omega_\delta^*$, then $|\mathcal{G}(u^*)(t)| < \delta$ and $u^*(t) \in \partial U(t)$. Hence, by (2.10) from Assumption 2.4 for $\delta \le \delta_1 \le \delta_0$

$$\operatorname{dist}(t, \partial A) < \delta/c_1.$$

This implies that

$$A^* \setminus \Omega_\delta^* \subset E_{\delta/c_1}$$

and therefore, by (2.9) from Assumption 2.4,

$$\mu(A^* \setminus \Omega_\delta^*) \le c_0 c_1^{-d} \delta^d.$$

This completes the proof with $c_2 = c_0 c_1^{-d}$. □

Let $\sigma_1 = \min(\sigma_0, \delta_1/\bar{\gamma})/2$. Under all of our assumptions, for a given $u \in C$ such that $\|u - u^*\|_X < \sigma \le \sigma_1$, we have by Lemma 2.4 that

$$\Omega_{\delta_1}(u) \subset \Omega_{\delta_1/2}^* \subset A^*.$$

Define

$$\bar{I} = \{t \mid \mathcal{K}(u)(t) \in \operatorname{int}(U)(t)\} \cap I^*. \tag{2.12}$$

Since $A^* \cap I^* = \emptyset$, $\Omega_{\delta_2}^* \subset A^*$, and $\bar{I} \subset I^*$, we can decompose Ω into the disjoint union

$$\Omega = \Omega_{\delta_2}^* \cup \bar{I} \cup R(u), \tag{2.13}$$

where $\delta_2 = \delta_1/2$ and

$$R(u) = \Omega \setminus (\bar{I} \cup \Omega_{\delta_2}^*). \tag{2.14}$$

As a trivial corollary to Lemma 2.5 we obtain the following estimate.

COROLLARY 2.6. *Let Assumptions* 2.1 *and* 2.4 *hold. If* $u \in C$ *and* $\|u - u^*\|_X < \sigma_1$ *then*

$$\mu((R(u) \cup \bar{I}) \setminus I^*) = \mu(A^* \setminus \Omega_{\delta_2}^*) \le c_2 \delta_2^d.$$

The next lemma will enable us to estimate the size of $R(u)$.

LEMMA 2.7. *Let Assumption* 2.4 *hold. There is* c_3 *such that for all* $\delta > 0$

$$\mu\left(\{t \mid \operatorname{dist}(\mathcal{K}(u^*)(t), \partial U(t)) < \delta\}\right) \le c_3 \delta^d. \tag{2.15}$$

Proof. Let

$$S = \{t \mid \operatorname{dist}(\mathcal{K}(u^*)(t), \partial U(t)) < \delta\}.$$

If $t \in S \cap A^*$, then

$$\operatorname{dist}(\mathcal{K}(u^*)(t), \partial U(t)) = \operatorname{dist}(\mathcal{K}(u^*)(t), \mathcal{P}(\mathcal{K}(u^*)(t))) = |\mathcal{G}(u^*)(t)| < \delta$$

and hence, by (2.10) from Assumption 2.4, $\operatorname{dist}(t, \partial A^*) < \delta/c_1$. Hence $S \cap A^* \subset E_{\delta/c_1}$.

If $t \in S \cap I^*$ then (2.10) from Assumption 2.4 implies that $\text{dist}(t, A^*) < c_1^{-1}\delta$ and therefore $S \cap I^* \subset E_{\delta/c_1}$. Hence $S \subset E_{\delta/c_1}$ and therefore

$$\mu(S) \leq \mu(E_{\delta/c_1}) \leq c_0 c_1^{-d} \delta^d.$$

This completes the proof with $c_3 = c_0 c_1^{-d}$. □

LEMMA 2.8. *Let Assumptions* 2.1 *and* 2.4 *hold. Then if* $u \in C$ *is such that* $\|u - u^*\|_X \leq \sigma_1$ *and* $R(u)$ *is defined by* (2.14), *then*

$$\mu(R(u)) \leq c_3 \delta_2^d. \tag{2.16}$$

Proof. If $t \in R(u)$ then $|\mathcal{G}(u^*)(t)| < \delta_2$ because $R(u) \cap \Omega^*_{\delta_2} = \emptyset$. We divide the remainder of the proof into two cases. $t \in R(u)$ implies that $|\mathcal{G}(u^*)(t)| < \delta_2$ and that either (1) $\mathcal{K}(u^*)(t) \notin \text{int}(U)(t)$ or (2) $\mathcal{K}(u)(t) \notin \text{int}(U)(t)$ and $\mathcal{K}(u^*)(t) \in \text{int}(U)(t)$.

In case (1) $|\mathcal{G}(u^*)(t)| < \delta_2$ implies that $\text{dist}(\mathcal{K}(u^*)(t), \partial U(t)) < \delta_2$. In case (2), Lipschitz continuity implies that

$$|\mathcal{K}(u)(t) - \mathcal{K}(u^*)(t)| \leq \gamma \sigma_1$$

and hence, since $\mathcal{K}(u^*)(t) \in \text{int}(U)(t)$ in case (2),

$$\text{dist}(\mathcal{K}(u^*)(t), \partial U(t)) \leq \gamma \sigma_1 \leq \bar{\gamma} \sigma_1 \leq \delta_2.$$

These estimates imply that

$$R(u) \subset \{t \mid \text{dist}(\mathcal{K}(u^*)(t), \partial U(t)) < \delta_2\}$$

and therefore, by Lemma 2.7,

$$\mu(R(u)) \leq c_3 \delta_2^d.$$

This completes the proof. □

COROLLARY 2.9. *Let Assumptions* 2.1 *and* 2.4 *hold. Then if* $u \in X$ *is such that* $\|u - u^*\|_X \leq \sigma_1$, *then*

$$\mu(\bar{I} \Delta I^*) \leq (c_2 + c_3) \delta_2^d.$$

LEMMA 2.10. *Let Assumption* 2.1 *hold. Then if* $u \in X$ *is such that* $\|u - u^*\|_X < \sigma_1$, *then*

$$\mathcal{P}(\mathcal{K}(u)(t)) = u^*(t)$$

for all $t \in \Omega^*_{\delta_2}$, *and*

$$|u(t) - u^*(t)| \leq \|\mathcal{F}(u)\|_X$$

for all $t \in \Omega^*_{\delta_2}$.

Proof. Let $t \in \Omega^*_{\delta_2} \subset A^*$. Without loss of generality we may assume that $u^*(t) = u_{\max}(t)$ and $\mathcal{K}(u^*)(t) \geq u_{\max}(t) + \delta_2$. Since $\|u - u^*\| < \sigma_1$,

$$\mathcal{K}(u)(t) \geq \mathcal{K}(u^*)(t) - \gamma \sigma_1 \geq u_{\max}(t) + \delta_2 - \bar{\gamma} \sigma_1 > u_{\max}(t).$$

Hence $\mathcal{P}(\mathcal{K}(u))(t) = u_{\max}(t) = u^*(t)$. Therefore

$$|u(t) - u^*(t)| = |u(t) - \mathcal{P}(\mathcal{K}(u)(t))| = |\mathcal{F}(u)(t)| \leq \|\mathcal{F}(u)\|_X.$$

This completes the proof. □

2.3. The main estimate. Now we reduce δ_1 if necessary so that

$$(c_2 + c_3)\delta_2^d < \eta_1, \tag{2.17}$$

where η_1 is the bound in Assumption 2.3. This has the effect of reducing δ_2 and σ_1. By Corollary 2.9 this implies that $\mathcal{G}'_{\bar{I}}(u)$ satisfies (2.5).

We can now formally state and prove the main result in this section

THEOREM 2.11. *Let Assumptions* 2.1–2.4 *hold. There are* $\sigma_2 > 0$ *and* $K > 0$ *such that if* $u \in C$ *and* $\|u - u^*\|_X \le \sigma_2$, *then*

$$\|u - u^*\|_X \le K\|\mathcal{F}(u)\|_X.$$

Proof. We let $\sigma_2 = \sigma_1$ for now. We will reduce σ_2 as the proof progresses. Let $\sigma^* = \|u - u^*\|_X \le \sigma_2$ and let $\|\mathcal{F}(u)\|_X = \epsilon$. We will estimate σ^* in terms of ϵ in the course of the proof in a way that can be applied to show $\sigma^* = O(\epsilon)$.

Without loss of generality we assume that $\epsilon < \sigma^*$ as if that is not the case the lemma holds with $K = 1$. We decompose the projected gradient map into three parts

$$u - \mathcal{P}(\mathcal{K}(u)) = \chi_{\bar{I}}(u - \mathcal{P}(\mathcal{K}(u))) + \chi_{R(u)}(u - \mathcal{P}(\mathcal{K}(u))) + \chi_{\Omega^*_{\delta_2}}(u - \mathcal{P}(\mathcal{K}(u))).$$

Since $\mathcal{K}(u) = u^*$ on $\Omega^*_{\delta_2}$ by Lemma 2.10 and $\mathcal{P}\mathcal{K}(u) = \mathcal{K}(u)$ on $\bar{I}$ by definition, we have

$$u - \mathcal{P}\mathcal{K}(u) = \chi_{\bar{I}}(u - \mathcal{K}(u)) + \chi_{R(u)}(u - \mathcal{P}(\mathcal{K}(u))) + \chi_{\Omega^*_{\delta_2}}(u - u^*).$$

Let $e = u - u^*$, $e_I = \chi_{\bar{I}}e$, $e_R = \chi_{R(u)}e$, and $e_\delta = \chi_{\Omega^*_{\delta_2}}e$. Therefore $\|e_\delta\|_X \le \|\mathcal{F}(u)\|_X = \epsilon$ by Lemma 2.10. To analyze the sizes of e_I and e_R, we note that $\mathcal{K}$ is a Lipschitz continuous map from $L^p(\Omega)$ to X with Lipschitz constant M_K, where M_K is the bound on $\|\mathcal{K}'\|_{\mathcal{L}(L^p(\Omega),X)}$ from (2.2), Hence

$$\|\mathcal{K}(u) - \mathcal{K}(u^* + e_I)\|_X \le \gamma\|e_\delta\|_X + M_K\|e_R\|_{L^p} \le \gamma\epsilon + M_K\|e_R\|_{L^p}.$$

We write $\chi_{\bar{I}}(u - \mathcal{K}(u))$ as

$$\chi_{\bar{I}}(u - \mathcal{K}(u)) = \chi_{\bar{I}}(u - \mathcal{K}(u^* + e_I)) + \zeta_1(u) = \chi_{\bar{I}}\mathcal{G}_{\bar{I}}(u) + \zeta_1(u), \tag{2.18}$$

where

$$\zeta_1(u) = \chi_{\bar{I}}(\mathcal{K}(u^* + e_I) - \mathcal{K}(u^* + e_I + e_R + e_\delta)).$$

Recalling that

$$\|\mathcal{K}'(u)\|_{\mathcal{L}(L^p,X)} \le M_K$$

for all $u \in \mathcal{N}$ we have, since $\|e_R\|_{L^p} \le \|e_R\|_X \mu(R(u))^{1/p}$,

$$\begin{aligned} \|\zeta_1(u)\|_X &\le \gamma\|e_\delta\|_X + M_K\|e_R\|_{L^p} \\ &\le \gamma\epsilon + M_K\mu(R(u))^{1/p}\|e_R\|_X \le \gamma\epsilon + M_K\mu(R(u))^{1/p}\sigma^*. \end{aligned} \tag{2.19}$$

Hence, by (2.5),

$$\|e_I\|_X \le M_2\|\zeta_1(u)\|_X \le M_2(\gamma\epsilon + M_K c_3^d \delta_2^{d/p}\sigma^*). \tag{2.20}$$

For convenience we rewrite (2.20) as

$$\|e_I\|_X \le c_4(\epsilon + \delta_2^{d/p}\sigma^*), \tag{2.21}$$

where $c_4 = M_2 \max(\gamma, M_K c_3^d)$.

Now, reduce δ_1 if necessary so that

$$c_4\delta_2^{d/p} < 1/(8\gamma) \quad \text{and} \quad M_K(c_3\delta_2^d)^{1/p} < 1/8.$$

At this point we have shown that if δ_1 is sufficiently small, then $\|e_\delta\| \le \|\mathcal{F}(u)\|$

$$\|e_I\| \le c_4\|\mathcal{F}(u)\| + \sigma^*/(8\gamma).$$

We will obtain a similar estimate for $\|e_R\|_X$ and then apply these estimates to obtain the conclusion $\|e\|_X = O(\epsilon)$.

Let $\xi = \chi_{R(u)}\mathcal{F}(u)$. Letting $e_T = e_I + e_\delta$, we have

$$\|e_T\|_X \le (1 + c_4)\epsilon + \sigma^*/(8\gamma).$$

Since $u^* = \mathcal{P}(\mathcal{K}(u^*))$ we may write

$$e_R = \xi + \zeta_2(u),$$

where

$$\zeta_2(u) = \chi_{R(u)}(\mathcal{P}(\mathcal{K}(u^* + e_T + e_R)) - \mathcal{P}(\mathcal{K}(u^*))).$$

Since $\|e_R\|_X \le \sigma^*$,

$$\begin{aligned}
\|\zeta_2(u)\|_X &\le \gamma\|e_T\|_X + \|\mathcal{P}(\mathcal{K}(u^* + e_R)) - \mathcal{P}(\mathcal{K}(u^*))\|_X \\
&\le \gamma((1 + c_4)\epsilon + \sigma^*/(8\gamma)) + M_K\mu(R(u))^{1/p}\|e_R\|_X \\
&\le \gamma(1 + c_4)\epsilon + \sigma^*/8 + M_K(c_3\delta_2^d)^{1/p}\sigma^* \\
&\le \gamma(1 + c_4)\epsilon + \sigma^*/4.
\end{aligned} \tag{2.22}$$

Here we use the fact that $\mathcal{P}$ has Lipschitz constant 1 because it is a projection onto a convex set.

Noting that $\|\xi\|_X \le \|\mathcal{F}(u)\|_X = \epsilon$, we have that

$$\|e_R\|_X \le \epsilon + \|\zeta_2(u)\|_X \le K_2\epsilon + \sigma^*/4,$$

where $K_2 = 1 + \gamma(1 + c_4)$. Therefore

$$\|e\|_X = \sigma^* \le \|e_R\|_X + \|e_T\|_X \le (1 + c_4 + K_2)\epsilon + 3\sigma^*/4$$

and

$$\sigma^*/4 \le (1 + c_4 + K_2)\epsilon.$$

This completes the proof with $K = 4(1 + c_4 + K_2)$. □

3. The algorithm. In this section we describe our Newton-like iteration in broad terms. The details of an efficient implementation will be presented in §4. Our first task is to formulate the projected Newton iteration and analyze its convergence properties. Following that, it is easy to describe the class of algorithms that we implement.

Assume that the assumptions of Theorem 2.11 hold. Let $u_c \in C$ be such that $\|e_c\| < \sigma_2$, where $e_c = u_c - u^*$. We let

$$\|\mathcal{F}(u_c)\|_X = \delta_c.$$

Let $\bar{p} \in (0, 1)$. We define the sets

$$\begin{aligned} A_c &= \{t \mid \mathcal{P}(\mathcal{K}(u_c))(t) \neq \mathcal{K}(u_c)(t),\ |\mathcal{G}(u_c)(t)| \geq \delta_c^{\bar{p}}\}, \\ I_c &= \{t \mid \mathcal{P}(\mathcal{K}(u_c))(t) = \mathcal{K}(u_c)(t)\}, \\ R_c &= \{t \mid \mathcal{P}(\mathcal{K}(u_c))(t) \neq \mathcal{K}(u_c)(t),\ |\mathcal{G}(u_c)(t)| < \delta_c^{\bar{p}}\}. \end{aligned} \tag{3.1}$$

Note that the point evaluations required to to determine the sets A_c, I_c, and R_c are well defined since $u_c \in C$. Note also that I_c is a closed set because $u_{\min}$ and $u_{\max}$ are continuous.

Using (2.6) as an iteration would, as we shall see, not produce iterates that converge in the uniform norm. We use an evaluation of $\mathcal{K}$ to remedy this. We propose the iteration

$$\begin{aligned} u^{1/3} &= \chi_{A_c}\mathcal{P}\mathcal{K}(u_c) + \chi_{I_c \cup R_c} u_c, \\ u^{2/3} &= u^{1/3} - \mathcal{G}'_{I_c}(u^{1/3})^{-1}\mathcal{F}(u^{1/3}), \\ u_+ &= \mathcal{P}(\mathcal{K}(u^{2/3})). \end{aligned} \tag{3.2}$$

In the final form of the algorithm we advocate here, we will replace

$$\mathcal{G}'_{I_c}(u^{1/3})^{-1}\mathcal{F}(u^{1/3})$$

with an approximation of the action of $\mathcal{G}'_{I_c}(u^{1/3})^{-1}$ on $\mathcal{F}(u^{1/3})$ that can be evaluated efficiently. We note here that even though the intermediate iterate $u^{1/3}$ is not continuous on [0,T], it is everywhere defined and continuous on the set $I_c \cup R_c$, and particularly on the closed set I_c; and so methods for approximation of integral operators on the space of continuous functions are applicable to the approximation of the action of $\mathcal{G}'_{I_c}(u^{1/3})^{-1}$. We defer the construction of this approximation to §4, and in this section consider (3.2) and estimate $e^{2/3}$ in the L^p norm, where p is the exponent in Assumption 2.2. That estimate leads directly to a uniform estimate for $e_+ = u_+ - u^*$ by Assumption 2.2.

LEMMA 3.1. *Let Assumptions* 2.1–2.4 *hold. Let* σ_2 *and* K *be the constants from Theorem* 2.11. *Let* $u_c \in X$ *be such that*

$$\|e_c\|_X \leq \min\{\sigma_2, (1/2 + \gamma/K)^{-1/(1-\bar{p})}/\gamma, \nu/\gamma\} \tag{3.3}$$

then $A_c \subset A^*$ *and* $u^{1/3} = u^*$ *on* A_c.

Proof. Since $\delta_c \leq \gamma\|e_c\|_X$, (3.3) implies that

$$\delta^{1-\bar{p}} < (1/2 + \gamma/K)^{-1}. \tag{3.4}$$

Let $t \in A_c$, since

$$|\mathcal{G}(u^*)(t)| \geq |\mathcal{G}(u_c)(t)| - \gamma\|e_c\| \geq \delta_c^{\bar{p}} - \gamma\delta_c/K \geq \delta_c/2$$

by (3.4). Hence $A_c \subset A^*$ by Lemma 2.3.

To complete the proof, note that if $t \in A_c$ then $\mathcal{K}(u)(t) \in \partial U(t)$. As

$$|u^*(t) - \mathcal{P}(\mathcal{K}(u^*))(t)| = |\mathcal{P}(\mathcal{K}(u))(t) - \mathcal{P}(\mathcal{K}(u^*))(t)| \le \gamma \|e_c\|_X < \nu$$

we must have $u^*(t) = \mathcal{P}(\mathcal{K}(u))(t) = u^{1/3}(t)$ for $t \in A_c$. This completes the proof. □

The set R_c is small. This is made precise by the following lemma.

LEMMA 3.2. *Let Assumptions* 2.1–2.4 *and* (3.3) *hold. Then if* $u_c \in X$ *is such that*

$$\|e_c\|_X \le (\gamma K)^{-1/(1-\bar{p})}/\gamma, \tag{3.5}$$

then

$$\mu(R_c) \le c_3 3^d \delta_c^{\bar{p}/d},$$

where c_3 *is the constant from Lemma* 2.7.

Proof. By (3.3) $\|e_c\|_X < 1/\gamma$. Note that if $t \in R_c$, then $\mathcal{P}(\mathcal{K}(u_c))(t) \in \partial U(t)$ and also

$$\begin{aligned} |\mathcal{K}(u_c)(t) - \mathcal{P}(\mathcal{K}(u_c))(t)| &\le |\mathcal{G}(u_c)(t)| + |\mathcal{F}(u_c)(t)| \\ &\le \delta_c^{\bar{p}} + \delta_c \le 2\delta_c^{\bar{p}}. \end{aligned}$$

The last estimate follows from the assumption that $\|e_c\|_X < 1/\gamma$, which implies that $\delta_c < 1$. Therefore

$$\begin{aligned} R_c &\subset \{t \mid \operatorname{dist}(\mathcal{K}(u_c)(t), \partial U(t)) < 2\delta_c^{\bar{p}}\} \\ &\subset \{t \mid \operatorname{dist}(\mathcal{K}(u^*)(t), \partial U(t)) < 2\delta_c^{\bar{p}} + \bar{\gamma}\|e_c\|\} \\ &\subset \{t \mid \operatorname{dist}(\mathcal{K}(u^*)(t), \partial U(t)) < 2\delta_c^{\bar{p}} + \bar{\gamma}K\delta_c\} \\ &\subset \{t \mid \operatorname{dist}(\mathcal{K}(u^*)(t), \partial U(t)) < 3\delta_c^{\bar{p}}\}. \end{aligned}$$

The last estimate above follows from (3.5). The conclusion of the lemma is a direct application of Lemma 2.7. □

A_c will serve as the approximation to the active set. An immediate corollary of Lemmas 3.1 and 3.2 is a theorem on identification of the active set.

THEOREM 3.3. *Let Assumptions* 2.1–2.4 *and* (3.3) *hold. Assume that* $\delta_c < 1$. *Then there is* c_A *such that*

$$\mu(A^* \setminus A_c) \le c_A \delta_c^{\bar{p}/d}.$$

Proof. If $t \in A^* \setminus A_c$ then either $t \in R_c$ or $t \in I_c \cap A^*$. If $t \in I_c \cap A^*$ then $\mathcal{G}(u_c)(t) = \mathcal{F}(u_c)(t)$ and Assumption 2.4 implies

$$\begin{aligned} c_1 \operatorname{dist}(t, \partial A^*) &\le |\mathcal{G}(u^*)(t)| \le |\mathcal{G}(u_c)(t)| + \gamma\|e_c\|_X \\ &= |\mathcal{F}(u_c)(t)| + \gamma\|e_c\|_X \le \delta_c(1 + \gamma/K). \end{aligned}$$

This implies that

$$I_c \cap A^* \subset E_{c_1^{-1}\delta_c(1+\gamma/K)}$$

and therefore

$$\mu(I_c \cap A^*) \le c_0(c_1^{-1}\delta_c(1 + \gamma/K))^d.$$

This completes the proof as $A^* \setminus A_c \subset R_c \cup (I_c \cap A^*)$ with

$$c_A = c_3 + c_0(c_1^{-1}(1 + \gamma/K))^d. \qquad \square$$

The reader should compare the following lemma with (2.18) and (2.19) that used a similar decomposition of Ω to obtain a similar result.

LEMMA 3.4. *Let Assumptions* 2.1–2.4 *hold. Let* u_c *satisfy the assumptions of Lemma* 3.2. *Then* $|e^{1/3}(t)| \le |e_c(t)|$ *for all* $t \in \Omega$, *and there are* $w_c \in X$ *and* $K_w > 0$ *such that*

$$\|w_c\|_{L^p(\Omega)} \le K_w \|e_c\|_X \delta_c^{\bar{p}/(pd)}$$

and

$$\mathcal{F}(u^{1/3}) = \chi_{I_c} \mathcal{G}_{I_c}(u^* + \chi_{I_c} e^{1/3}) - w_c.$$

Proof. By Lemma 3.1, $\chi_{A_c} \mathcal{F}(u^{1/3}) = \chi_{A_c} e^{1/3} = 0$. Hence

$$\begin{aligned}\mathcal{F}(u^{1/3}) &= u^{1/3} - \mathcal{P}(\mathcal{K}(u^* + \chi_{I_c \cup R_c} e_c)) \\ &= \chi_{I_c \cup R_c}(u^{1/3} - \mathcal{P}(\mathcal{K}(u^* + \chi_{I_c \cup R_c} e_c))).\end{aligned}$$

By Lemma 3.2,

$$\|\chi_{R_c} \mathcal{F}(u^{1/3})\|_{L^p} \le c_5 \|e_c\|_X \delta_c^{\bar{p}/(pd)},$$

where

$$c_5 = (c_3 3^d)^{1/p}.$$

Similarly,

$$\|\mathcal{K}(u^* + \chi_{I_c} e_c + \chi_{R_c} e_c) - \mathcal{K}(u^* + \chi_{I_c} e_c)\|_X \le M_K \|e_c\|_X \mu(R_c)^{1/p} \le c_6 \|e_c\|_X \delta_c^{\bar{p}/(pd)},$$

where $c_6 = M_K c_5$. Hence if we let

$$w_c = \chi_{R_c} \mathcal{F}(u_c^{1/3}) + \chi_{I_c}(\mathcal{K}(u^* + \chi_{I_c} e_c + \chi_{R_c} e_c) - \mathcal{K}(u^* + \chi_{I_c} e_c)),$$

note that for any $f \in X$, $\|f\|_{L^p} \le \mu(\Omega)^{1/p} \|f\|_X$, and set $K_w = (1 + \mu(\Omega)^{1/p} M_K) c_5$, then the proof is complete. □

A trivial corollary of the proof is found in Theorem 3.5.

THEOREM 3.5. *Let Assumptions* 2.1–2.4 *hold. Let* $u_c \in C$ *satisfy the assumptions of Lemma* 3.2 *and assume that* $\|e_c\|_{L^p}, \|e_c\|_X < 1$. *Then there is* K_P *such that*

$$\|e^{2/3}\|_{L^p} \le K_P \|e_c\|_X^{1+(\bar{p}/pd)}, \tag{3.6}$$

and there is K_N *such that*

$$\|e_+\|_X \le K_N \|e_c\|_X^{1+(\bar{p}/pd)}. \tag{3.7}$$

Proof. By Lemma 3.4,

$$u^{2/3} = u^{1/3} - \mathcal{G}'_{I_c}(u^{1/3})^{-1} \mathcal{G}_{I_c}(u^* + \chi_{I_c} e^{1/3}) - \mathcal{G}'_{I_c}(u^{1/3})^{-1} w_c.$$

By Assumption 2.3, $\|\mathcal{G}'_{I_c}(u^{1/3})^{-1}\|_{L^p} \le M_2$ and hence by Lemma 3.4

$$\begin{aligned}\|\mathcal{G}'_{I_c}(u^{1/3})^{-1} w_c\|_{L^p} &\le M_2 K_w \|e_c\|_X \delta_c^{\bar{p}/(pd)} \\ &\le M_2 K_w \bar{\gamma}^{\bar{p}/(pd)} \|e_c\|_X^{1+(\bar{p}/pd)}.\end{aligned}$$

By standard estimates for Newton iterates and Assumption 2.3

$$\|e^{1/3} - \mathcal{G}'_{I_c}(u^{1/3})^{-1} \mathcal{G}_{I_c}(u^* + \chi_{I_c} e^{1/3})\|_X \le M_2 \bar{\gamma} \|e^{1/3}\|_X / 2.$$

Hence

$$\|e^{1/3} - \mathcal{G}'_{I_c}(u^{1/3})^{-1} \mathcal{G}_{I_c}(u^* + \chi_{I_c} e^{1/3})\|_{L^p} \le \mu(\Omega)^{1/p} M_2 \bar{\gamma} \|e^{1/3}\|_X / 2.$$

The first assertion therefore holds with

$$K_P = M_2 K_w \bar{\gamma}^{\bar{p}/(pd)} + \mu(\Omega)^{1/p} M_2 \bar{\gamma}/2.$$

The second assertion follows from Assumption 2.2 with

$$K_N = M_K K_P.$$

This completes the proof. □

The algorithms we implement replace $\mathcal{G}'_{I_c}(u_c)^{-1}$ in (3.2) with an approximation. The behavior of these algorithms is described by the next theorem, which is a direct consequence of Theorem 3.5.

THEOREM 3.6. *Let the assumptions of Theorem* 3.5 *hold and let* u_+ *be defined by*

$$\begin{aligned} u^{1/3} &= \chi_{A_c} \mathcal{P}\mathcal{K}(u_c) + \chi_{I_c \cup R_c} u_c, \\ u^{2/3} &= u^{1/3} - B_c^{-1} \mathcal{F}(u^{1/3}), \\ u_+ &= \mathcal{P}(\mathcal{K}(u^{2/3})), \end{aligned} \tag{3.8}$$

where

$$\|(B_c^{-1} - \mathcal{G}'_{I_c}(u^{1/3})^{-1})\|_{\mathcal{L}(X, L^p)} \le \rho_c.$$

Then there is $K_C > 0$ *such that*

$$\|e_+\|_X \le K_N \|e_c\|_X^{1+(\bar{p}/pd)} + K_C \rho_c \|e_c\|_X. \tag{3.9}$$

Proof. This follows directly from Theorem 3.5 with $K_C = M_K \gamma$. To see this, note that

$$\begin{aligned} \|e^{2/3}\|_{L^p} &\le K_P \|e_c\|_X^{1+(\bar{p}/pd)} + \rho \|\mathcal{F}(u^{1/3})\|_X \\ &\le K_P \|e_c\|_X^{1+(\bar{p}/pd)} + \gamma \rho_c \|e_c\|_X. \end{aligned}$$ □

Finally, we give an inexact variant of Theorem 3.5. The analysis is like that of the finite dimensional case [9].

THEOREM 3.7. *Let the assumptions of Theorem* 3.5 *hold and let* u_+ *be defined by*

$$\begin{aligned} u^{1/3} &= \chi_{A_c} \mathcal{P}\mathcal{K}(u_c) + \chi_{I_c \cup R_c} u_c, \\ u^{2/3} &= u^{1/3} + \tilde{s}, \\ u_+ &= \mathcal{P}(\mathcal{K}(u^{2/3})), \end{aligned} \tag{3.10}$$

where $\tilde{s} \in X \cap C(I_c)$ *and*

$$\|\mathcal{G}'_{I_c}(u^{1/3})\tilde{s} + \mathcal{F}(u^{1/3})\|_{L^p} \le \rho_c \|\mathcal{F}(u^{1/3})\|_X. \tag{3.11}$$

Then there is $K_I > 0$ *such that*

$$\|e_+\|_X \le K_N \|e_c\|_X^{1+(\bar{p}/pd)} + K_I \rho_c \|e_c\|_X. \tag{3.12}$$

In particular, if $\rho_c = \rho$ *is independent of* $u_c \in \mathcal{N}$, *we have q-linear convergence and if* $\rho_n \to 0$ *as* $n \to \infty$ *the convergence is q-superlinear.*

Proof. Note that

$$\begin{aligned} e^{2/3} = e^{1/3} &- \mathcal{G}'_{I_c}(u^{1/3})^{-1} \mathcal{F}(u^{1/3}) \\ &- \mathcal{G}'_{I_c}(u^{1/3})^{-1} (\mathcal{G}'_{I_c}(u^{1/3})\tilde{s} + \mathcal{F}(u^{1/3})). \end{aligned}$$

Hence, as in the proof of Theorem 3.5,

$$\|e^{2/3}\|_{L^p} \le K_P \|e_c\|_X^{1+(\bar{p}/pd)} + M_2 \rho_c \|\mathcal{F}(u^{1/3})\|_X$$
$$\le K_P \|e_c\|_X^{1+(\bar{p}/pd)} + M_2 \gamma \rho_c \|e_c\|_X.$$

Therefore

$$\|e_+\|_X \le M_K \|e^{2/3}\|_{L^p}$$
$$\le M_K (K_P \|e_c\|_X^{1+(\bar{p}/pd)} + M_2 \gamma \rho_c \|e_c\|_X).$$

Setting $K_I = M_K M_2 \gamma$ completes the proof. □

4. Implementation and an example. As in [16] we construct the operators B_c using a collectively compact sequence of approximations to $\mathcal{K}$. Recall that a family of maps $\{K_\alpha\}_{\alpha \in \Lambda} \subset \mathcal{L}(X, Y)$ is collectively compact if

$$\cup_{\alpha \in \Lambda} K_\alpha B$$

is precompact in Y for every bounded set $B \subset X$.

We incorporate an estimate of the set I_c using a "coarse mesh" approximation. We consider a sequence of approximations $\{\mathcal{K}_m\}$ to $\mathcal{K}$. We refer to the equation $u - \mathcal{P}(\mathcal{K}_m(u)) = 0$ as the equation for level m. Such approximations can be obtained, for example, by replacing the integral in (1.3) by a quadrature rule. Typically the problem at level m is equivalent to a finite dimensional problem of dimension N_m, say. Our compactness assumption is given below.

Assumption 4.1. The sequence of maps $\mathcal{K}_m$ converge strongly to $\mathcal{K}$ in $\mathcal{N}$ and the family $\{\mathcal{K}'_m(u)\}_{m \le \infty, u \in \mathcal{N}}$ is uniformly Lipschitz continuous as a family of maps from $\mathcal{N}$ to $\mathcal{L}(L^p, C)$ and is a collectively compact subset of $\mathcal{L}(X, C)$.

In Assumption 4.1, $\mathcal{K}_\infty = \mathcal{K}$. In the case of smooth nonlinearities, [2], [13], Assumption 4.1, solvability of $\mathcal{F}(u) = 0$, uniform Lipschitz continuity of the family $\{\mathcal{K}'_m(u)\}_{m \le \infty, u \in \mathcal{N}}$, and nonsingularity of $\mathcal{F}'(u^*)$ would imply that $u^m = \mathcal{K}_m(u^m)$ would have a solution for m sufficiently large and that $u^m \to u^*$ in X. In the nonsmooth case considered here we must assume that the approximate problems have solutions that converge to u^*.

Assumption 4.2. For m sufficiently large there is a solution to the fixed point problem $u^m = \mathcal{K}(u^m) \in C$. u^m converges uniformly to u^*.

Our algorithm, like the original Atkinson–Brakhage algorithm, [3], [6], or multigrid methods of the second kind [11], begins with a coarse level l for which the problem at that level is solved. The Atkinson–Brakhage approach uses low-level information to construct an approximate inverse B_l^L that is used to solve the problem at a higher level L. Our general scheme may be described in terms of the transition from a solution u^L at level L to a solution u^{L+1} for level $L + 1$. The initial iterate at level $L + 1$ is $u_0^{L+1} = u^L$. We apply the iteration (3.2) with $B_c = B_l^{L+1}$ until $\|\mathcal{F}_{L+1}(u^{L+1})\|_X \le \epsilon_{L+1}$, where ϵ_{L+1} is an estimate for truncation error at level $L + 1$. For a sufficiently fine coarse mesh one would hope to require only one iterate for each fine level L. Any necessary transfer of information between grids between levels can be done through Nyström interpolation [3] or directly by polynomial interpolation [14], [16].

The remaining issue is the construction of the maps B_l^L. We denote

$$\mathcal{G}_{l,S}(u) = I - \chi_S \mathcal{K}'_l(u) \chi_S$$

and let

$$I_l = \{t \mid \mathcal{K}(u^l) = \mathcal{P}(\mathcal{K}(u^l))\}.$$

We describe the action of $(B_l^L)^{-1}$ on a function v by the basic formula

$$(B_l^L)^{-1}v = v + \chi_{I_c}\chi_{I_l}\mathcal{G}'_{l,I_l}(u^l)^{-1}\chi_{I_l}\chi_{I_c}\mathcal{K}'_L(u_c)\chi_{I_c}v. \tag{4.1}$$

In (4.1) u_c is the current iterate at level L for the computation of u^L. Assumptions 2.2, 4.1, and 4.2 imply that B_l^L will satisfy the assumptions of Theorem 3.5 with ρ independent of L if the index of the lowest level, l, is sufficiently large. As with the original Atkinson–Brakhage iteration, (4.1) is an approximation to a two-term Neumann series expansion of $(\mathcal{G}'_L(u_c))^{-1}$.

In [16] the levels corresponded to piecewise linear approximations of the functions and the equations at the levels were equivalent to nonlinear equations for the values of the piecewise linear functions at the nodal points. The action of the fine mesh derivative $w = \mathcal{K}'(u_c)v$ was approximated by a difference quotient. The action of $(I - \mathcal{K}'_l(u^l))^{-1}$ was approximated by forming $I - \mathcal{K}'_l(u^l)$ as a matrix by numerically differentiating in coordinate directions, forming an LU factorization of the resulting matrix and storing the factors. Then w was projected onto the coarse mesh, the equation $(I - \mathcal{K}'_l(u^l))z = w$ was solved, and z extended to the fine mesh. The final approximation was $B_c^{-1}v = v - z$. Of course, a similar technique could be used for the constrainted problems considered here, setting

$$w = \chi_{I_l}\chi_{I_c}\mathcal{K}'_L(u_c)\chi_{I_c}v,$$

forming a matrix equivalent of $\mathcal{G}'_{l,I_l}$ and factoring it, and then using that factorization to compute $z = (\mathcal{G}'_{l,I_l})^{-1}w$. Instead, we follow the technique of [12] and use a GMRES [19] iteration with a discrete L^2 inner product to solve a finite dimensional problem associated with the linear system

$$\mathcal{G}'_{l,I_l}(u^l)z = w, \tag{4.2}$$

and then use interpolation to approximate the solution to (4.2). The matrix-vector products that are required by GMRES are done with a forward difference approximation to the action of the Fréchet derivative. As our assumptions on the family $\{\mathcal{K}_l\}$ will guarantee that the condition number of $I - \mathcal{K}'_l(u^l)$ is bounded independently of l, the results in [18] guarantee that the behavior of a GMRES iteration will be mesh independent in the sense of [1].

The theory in [16] allowed for inaccuracy in the evaluation of $\mathcal{K}_L$, which can be introduced, say, by the control of relative and absolute errors in adaptive integration codes or ordinary differential equation (ODE) codes like DASSL. In [16] we showed how one should adjust this accuracy to take into account the expected error $\|u^L - u^*\|_X$. That analysis can be incorporated here in a direct way in the evaluation of the difference approximations to directional derivatives $\mathcal{K}(u)'w$ that are required by both the Atkinson–Brakhage algorithm in the approximation of $\mathcal{K}'_L$ at the fine mesh level and the solution of (4.2) by Newton-GMRES iteration. In [16] the analysis was complicated by the numerical Jacobian, and additional accuracy was needed in the coarse mesh function evaluations. The use of GMRES eliminates some of that complexity. As in [16] the use of inaccurate function evaluations introduces an absolute error in the iteration. Making this absolute error smaller as the iteration progresses means that the convergence rate of the algorithm becomes r-linear.

Assume that the compact fixed point maps, $\mathcal{K}_m$, can be evaluated to a given absolute accuracy τ_m, with smaller values of τ_m resulting in a increase of the cost of the nonlinear function evaluation. In the case of the method of lines solution of a parabolic partial differential equation, for example, τ_m would be the absolute error tolerance given to the routine that performs the integration in time. Letting $h_m = O(\sqrt{\tau_m})$, we approximate the action of $\mathcal{K}'_m(u)$ on a vector w by the forward difference map

$$A_m(u, w) = \begin{cases} 0, & w = 0, \\ \|w\|_X \frac{\mathcal{K}(u + h_m w/\|w\|_X) - \mathcal{K}(u)}{h_m}, & w \neq 0. \end{cases}$$

$A_m(u, w)$, though nonlinear in w, approximates $\mathcal{K}_m'(u)w$ up to a relative error of $O(h_l)$. We have

$$\|A_m(u, w) - \mathcal{K}_m'(u)w\|_X \leq (\gamma/2)h_m\|w\|_X. \tag{4.3}$$

Hence $\mathcal{G}_{l,I_l}'(u^l)z$, which is used in the construction of the Atkinson–Brakhage approximate inverse, would be approximated by

$$G_l(u^l, z) = I - \chi_{I_l}A_l(u^l, \chi_{I_l}z).$$

The effect of this approximation is that in the evaluation of the action of the Atkinson–Brakhage approximate inverse given by (4.1), $w = \mathcal{K}_L'(u_c)\chi_{I_c}v$ is approximated by $A_L(u_c, v)$ and an error of $O(h_L\|v\|_X)$ is made. Then $z = \mathcal{G}_{l,I_l}'(u^l)^{-1}\chi_{I_l}\tilde{w}$ is approximated by $\tilde{z}$, where $\tilde{z}$ is computed by using GMRES to solve $\mathcal{G}_{l,I_l}'(u^l)\tilde{z} = \chi_{I_l}\tilde{w}$ with $G_l(u^l, z)$ used in the matrix-vector product routine whenever $\mathcal{G}_{l,I_l}'(u^l)z$ is requested and an approximate L^2 inner product used in the GMRES routine itself. We use the zero function as the initial iterate to GMRES and request a reduction in the residual by a factor of $\tilde{\rho}$. Following [8], we assume that the GMRES routine returns a vector that satisfies

$$\|G_l(u^l, \tilde{z}) - \chi_{I_l}\tilde{w}\|_{2,l} \leq \tilde{\rho}\|\chi_{I_l}\tilde{w}\|_{2,l}. \tag{4.4}$$

In (4.4), $\|\cdot\|_{2,l}$ is the norm associated with the approximate L^2 norm for level l. In our examples from control problems, $\|\cdot\|_{2,l} = \|\cdot\|_{L^2}$ and the functions at level l are piecewise linear. In the example from integral equations the approximate inner product is given by a quadrature rule. In either case there are $C_\infty > 0$ and $N_l \to \infty$ such that

$$N_l^{-1}\|\cdot\|_X \leq \|\cdot\|_{2,l} \leq C_\infty\|\cdot\|_X. \tag{4.5}$$

We can use (4.4) and (4.5) to obtain a uniform estimate

$$\|G_l(u^l, \tilde{z}) - \chi_{I_l}\tilde{w}\|_X \leq \tilde{\rho}N_lC_\infty\|\chi_{I_l}\tilde{w}\|_X. \tag{4.6}$$

Using (4.3) we have

$$\|G_l(u^l, \tilde{z}) - \mathcal{G}_l'(u^l)\tilde{z}\|_X \leq (\gamma h_l/2)\|\tilde{z}\|_X$$

and hence by Assumption 2.3, if $h_l \leq 2/(M_2\gamma)$,

$$\|\tilde{z}\|_X \leq \frac{M_2\|G_l(u^l, \tilde{z})\|_X}{1 - M_2\gamma h_l/2} \leq \frac{M_2}{1 - M_2\gamma h_l/2}\|\chi_{I_l}\tilde{w}\|_X(1 + \tilde{\rho}N_lC_\infty).$$

Hence if

$$\tilde{\rho}N_lC_\infty < 1 \text{ and } h_l \leq 1/(M_2\gamma), \tag{4.7}$$

we have

$$\|\tilde{z}\|_X \leq 4M_2\|\chi_{I_l}\tilde{w}\|_X.$$

Summarizing, if (4.7) holds, then

$$\|\mathcal{G}_l'(u^l)\tilde{z} - \chi_{I_l}\tilde{w}\|_X \leq (\tilde{\rho}N_lC_\infty + 2M_2\gamma h_l)\|\chi_{I_l}\tilde{w}\|_X.$$

This leads us to the rule to thumb that $h_l = O(\tilde{\rho}N_l)$ as a guide to selection of the accuracy required by a coarse mesh function evaluation. Also we get insight into the tolerances $\tilde{\rho}$ and $\tau_l = O(h_l)$ from (4.7).

If a Newton-GMRES [8] iteration is used, Theorem 3.7 and the analysis above provides guidance in the choice of h_c as a function of N_c and ρ_c, where ρ_c is the factor in (3.11). In the computations reported below, we set $\rho_c = \tilde{\rho} = .001$ with a view toward making $\tilde{\rho} N_l C_\infty$ and ρ_n small.

The overall effect of the GMRES approximations is to introduce a relative error of $O(\tilde{\rho} N_l + h_l + h_L)$ into the approximate Newton iteration and, if $\tilde{\rho} N_l + h_l + h_L$ is sufficiently small, the overall rate of r-linear convergence will be preserved. Approximating $\mathcal{F}_L$ will introduce an absolute error of τ_L and accordingly τ_L should be reduced as the grids are refined to approximate the expected truncation error. Note that it is not necessary to approximate $\mathcal{K}_L(u_c)$, which is used in the evaluation of $\mathcal{F}_L$, and the perturbations used in the approximation of $\mathcal{K}_L'$ to the same accuracy. We use an accuracy of τ_l for the perturbed evaluation since that does not change the size of the relative error in B_l^{-1}.

As in [16], for an example we consider maps $\mathcal{K}$ defined by

$$\mathcal{K}(u) = -d(t, 1)/\alpha,$$

where for $x \in (0, 1)$ and $t \in (0, T)$,

$$\begin{aligned} -d_t &= d_{xx} + f_2(t, x), \quad d(T, x) = y(T, x) - z(x), \\ d_x(t, 0) &= h_l(t), \quad d_x(t, 1) = h_r(t), \end{aligned}$$

(4.8)

$$\begin{aligned} y_t &= y_{xx} + f_1(t, x), \quad y(0, x) = y_0(x), \\ y_x(t, 0) &= g_l(t), \quad y_x(t, 1) = g_r(y(t, 1)) + q(t) + u(t) \end{aligned}$$

where the right-hand sides and boundary values can be used to specify a solution to the problem. Discretization in space was by piecewise linear finite elements. The mesh at level L consisted of the $L + 1$ points $\{i/L\}_{i=0}^{L}$. The functions $\mathcal{K}_L$ were evaluated by using DASSL to solve the systems of ODEs obtained by discretizing (4.8) in space and setting the absolute error to τ_L. The sets A_c, R_c, and I_c were specified as in (3.1) with $\bar{p} = \frac{1}{2}$.

In the computations we set the right-hand sides and boundary conditions so that the solution to (4.8) was

$$y(t, x) = (1 + xt)/(1 + x^2 t)$$

and

$$d(t, x) = \alpha(1 - x(2x(1 - t) - 1)^2)(1 - \exp(-(t - 1/4)^2)),$$

with

$$u^*(t) = \mathcal{P}(d(t, 1)/\alpha) = \mathcal{P}((1 - (1 - 2t)^2)(1 - \exp(-(t - 1/4)^2))).$$

We imposed constraints given by

$$u_{\max}(t) = 1 + 10(t - .5)^2 \quad \text{and} \quad u_{\min}(t) = t/4.$$

The solution to (1.1) is plotted in Figure 1. The solution is Lipschitz continuous but not differentiable. Hence we would expect piecewise linear approximations to be first-order accurate.

We have constructed the examples in such a way that tthe solution u^* does not depend on α. The reason for this was to isolate the ill conditioning that arises from small values of α in

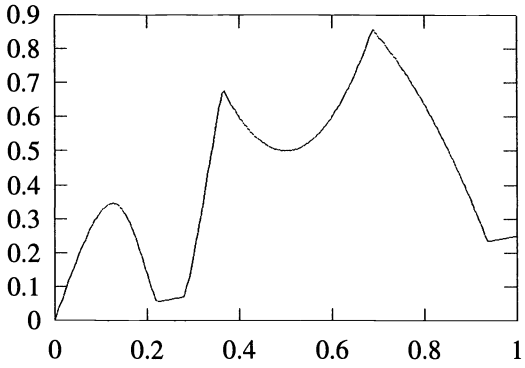


FIG. 1. *Plot of* u^*.

the operator and to make it possible to directly compare results for different values of α. We report results for $\alpha = 1, .5, .25, .1$. Since the Lipschitz constants of $\mathcal{K}$ are proportional to $1/\alpha$ and the solution of the linear equation for the step becomes very sensitive to high frequency perturbations of the solution for small α, one would expect that finer coarse meshes and more accurate initial iterates will be required for the smaller values of α. This is the case for both the Atkinson–Brakhage and Newton-GMRES forms of the algorithm.

The maps $\mathcal{K}_L$ were constructed exactly as reported in [16] and the absolute accuracy of the function evaluation at level L was $\tau_L = .001/L$, consistent with the expected first-order accuracy. In the forward differencing we used $h_l = .1/\sqrt{L}$. In the computations we used a coarse mesh of $N_l = l + 1$ points. Consistently with our requirement that $h_l = O(\tilde{\rho} N_l)$ we set $\tilde{\rho} = .01$. The iteration at the coarse mesh was terminated when $\|\mathcal{F}_l(u)\|_\infty < \epsilon_l = 10^{-4}$. The iteration at a higher mesh level L was terminated if $\|\mathcal{F}_L(u)\|_\infty < \epsilon_L = .01/L$. For both methods considered, projected Newton-GMRES and the Atkinson–Brakhage iteration, only one outer iterate was required for termination at the higher mesh levels. After that the mesh spacing was halved and the iteration continued. The initial iterate was

$$u_0 = \mathcal{P}(u(t) + \delta \sin(t)(t - .5)t).$$

δ controls the size of the initial iterate error and was reduced as α is reduced. The number of points in the coarse mesh N_l is increased as α is reduced.

In the tables that follow we report the norm ν_n of $\mathcal{F}_L$ at each iterate n and for the final iterate (n=1) at each level the ratio ν_1/ν_0 and the number of GMRES iterates I_G required for (3.11) to hold in the case of Newton-GMRES or needed to solve (4.2) in the case of the Atkinson–Brakhage iteration. In the header for each table we report α, δ, l, and the computation time T_C in seconds.

All computations were done on the CRAY Y-MP at the North Carolina Supercomputing Center running UNICOS 6.0. All codes were written in CRAY FORTRAN cft77 version 5.0.0.0. Computation times were taken from the output of the CRAY hardware performance monitor.

We report on two methods of solution. The first, reported in Tables 1–4, is a direct Newton-GMRES approach that satisfies (3.11) with a GMRES iteration, using $\rho_c = .001$.

In Tables 5–8, we report the results of the Atkinson–Brakhage algorithm using $\tilde{\rho} = .001$.

In both methods a fine mesh function evaluation was used to test for termination. A more efficient approach would be to see if the function norm at the next finer mesh has the predicted size of half the initial evaluation at the previous mesh. This would not avoid the

TABLE 1
Projected Newton-GMRES.

$\alpha = 1.0, \delta = .1, l = 20, T_C = 669$

L	n	I_G	ν_n	ν_n/ν_{n-1}
20	0		0.23E–01	
	1	3	0.26E–03	0.11E–01
40	0		0.70E–01	
	1	3	0.11E–02	0.15E–01
80	0		0.38E–01	
	1	3	0.39E–03	0.10E–01
160	0		0.14E–01	
	1	3	0.59E–05	0.42E–03
320	0		0.63E–02	
	1	3	0.15E–04	0.25E–02
640	0		0.45E–02	
	1	3	0.24E–05	0.54E–03
1280	0		0.28E–02	
	1	3	0.63E–05	0.23E–02
2560	0		0.68E–03	
	1	3	0.31E–07	0.45E–04
5120	0		0.42E–03	
	1	3	0.47E–06	0.11E–02

TABLE 2
Projected Newton-GMRES.

$\alpha = .5, \delta = .1, l = 40, T_C = 745$

L	n	I_G	ν_n	ν_n/ν_{n-1}
40	0		0.20E–01	
	1	3	0.53E–03	0.27E–01
80	0		0.37E–01	
	1	4	0.11E–02	0.30E–01
160	0		0.18E–01	
	1	3	0.11E–03	0.63E–02
320	0		0.66E–02	
	1	3	0.54E–04	0.81E–02
640	0		0.45E–02	
	1	3	0.17E–03	0.38E–01
1280	0		0.31E–02	
	1	3	0.16E–04	0.51E–02
2560	0		0.65E–03	
	1	3	0.15E–05	0.23E–02
5120	0		0.42E–03	
	1	4	0.58E–07	0.14E–03

fine mesh function evaluation to test for termination at the finest and ultimate mesh. Hence at least four fine mesh function evaluations per level were done: one to compute $\mathcal{K}(u_c)$, one to compute $\mathcal{K}(u^{1/3})$, one to compute $u_+ = \mathcal{P}(\mathcal{K}(u^{2/3}))$, and one to test for termination. The Newton-GMRES required one additional fine mesh evaluation for each inner iterate, while the Atkinson–Brakhage method required a low accuracy (i.e., using τ_l instead of τ_L) fine mesh function evaluation to compute $\mathcal{K}_L'(u_c)$ via a difference in (4.1) and at most a few coarse mesh evaluations for the GMRES iteration. This accounts for the advantage in the Atkinson–Brakhage method, which, as the tables show, executes in roughly 60% of the time of Newton-GMRES. Note that for the computations reported in the tables that required more GMRES iterations in the inner iteration of Newton-GMRES, the advantage of the Atkinson–Brakhage method was larger. Having said that, the nested iteration form of Newton-GMRES is still a fast algorithm for compact fixed-point problems since the number of inner iterations required at each mesh level is bounded independently of the mesh size.

TABLE 3
Projected Newton-GMRES.

$\alpha = .25, \delta = .05, l = 80, T_C = 651$

L	n	I_G	ν_n	ν_n/ν_{n-1}
80	0		0.18E–01	
	1	3	0.14E–02	0.74E–01
	2	2	0.21E–04	0.16E–01
160	0		0.15E–01	
	1	3	0.63E–03	0.42E–01
320	0		0.59E–02	
	1	3	0.49E–03	0.83E–01
640	0		0.50E–02	
	1	4	0.13E–03	0.26E–01
1280	0		0.29E–02	
	1	3	0.72E–04	0.25E–01
2560	0		0.74E–03	
	1	3	0.49E–05	0.67E–02
5120	0		0.42E–03	
	1	3	0.16E–05	0.37E–02

TABLE 4
Projected Newton-GMRES.

$\alpha = .1, \delta = .025, l = 640, T_C = 730$

L	n	I_G	ν_n	ν_n/ν_{n-1}
640	0		0.22E–01	
	1	3	0.68E–02	0.33E+00
	2	2	0.29E–03	0.43E–01
1280	0		0.26E–02	
	1	4	0.35E–03	0.14E+00
2560	0		0.85E–03	
	1	3	0.96E–04	0.11E+00
5120	0		0.48E–03	
	1	3	0.27E–05	0.55E–02

TABLE 5
Atkinson–Brakhage iteration.

$\alpha = 1.0, \delta = .1, l = 20, T_C = 391$

L	n	I_G	ν_n	ν_n/ν_{n-1}
20	0		0.23E–01	
	1	3	0.26E–03	0.11E–01
40	0		0.70E–01	
	1	3	0.82E–03	0.12E–01
80	0		0.38E–01	
	1	3	0.17E–03	0.45E–02
160	0		0.14E–01	
	1	2	0.88E–04	0.63E–02
320	0		0.63E–02	
	1	2	0.19E–04	0.30E–02
640	0		0.45E–02	
	1	3	0.14E–05	0.31E–03
1280	0		0.28E–02	
	1	3	0.11E–05	0.38E–03
2560	0		0.69E–03	
	1	3	0.52E–06	0.75E–03
5120	0		0.42E–03	
	1	3	0.14E–06	0.34E–03

TABLE 6
Atkinson–Brakhage iteration.

$\alpha = .5, \delta = .1, l = 40, T_C = 444$

L	n	I_G	v_n	v_n/v_{n-1}
40	0		0.20E–01	
	1	3	0.53E–03	0.27E–01
80	0		0.37E–01	
	1	3	0.12E–02	0.32E–01
160	0		0.18E–01	
	1	3	0.74E–03	0.42E–01
320	0		0.76E–02	
	1	3	0.14E–03	0.18E–01
640	0		0.48E–02	
	1	3	0.38E–04	0.79E–02
1280	0		0.29E–02	
	1	3	0.14E–04	0.50E–02
2560	0		0.65E–03	
	1	3	0.44E–05	0.67E–02
5120	0		0.42E–03	
	1	3	0.26E–05	0.61E–02

TABLE 7
Atkinson–Brakhage iteration.

$\alpha = .25, \delta = .05, l = 80, T_C = 428$

L	n	I_G	v_n	v_n/v_{n-1}
80	0		0.18E–01	
	1	3	0.14E–02	0.74E–01
	2	2	0.21E–04	0.16E–01
160	0		0.15E–01	
	1	3	0.43E–03	0.28E–01
320	0		0.67E–02	
	1	3	0.25E–02	0.37E+00
640	0		0.69E–02	
	1	2	0.21E–03	0.30E–01
1280	0		0.30E–02	
	1	2	0.24E–03	0.82E–01
2560	0		0.60E–03	
	1	2	0.57E–05	0.96E–02
5120	0		0.42E–03	
	1	2	0.70E–06	0.17E–02

TABLE 8
Atkinson–Brakhage iteration.

$\alpha = .1, \delta = .025, l = 440, T_C = 446$

L	n	I_G	v_n	v_n/v_{n-1}
640	0		0.22E–01	
	1	3	0.68E–02	0.33E+00
	2	2	0.29E–03	0.43E–01
1280	0		0.26E–02	
	1	3	0.15E–03	0.60E–01
2560	0		0.67E–03	
	1	4	0.24E–04	0.35E–01
5120	0		0.42E–03	
	1	4	0.83E–05	0.20E–01

REFERENCES

[1] E. L. ALLGOWER, K. BÖHMER, F. A. POTRA, AND W. C. RHEINBOLDT, *A mesh-independence principle for operator equations and their discretizations*, SIAM J. Numer. Anal., 23 (1986), pp. 160–169.

[2] P. M. ANSELONE, *Collectively Compact Operator Approximation Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[3] K. E. ATKINSON, *Iterative variants of the Nyström method for the numerical solution of integral equations*, Numer. Math., 22 (1973), pp. 17–31.

[4] D. B. BERTSEKAS, *On the Goldstein–Levitin–Polyak gradient projection method*, IEEE Trans. Autom. Control, 21 (1976), pp. 174–184.

[5] ———, *Projected Newton methods for optimization problems with simple constraints*, SIAM J. Control Optim., 20 (1982), pp. 221–246.

[6] H. BRAKHAGE, *Über die numerische Behandlung von Integralgleichungen nach der Quadraturformelmethode*, Numer. Math., 2 (1960), pp. 183–196.

[7] K. E. BRENAN, S. L. CAMPBELL, AND L. R. PETZOLD, *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*, Elsevier, New York, 1989.

[8] P. N. BROWN AND Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 450–481.

[9] R. DEMBO, E. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[10] W. HACKBUSCH, *On the fast solving of parabolic boundary control problems*, SIAM J. Control Optim., 17 (1979), pp. 231–244.

[11] ———, *Multi-Grid Methods and Applications*, Vol. 4, Springer Series in Computational Mathematics, Springer-Verlag, New York, 1985.

[12] M. HEINKENSCHLOß, C. T. KELLEY, AND H. T. TRAN, *Fast algorithms for nonsmooth compact fixed point problems*, SIAM J. Numer. Anal., 29 (1992), pp. 1769–1792.

[13] L. KANTOROVICH AND G. AKILOV, *Functional Analysis in Normed Spaces*, Pergamon Press, New York, 1964.

[14] C. T. KELLEY, *Operator prolongation methods for nonlinear equations*, in Computational Solution of Nonlinear Systems of Equations, E. L. Allgower and K. Georg, eds., Vol. 26, AMS Lectures in Applied Mathematics, American Mathematical Society, Providence, RI, 1990, pp. 359–388.

[15] C. T. KELLEY AND E. W. SACHS, *Multilevel algorithms for constrained optimal control problems*, Proc. Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, 1992.

[16] ———, *Fast algorithms for compact fixed point problems with inexact function evaluations*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 725–742.

[17] ———, *Mesh independence of the gradient projection method for optimal control problems*, SIAM J. Control Optim., 30 (1992), pp. 477–493.

[18] 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.

[19] 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.

[20] E. SACHS, *A parabolic control problem with a boundary condition of the Stefan–Boltzmann type*, ZAMM, 58 (1978), pp. 443–449.

[21] J. P. YVON, *Controle optimal d'un four industriel*, Tech. Report, University of Paris, 1973.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 668–680, May 1994

011

PRECONDITIONED RICHARDSON AND MINIMAL RESIDUAL ITERATIVE METHODS FOR PIECEWISE HERMITE BICUBIC ORTHOGONAL SPLINE COLLOCATION EQUATIONS*

BERNARD BIALECKI†

Abstract. The preconditioned Richardson and preconditioned minimal residual iterative methods are presented for the solution of linear equations arising when orthogonal spline collocation with piecewise Hermite bicubics is applied to a selfadjoint elliptic Dirichlet boundary value problem on a rectangle. For both methods, the orthogonal spline collocation discretization of Laplace's operator is used as a preconditioner. In the preconditioned Richardson method, an approximation of the optimal iteration parameter is computed from knowledge of spectral equivalence constants. Such a priori information is not required for the preconditioned minimal residual method. In each iteration of both methods, orthogonal spline collocation Poisson's problems are solved by a fast direct algorithm which employs fast Fourier transforms. An application of the preconditioned minimal residual method is also discussed for the solution of linear equations arising from the orthogonal spline collocation discretization of nonselfadjoint elliptic Dirichlet boundary value problems.

Key words. Dirichlet boundary value problem, orthogonal spline collocation, preconditioned iterative methods, Richardson method, minimal residual method, fast Fourier transforms

AMS subject classifications. 65N35, 65F10

1. Introduction.

Consider the Dirichlet boundary value problem

$$\begin{aligned} &Lu = f(x, y), \qquad (x, y) \in \Omega = (0, 1) \times (0, 1), \\ &u(x, y) = 0, \qquad (x, y) \in \partial\Omega, \end{aligned} \tag{1.1}$$

where L is the selfadjoint elliptic differential operator given by

$$Lu = -\frac{\partial}{\partial x}\left(a(x, y)\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(b(x, y)\frac{\partial u}{\partial y}\right) + c(x, y)u. \tag{1.2}$$

(The homogeneous boundary condition in (1.1) is chosen to simplify the presentation.) Let $\{t_i\}_{i=0}^N$ be a uniform partition of $[0, 1]$ such that $t_i = ih$, where $h = \frac{1}{N}$. Let $\mathcal{M}_h$ be the space of piecewise Hermite cubics on $[0, 1]$ defined by

$$\mathcal{M}_h = \{v \in C^1[0, 1] : v|_{[t_i, t_{i+1}]} \in P_3, i = 0, \ldots, N-1, v(0) = v(1) = 0\},$$

where P_3 denotes the set of all polynomials of degree ≤ 3. Let V_h be the space of piecewise Hermite bicubics on $\overline{\Omega}$ defined by $V_h = \mathcal{M}_h \otimes \mathcal{M}_h$; that is, V_h is the set of all functions on $\overline{\Omega}$ that are finite linear combinations of products of the form $v(x)w(y)$, where $v, w \in \mathcal{M}_h$. Let $\{\xi_j\}_{j=1}^{2N}$ be the Gauss points in $(0, 1)$ given by

$$\xi_{2i+1} = t_i + h\frac{3 - \sqrt{3}}{6}, \quad \xi_{2i+2} = t_i + h\frac{3 + \sqrt{3}}{6}, \quad i = 0, \ldots, N-1,$$

and let $\mathcal{G}$ be the collection of Gauss points in Ω defined by $\mathcal{G} = \{(x, y) : x, y \in \{\xi_j\}_{j=1}^{2N}\}$. The piecewise Hermite bicubic orthogonal spline collocation solution to (1.1) is a function $u_h \in V_h$ such that

$$Lu_h(\xi) = f(\xi), \qquad \xi \in \mathcal{G}. \tag{1.3}$$

*Received by the editors May 18, 1992; accepted for publication (in revised form) December 22, 1992. This research was supported in part by National Science Foundation grant CCR-9103451.

†Department of Mathematics, University of Kentucky, Lexington, Kentucky 40506 (bialecki@ms.uky.edu).

Sufficient conditions for the existence and uniqueness of u_h were given recently in [2], where the $O(h^3)$ rate of convergence of u_h to the exact solution u in the H^1-norm was also established.

When finite difference or finite element Galerkin methods are used to discretize (1.1), the resulting linear systems have symmetric, positive definite coefficient matrices. Such systems can be solved efficiently by the preconditioned Chebyshev or preconditioned conjugate gradient methods (see, for example, [3], and [14, §§6.2 and 8.4.1]) in which the discretization of the Laplacian plays the role of a preconditioner. In conjunction with alternating-direction methods, the idea of preconditioning finite difference linear systems was proposed by D'yakonov [6], and has been used by many researchers for the last three decades. For orthogonal spline collocation, the situation appears to be more complicated than that for a finite difference or a finite element Galerkin discretization, since even for Poisson's equation, the matrices arising from (1.3) are nonsymmetric. However, it was shown recently in [2] that the orthogonal spline collocation Laplace operator generating such matrices is selfadjoint and positive definite with respect to the discrete inner product, which is defined in terms of the composite Gauss–Legendre quadrature rule. Moreover, fast direct solvers based on fast Fourier transforms have been also developed in [1] for linear systems arising from the piecewise Hermite bicubic orthogonal spline collocation discretization of Poisson's equation. It is the purpose of this paper to discuss applications of these solvers to one-step preconditioned iterative methods for the solution of (1.3). Although the orthogonal spline collocation operator corresponding to this problem is not selfadjoint with respect to the discrete inner product, nevertheless it is spectrally equivalent to the orthogonal spline collocation Laplace operator. This spectral equivalence makes it possible to employ the preconditioned Richardson method and also the preconditioned minimal residual method that does not require a priori information for the selection of the iteration parameters.

Some recent works on preconditioned iterative methods for solving linear systems arising from discretizations of elliptic boundary value problems include [9] and [10]. In [9], the preconditioned conjugate gradient method applied to normal equations and the preconditioned Orthomin method were used for the solution of finite difference linear systems corresponding to nonselfadjoint elliptic boundary value problems. It was shown that the convergence rates of these methods were independent of the partition mesh size. In [10], the effect of boundary conditions was discussed on preconditioning discrete linear systems on the left and on the right. Conditions were given under which the condition numbers of the preconditioned matrices were uniformly bounded. It should be noted that preconditioned iterative methods based on spectrally equivalent operators have serious limitations. Even though the numbers of iterations in such methods are independent of the partition mesh size, they may still be quite large if the variations in the differential operator coefficients are large. This drawback of preconditioned iterative methods is characteristic of orthogonal spline collocation, finite difference, and finite element Galerkin discretizations.

A brief outline of this paper is as follows. The preconditioned Richardson and preconditioned minimal residual iterative methods for solving nonselfadjoint equations in a general Hilbert space framework are discussed in §2. These methods for solving the orthogonal spline collocation equations (1.3) are presented and analyzed in §3. In §4 an application of the preconditioned minimal residual method is considered for the solution of (1.3) when the differential operator L is nonselfadjoint. Finally, §5 contains results of numerical experiments and a brief discussion of these.

2. Preliminaries. In this section, following the approach of [14], we present the preconditioned Richardson and preconditioned minimal residual iterative methods for the solution of a nonselfadjoint positive definite linear operator equation in a general Hilbert space setting.

Let H be a finite dimensional Hilbert space over the field of real numbers and let $(\cdot, \cdot)$ and $\|\cdot\| = \sqrt{(\cdot, \cdot)}$ denote the inner product and norm in H, respectively. Let A be a linear operator from H into H. An operator A^* from H into H such that $(Av, w) = (v, A^*w)$ for all $v, w \in H$ is called the adjoint operator to A. The operator A is said to be *selfadjoint* if $A = A^*$. The operator A is said to be *positive definite* ($A > 0$) if $(Av, v) > 0$ for all nonzero $v \in H$. If A is selfadjoint and positive definite, then $\|v\|_A = \sqrt{(Av, v)}$ denotes the energy norm of $v \in H$ induced by A. The norm of the operator A is defined in the usual way, that is, $\|A\| = \sup_{x \neq 0} \|Ax\|/\|x\|$. If A and B are two linear operators from H into H, then $A \leq B$ means that $(Av, v) \leq (Bv, v)$ for all $v \in H$. In the following, E denotes the identity operator in H.

The following result plays an important role in establishing convergence rates of the iterative methods discussed in this paper.

LEMMA 2.1. *Assume that the linear operators A and B from H into H and the real numbers γ_i, $i = 1, 2, 3$, are such that*

$$B = B^* > 0, \quad \gamma_1 B \leq A \leq \gamma_2 B, \quad \gamma_1 > 0, \tag{2.1}$$

and

$$(A^* - A)B^{-1}(A - A^*) \leq 4\gamma_3^2 B, \qquad \gamma_3 \geq 0. \tag{2.2}$$

If

$$\overline{\tau} = \frac{2}{\gamma_1 + \gamma_2}(1 - \rho\kappa), \tag{2.3}$$

where

$$\rho = \frac{(1+\kappa)\gamma_2 - (1-\kappa)\gamma_1}{(1+\kappa)\gamma_2 + (1-\kappa)\gamma_1}, \qquad \kappa = \frac{\gamma_3}{\sqrt{\gamma_1\gamma_2 + \gamma_3^2}}, \tag{2.4}$$

then

$$\|E - \overline{\tau} D^{1/2} B^{-1} A D^{-1/2}\| \leq \rho \tag{2.5}$$

for any $D = S^*S$, where S is a linear nonsingular operator from H into H such that

$$SB^{-1}AS^{-1} = B^{-1/2}AB^{-1/2}. \tag{2.6}$$

Proof. Setting $\hat{A} = B^{-1/2}AB^{-1/2}$, it is easy to verify that

$$\|E - \tau D^{1/2} B^{-1} A D^{-1/2}\| = \|E - \tau \hat{A}\|, \qquad \tau \in \mathcal{R},$$

for any $D = S^*S$, where S satisfies (2.6). Since conditions (2.1) and (2.2) are equivalent to

$$\gamma_1 E \leq \hat{A} \leq \gamma_2 E, \qquad \|\hat{A} - \hat{A}^*\| \leq 2\gamma_3, \tag{2.7}$$

(2.5) follows from Lemma 4 in §6.4.2.2 of [14], where a certain bound on $\|E - \tau\hat{A}\|$ is minimized with respect to τ. □

It should be noted that the condition (2.6) is satisfied, for example, by $S = B^{1/2}(B^{-1}A)^l$, where l is a nonnegative integer.

It is important to observe that the formula (2.3) for $\overline{\tau}$ and the bound (2.5) are optimal in the sense that, if $A = A^*$, then

$$\min_{\tau} \|E - \tau D^{1/2} B^{-1} A D^{-1/2}\| = \left\| E - \frac{2}{\gamma_1' + \gamma_2'} D^{1/2} B^{-1} A D^{-1/2} \right\| = \frac{\gamma_2' - \gamma_1'}{\gamma_2' + \gamma_1'},$$

where γ_1' is the largest γ_1 such that $\gamma_1 B \leq A$, and γ_2' is the smallest γ_2 such that $A \leq \gamma_2 B$.

Let A be a linear nonsingular operator from H into H (A may be nonselfadjoint) and let $f \in H$. Assume that the equation

$$Au = f \tag{2.8}$$

is solved by the preconditioned Richardson (PR) method

$$B(u^{(k+1)} - u^{(k)}) = \tau r^{(k)}, \qquad k = 0, 1, \ldots, \tag{2.9}$$

where B is a nonsingular linear operator from H into H, τ is an iteration parameter to be selected, $u^{(0)} \in H$ is assumed to be given, and here and in what follows,

$$r^{(k)} = f - Au^{(k)}.$$

THEOREM 2.1. *Assume that A, B, and γ_i, $i = 1, 2, 3$, satisfy the assumptions of Lemma 2.1 and that $\bar{\tau}$ and ρ are given by (2.3) and (2.4), respectively. If $u^{(k)}$, $k = 1, \ldots,$ are generated by the* PR *method (2.9) with $\tau = \bar{\tau}$, then*

$$\|u^{(k)} - u\|_D \leq \rho^k \|u^{(0)} - u\|_D, \tag{2.10}$$

with $D = B$ and $D = A^* B^{-1} A$.

Proof. Using (2.8) and (2.9), it is easy to verify that

$$D^{1/2}(u^{(k+1)} - u) = (E - \tau D^{1/2} B^{-1} A D^{-1/2}) D^{1/2}(u^{(k)} - u),$$

and hence (2.10) follows easily from Lemma 2.1 on taking $S = B^{1/2}(B^{-1}A)^l$ with $l = 0, 1$. □

Let B be a nonsingular linear operator from H into H, and let D be an arbitrary linear operator from H into H such that $D = D^* > 0$. Consider solving (2.8) by the preconditioned one-step gradient method

$$B(u^{(k+1)} - u^{(k)}) = \tau_k r^{(k)}, \qquad k = 0, 1, \ldots, \tag{2.11}$$

where

$$\tau_k = \frac{(Dw^{(k)}, u - u^{(k)})}{(Dw^{(k)}, w^{(k)})}, \qquad Bw^{(k)} = r^{(k)}. \tag{2.12}$$

It should be noted that, for a given $u^{(k)}$, and $u^{(k+1)}$ defined by (2.11), (2.12) arises from minimizing $\|u^{(k+1)} - u\|_D$ with respect to τ_k. Therefore, if $u^{(k+1)}$ is computed from (2.11) with τ_k given in (2.12), then

$$\|u^{(k+1)} - u\|_D = \min_{\tau} \|(E - \tau D^{1/2} B^{-1} A D^{-1/2}) D^{1/2}(u^{(k)} - u)\|. \tag{2.13}$$

If $B = B^* > 0$ and $D = A^* B^{-1} A$, then (2.12) yields

$$\tau_k = \frac{(Aw^{(k)}, w^{(k)})}{(Aw^{(k)}, v^{(k)})}, \qquad Bw^{(k)} = r^{(k)}, \qquad Bv^{(k)} = Aw^{(k)}, \tag{2.14}$$

and the resulting method (2.11) can be regarded as the preconditioned minimal residual (PMR) method. It follows from (2.11) and (2.14) that the PMR method can be written in the following way:

$$
\begin{aligned}
&\text{Choose } u^{(0)} \in H, \text{ compute } r^{(0)} = f - Au^{(0)}, \text{ and solve } Bw^{(0)} = r^{(0)}.\\
&\text{For } k = 0, 1, \ldots,\\
&\qquad \text{Compute } z^{(k)} = Aw^{(k)}.\\
&\qquad \text{Solve } Bv^{(k)} = z^{(k)}.\\
&\qquad \text{Compute } \tau_k = (z^{(k)}, w^{(k)})/(z^{(k)}, v^{(k)}).\\
&\qquad \text{Compute } u^{(k+1)} = u^{(k)} + \tau_k w^{(k)}.\\
&\qquad \text{Compute } w^{(k+1)} = w^{(k)} - \tau_k v^{(k)}.
\end{aligned}
\tag{2.15}
$$

Clearly each iteration of the PMR method requires the solution of only one equation with the operator B.

THEOREM 2.2. *Assume that A, B, and γ_i, $i = 1, 2, 3$, satisfy the assumptions of Lemma 2.1 and that ρ is given by (2.4). If $u^{(k)}$, $k = 1, \ldots$, are generated by the PMR method (2.15), then*

$$
\|u^{(k)} - u\|_{A^*B^{-1}A} \le \rho^k \|u^{(0)} - u\|_{A^*B^{-1}A}. \tag{2.16}
$$

Proof. The inequality (2.16) follows easily from (2.13) with $D = A^*B^{-1}A$ and Lemma 2.1. □

The two methods described in this paper for the solution of (2.8) are referred to as the PR and PMR methods since they arise from applying the Richardson (R) method [15] and the minimal residual (MR) method [8], respectively, to the preconditioned operator equation

$$
\hat{A}\hat{u} = \hat{f}, \tag{2.17}
$$

where $\hat{A} = B^{-1/2}AB^{-1/2}$, $\hat{u} = B^{1/2}u$, $\hat{f} = B^{-1/2}f$. For example, the application of the MR method to (2.17) yields

$$
\hat{u}^{(k+1)} = \hat{u}^{(k)} + \hat{\tau}_k \hat{r}^{(k)}, \qquad \hat{r}^{(k)} = \hat{f} - \hat{A}\hat{u}^{(k)}, \qquad k = 0, 1, \ldots, \tag{2.18}
$$

where $\hat{u}^{(0)} \in H$ is given and

$$
\hat{\tau}_k = \frac{(\hat{A}\hat{r}^{(k)}, \hat{r}^{(k)})}{(\hat{A}\hat{r}^{(k)}, \hat{A}\hat{r}^{(k)})}. \tag{2.19}
$$

It is easy to verify that if $\hat{u}^{(k)}$ is generated by (2.18), (2.19), then $u^{(k)} = B^{-1/2}\hat{u}^{(k)}$ satisfies (2.11), (2.14).

The significance of the inequalities in (2.1) and (2.2) relating A and B becomes more transparent in view of their equivalence with the inequalities in (2.7). It is clear, for example, that γ_3 in the second inequality of (2.7) measures the size of the skew-symmetric part of the preconditioned operator $\hat{A}$.

It should be noted that the MR method applied to (2.17) is equivalent to the Orthomin(l) method [16] with $l = 0$. Of course, instead of using the MR method, one could apply to (2.17) the Orthomin(l), $l \ge 1$, or the generalized minimal residual (GMRES) method of [13], which is theoretically equivalent to the generalized conjugate residual method of [8]. Obviously

Thereom 2.2 remains valid for the Orthomin(l), $l \geq 1$, and the GMRES method, although these methods may converge faster than (2.16) seems to indicate. Since Orthomin(l) with large l and the GMRES method require, in general, significant amounts of work and storage per iteration, in this paper, we concentrate on the one-step PR and PMR methods, which, with a proper choice of the preconditioner B, have very modest work and storage requirements. Obviously, if A and B satisfy (2.1) and if, in addition, $A = A^*$, then (2.8) can be solved by the preconditioned conjugate gradient method [11] with B as a preconditioner, yielding (2.10) with $D = A$ and $2\rho_1^k$ in place of ρ^k, where $\rho_1 = (\sqrt{\gamma_2} - \sqrt{\gamma_1})/(\sqrt{\gamma_2} + \sqrt{\gamma_1})$ (see, for example, [14, §8.3.1]).

3. PR and PMR methods for orthogonal spline collocation. For the iterative methods of the previous section to be applicable to the solution of (1.3), results on spectral equivalence of elliptic orthogonal spline collocation operators must first be established.

3.1. Spectral equivalence of orthogonal spline collocation operators. It follows from Lemma 2.3 of [5] that any $v \in V_h$ is uniquely determined by its values on the set of Gauss points $\mathcal{G}$. Therefore V_h is a Hilbert space with the inner product $\langle \cdot, \cdot \rangle$ defined by

$$\langle v, w \rangle = \frac{h^2}{4} \sum_{\xi \in \mathcal{G}} (vw)(\xi), \qquad v, w \in V_h. \tag{3.1}$$

(The expression on the right-hand side in (3.1) arises by formally applying the composite Gauss–Legendre quadrature rule $\int_0^1 g(t)\,dt \approx (\frac{h}{2}) \sum_{j=1}^{M} g(\xi_j)$ with respect to each variable to the standard L^2-inner product $\int\int_\Omega (vw)(x, y)\,dxdy$.) Let L_h and Δ_h be the operators from V_h into V_h defined by

$$(L_h v)(\xi) = Lv(\xi), \qquad (\Delta_h v)(\xi) = \Delta v(\xi), \qquad \xi \in \mathcal{G}, \tag{3.2}$$

where L is given by (1.2) and Δ is the Laplacian. Obviously Δ_h is a special case of the operator L_h. It follows from Lemma 2.2 of [2] that $-\Delta_h$ is a selfadjoint and positive definite operator. The following theorem gives spectral equivalence results for the operators L_h and $-\Delta_h$.

THEOREM 3.1. *Assume that a is five times continuously differentiable in $\overline{\Omega}$ with respect to x, b is five times continuously differentiable in $\overline{\Omega}$ with respect to y, c is continuous in $\overline{\Omega}$, and that*

$$0 < \alpha \leq a(x, y),\ b(x, y) \leq \beta, \qquad (x, y) \in \overline{\Omega}. \tag{3.3}$$

Then

$$\left[\alpha + \frac{\gamma_*}{\lambda(h)} - C\eta(h)\right](-\Delta_h) \leq L_h \leq \left[\beta + \frac{\gamma^*}{\lambda(h)} + C\eta(h)\right](-\Delta_h), \tag{3.4}$$

$$(L_h^* - L_h)(-\Delta_h)^{-1}(L_h - L_h^*) \leq 4C^2\eta^2(h)(-\Delta_h), \tag{3.5}$$

where the positive constant C is independent of h, a, b, c, and where

$$\gamma_* = \min(0, \min_{(x,y)\in\overline{\Omega}} c(x, y)), \qquad \gamma^* = \max(0, \max_{(x,y)\in\overline{\Omega}} c(x, y)),$$

$$\lambda(h) = 24h^{-2}\frac{8 + \nu - \mu}{7 - \nu}, \qquad \nu = \cos(h\pi), \qquad \mu = \sqrt{43 + 40\nu - 2\nu^2}, \tag{3.6}$$

$$\eta(h) = h \max\left(\left\|\frac{\partial a}{\partial x}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial b}{\partial y}\right\|_{C(\overline{\Omega})}\right)$$

$$+h^2 \max\left(\left\|\frac{\partial^2 a}{\partial x^2}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^3 a}{\partial x^3}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^2 b}{\partial y^2}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^3 b}{\partial y^3}\right\|_{C(\overline{\Omega})}\right) \tag{3.7}$$

$$+h^3 \max\left(\left\|\frac{\partial^4 a}{\partial x^4}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^4 b}{\partial y^4}\right\|_{C(\overline{\Omega})}\right) + h^4 \max\left(\left\|\frac{\partial^5 a}{\partial x^5}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^5 b}{\partial y^5}\right\|_{C(\overline{\Omega})}\right).$$

Proof. Employing the approach used in the proof of Theorem 4.4 in [4] (see also the proof of Theorem 4.2 in [2]), it can be shown that

$$\langle L_h v, w\rangle = \mathcal{B}_h^{(1)}(v, w) + \mathcal{B}_h^{(2)}(v, w) + \langle c(x, y)v, w\rangle, \qquad v, w \in V_h, \tag{3.8}$$

where $\mathcal{B}_h^{(1)}$ and $\mathcal{B}_h^{(2)}$ are bilinear forms from $V_h \times V_h$ into R such that

$$\mathcal{B}_h^{(1)}(v, w) = \mathcal{B}_h^{(1)}(w, v), \tag{3.9}$$

$$\alpha\langle -\Delta_h v, v\rangle \le \mathcal{B}_h^{(1)}(v, v) \le \beta\langle -\Delta_h v, v\rangle, \tag{3.10}$$

$$|\mathcal{B}_h^{(2)}(v, w)| \le C\eta(h)\langle -\Delta_h v, v\rangle^{1/2}\langle -\Delta_h w, w\rangle^{1/2}, \tag{3.11}$$

where α, β, $\eta(h)$ are as in (3.3), (3.7), and the positive constant C is independent of h, a, b, c. Moreover, by Theorem 2.1 and Corollary 2.1 in [1],

$$\lambda(h)E \le -\Delta_h, \tag{3.12}$$

where $\lambda(h)$ is given in (3.6). Therefore (3.4) follows easily from (3.8) and (3.10)–(3.12).

For $w = (-\Delta_h)^{-1}(L_h - L_h^*)v$, (3.8), (3.9), and (3.11) give

$$\langle w, (L_h - L_h^*)v\rangle = \mathcal{B}_h^{(2)}(v, w) - \mathcal{B}_h^{(2)}(w, v) \le 2C\eta(h)\langle -\Delta_h v, v\rangle^{1/2}\langle w, (L_h - L_h^*)v\rangle^{1/2},$$

and hence

$$\langle w, (L_h - L_h^*)v\rangle \le 4C^2\eta^2(h)\langle -\Delta_h v, v\rangle,$$

which implies (3.5). □

3.2. PR and PMR methods. Let the operator L_h from V_h into V_h be defined by (3.2), and let $f_h \in V_h$ be such that $f_h(\xi) = f(\xi)$, $\xi \in \mathcal{G}$. Then (1.3) is equivalent to finding $u_h \in V_h$ such that

$$L_h u_h = f_h. \tag{3.13}$$

Assume that the assumptions of Theorem 3.1 are satisfied and that

$$c(x, y) > -2\pi^2\alpha, \qquad (x, y) \in \overline{\Omega}. \tag{3.14}$$

By Corollary 5.2 in [7], $\lambda(h) = 2\pi^2 + O(h^4)$, and hence

$$\alpha + \frac{\gamma_*}{\lambda(h)} = \alpha + \frac{\gamma_*}{2\pi^2} + O(h^4). \tag{3.15}$$

Therefore (3.4), (3.5), (3.7), (3.14), and (3.15) show that for h sufficiently small the operators $A = L_h$ and $B = -\Delta_h$ satisfy the assumptions of Lemma 2.1 in the Hilbert space $H = V_h$ with

$$\gamma_1 = \alpha + \frac{\gamma_*}{\lambda(h)} - C\eta(h), \quad \gamma_2 = \beta + \frac{\gamma^*}{\lambda(h)} + C\eta(h), \quad \gamma_3 = C\eta(h). \tag{3.16}$$

It follows from (2.9) that the PR method for (3.13) takes the form

$$-\Delta_h(u_h^{(k+1)} - u_h^{(k)}) = \bar{\tau}(f_h - L_h u_h^{(k)}), \qquad k = 0, 1, \ldots, \tag{3.17}$$

where $\bar{\tau}$ is defined by (2.3) with γ_1, γ_2, γ_3 of (3.16), and $u_h^{(0)} \in V_h$ is given. Using (2.8) of [12] and (3.2) of [5] it can be shown that there exist positive constants C_1 and C_2, independent of h, such that

$$C_1\|v\|_{H^1(\Omega)} \le \|v\|_{-\Delta_h} \le C_2\|v\|_{H^1(\Omega)}, \qquad v \in V_h. \tag{3.18}$$

Therefore it follows from (2.10) with $D = B$ that $u_h^{(k)}$ generated by (3.17) satisfies

$$\|u_h^{(k)} - u_h\|_{H^1(\Omega)} \le C\rho^k\|u_h^{(0)} - u_h\|_{H^1(\Omega)}, \tag{3.19}$$

where C is a positive constant independent of h, a, b, and c, and ρ is given by (2.4) with γ_1, γ_2, γ_3 of (3.16), and hence

$$\rho = \frac{\beta - \alpha + (\gamma^* - \gamma_*)/(2\pi^2)}{\beta + \alpha + (\gamma^* + \gamma_*)/(2\pi^2)} + O(h). \tag{3.20}$$

Equations (3.19) and (3.20) show that, for sufficiently small h, the convergence rate of the PR method applied to (3.13) is independent of h. It should be noted that the constant C appearing in (3.16) is not known explicitly, and hence the exact value of the parameter $\bar{\tau}$ given by (2.3) is not available. However, it is justifiable to take

$$\tilde{\tau} = \frac{2}{\alpha + \beta + (\gamma_* + \gamma^*)/(2\pi^2)}$$

in place of $\bar{\tau}$ in (3.17), since $\bar{\tau} = \tilde{\tau} + O(h)$. In fact, this was done in all of the numerical experiments involving the PR method and no noticeable deterioration of the convergence rate was observed.

It follows from (2.15) that the PMR method for solving (3.13) takes the form:

Choose $u_h^{(0)} \in V_h$, compute $r_h^{(0)} = f_h - L_h u_h^{(0)}$, and solve $-\Delta_h w_h^{(0)} = r_h^{(0)}$.

For $k = 0, 1, \ldots,$

$$\begin{aligned}
&\text{Compute } z_h^{(k)} = L_h w_h^{(k)}.\\
&\text{Solve } -\Delta_h v_h^{(k)} = z_h^{(k)}.\\
&\text{Compute } \tau_k = \langle z_h^{(k)}, w_h^{(k)}\rangle / \langle z_h^{(k)}, v_h^{(k)}\rangle.\\
&\text{Compute } u_h^{(k+1)} = u_h^{(k)} + \tau_k w_h^{(k)}.\\
&\text{Compute } w_h^{(k+1)} = w_h^{(k)} - \tau_k v_h^{(k)}.
\end{aligned} \tag{3.21}$$

To examine more closely the convergence rate of (3.21), we require the following result.

LEMMA 3.1. *Assume that the assumptions of Theorem* 3.1 *are satisfied. Then*

$$\left[\left(\alpha+\frac{\gamma_*}{2\pi^2}\right)^2+O(h)\right](-\Delta_h)\le L_h^*(-\Delta_h)^{-1}L_h \le\left[\left(\beta+\frac{\gamma^*}{2\pi^2}\right)^2+O(h)\right](-\Delta_h). \tag{3.22}$$

Proof. It is easy to see, by (3.8)–(3.11), that

$$\begin{aligned}\langle(-\Delta_h)^{-1}L_hv, L_hv\rangle &= \langle(-\Delta_h)^{-1}L_h^{(1)}v, L_h^{(1)}v\rangle \\ &\quad+\langle(-\Delta_h)^{-1}L_h^{(2)}v, L_h^{(2)}v\rangle+2\langle(-\Delta_h)^{-1}L_h^{(1)}v, L_h^{(2)}v\rangle,\end{aligned} \tag{3.23}$$

where

$$L_h^{(1)}=\left[L_h^{(1)}\right]^*, \qquad \delta_1(-\Delta_h)\le L_h^{(1)}\le\delta_2(-\Delta_h), \tag{3.24}$$

$$|\langle L_h^{(2)}v, w\rangle|\le C\eta(h)\langle-\Delta_hv, v\rangle^{1/2}\langle-\Delta_hw, w\rangle^{1/2}, \tag{3.25}$$

and $\delta_1=\alpha+\gamma_*/\lambda(h)$, $\delta_2=\beta+\gamma^*/\lambda(h)$. It follows from (3.24) that

$$\delta_1\left[L_h^{(1)}\right]^{-1}\le(-\Delta_h)^{-1}\le\delta_2\left[L_h^{(1)}\right]^{-1},$$

and hence

$$\delta_1^2\langle-\Delta_hv, v\rangle\le\langle(-\Delta_h)^{-1}L_h^{(1)}v, L_h^{(1)}v\rangle\le\delta_2^2\langle-\Delta_hv, v\rangle. \tag{3.26}$$

With $w=(-\Delta_h)^{-1}L_h^{(2)}v$, (3.25) implies that

$$\langle w, L_h^{(2)}v\rangle^{1/2}\le C\eta(h)\langle-\Delta_hv, v\rangle^{1/2},$$

and hence

$$0\le\langle(-\Delta_h)^{-1}L_h^{(2)}v, L_h^{(2)}v\rangle\le C^2\eta^2(h)\langle-\Delta_hv, v\rangle. \tag{3.27}$$

Using the Cauchy–Schwarz inequality, (3.26), and (3.27), we also obtain

$$\begin{aligned}|\langle(-\Delta_h)^{-1}L_h^{(1)}v, L_h^{(2)}v\rangle| &= |\langle(-\Delta_h)^{-1/2}L_h^{(1)}v, (-\Delta_h)^{-1/2}L_h^{(2)}v\rangle| \\ &\le C\delta_2\eta(h)\langle-\Delta_hv, v\rangle.\end{aligned} \tag{3.28}$$

Finally, (3.22) follows from (3.23), (3.26)–(3.28). □

If h is sufficiently small, then it follows from (2.16), (3.22), and (3.18) that the $u_h^{(k)}$ generated by (3.21) satisfy (3.19) with C independent of h, and ρ given by (3.20). Therefore, as was the case for the PR method, the convergence rate of the PMR method applied to (3.13) is independent of h.

3.3. Implementation and cost. Scaled standard Hermite basis functions $\{\phi_j\}_{j=0}^{2N+1}$ for the space of piecewise Hermite cubics on the partition $\{t_i\}_{i=0}^N$ of [0, 1] were used to implement the orthogonal spline collocation PR and PMR methods. For $i, j=0,\dots,N$, these basis functions are defined by

$$\phi_{2j}(t_i)=\delta_{i,j}, \quad \phi_{2j}'(t_i)=0, \quad \phi_{2j+1}(t_i)=0, \quad \phi_{2j+1}'(t_i)=h^{-1}\delta_{i,j},$$

where $\delta_{i,j}$ is the Kronecker delta. Clearly, if $v_h \in V_h$, then

$$v_h(x, y) = \sum_{i,j=0}^{2N+1} v_{i,j}\phi_i(x)\phi_j(y), \tag{3.29}$$

where

$$v_{2i,2j} = v_h(t_i, t_j), \qquad v_{2i+1,2j} = h\frac{\partial v_h}{\partial x}(t_i, t_j),$$

$$v_{2i,2j+1} = h\frac{\partial v_h}{\partial y}(t_i, t_j), \qquad v_{2i+1,2j+1} = h^2\frac{\partial^2 v_h}{\partial x \partial y}(t_i, t_j).$$

Hence any $v_h \in V_h$ is uniquely represented by either $\{v_h(\xi)\}_{\xi\in\mathcal{G}}$, the values of v_h on $\mathcal{G}$, or by $\{v_{i,j}\}_{i,j=0}^{2N+1}$, the expansion coefficients of (3.29). In both iterative methods, the successive iterates $u_h^{(k)}$ were represented in terms of their expansion coefficients $\{u_{i,j}^{(k)}\}_{i,j=0}^{2N+1}$. Collocation problems of the form $-\Delta_h v_h = w_h$, where $v_h, w_h \in V_h$, were solved by the fast Fourier transform solver of [1]. For a given $\{w_h(\xi)\}_{\xi\in\mathcal{G}}$, this solver requires $O(N^2 \ln N)$ arithmetic operations to produce $\{v_{i,j}\}_{i,j=0}^{2N+1}$. It is important to note that, for a given

$$\vec{v}_h = [v_{0,0}, \ldots, v_{0,2N+1}, \ldots, v_{2N+1,0}, \ldots, v_{2N+1,2N+1}]^T,$$

the evaluation of $\{v_h(\xi)\}_{\xi\in\mathcal{G}}$ involves multiplications of $\vec{v}_h$ by a matrix, which is the tensor product of $(2N+2) \times (2N+2)$ almost block diagonal matrices with four nonzero elements in each row. Therefore, the cost of such multiplications is $O(N^2)$. Similarly, for a given $\{v_{i,j}\}_{i,j=0}^{2N+1}$, $\{L_h v_h(\xi)\}_{\xi\in\mathcal{G}}$ is computed by multiplying $\vec{v}_h$ by four matrices each of which is the tensor product of $(2N+2) \times (2N+2)$ almost block diagonal matrices with four nonzero elements in each row. Therefore, the cost of one iteration in the orthogonal spline collocation PR or PMR method is $O(N^2 \ln N)$. It also follows from (3.19) and (3.20) that if $\|u_h^{(n)} - u_h\|_{H^1(\Omega)} = O(h^3)$ is to be guaranteed, then the number of iterations n should be proportional to $\ln N$. Thus the total cost of the orthogonal spline collocation PR or PMR method for solving (1.3) is $O(N^2 \ln^2 N)$.

4. Extension to nonselfadjoint boundary value problems. Consider the Dirichlet boundary value problem (1.1), where L is the elliptic differential operator given by

$$Lu = -\frac{\partial}{\partial x}\left(a(x, y)\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(b(x, y)\frac{\partial u}{\partial y}\right) + d(x, y)\frac{\partial u}{\partial x} + e(x, y)\frac{\partial u}{\partial y} + c(x, y)u.$$

The corresponding orthogonal spline collocation problem is given by (1.3) or equivalently (3.13), where $(L_h v)(\xi) = Lv(\xi)$, $\xi \in \mathcal{G}$. The following theorem is a counterpart of Theorem 3.1.

THEOREM 4.1. *Assume that a and d are five times continuously differentiable in $\overline{\Omega}$ with respect to x, b and e are five times continuously differentiable in $\overline{\Omega}$ with respect to y, c is continuous in $\overline{\Omega}$, and that* (3.3) *is satisfied. Then*

$$\left(\alpha + \frac{\gamma_*}{\lambda(h)} - C\,[\eta(h) + \sigma(h)]\right)(-\Delta_h) \le L_h \le \left(\beta + \frac{\gamma^*}{\lambda(h)} + C\,[\eta(h) + \sigma(h)]\right)(-\Delta_h),$$

$$(L_h^* - L_h)(-\Delta_h)^{-1}(L_h - L_h^*) \le 4C^2\,[\delta + \eta(h) + \sigma(h)]^2\,(-\Delta_h),$$

where the positive constant C is independent of h, a, b, d, e, and c, and $\lambda(h)$ is given by (3.6), *$\eta(h)$ is given by* (3.7), *and*

$$\gamma_* = \min(0, \min_{(x,y)\in\overline{\Omega}} g(x, y)), \qquad \gamma^* = \max(0, \max_{(x,y)\in\overline{\Omega}} g(x, y)),$$

$$\sigma(h) = h \max\left(\|d\|_{C(\overline{\Omega})}, \|e\|_{C(\overline{\Omega})}\right)$$

$$+ h^2 \max\left(\left\|\frac{\partial d}{\partial x}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^2 d}{\partial x^2}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial e}{\partial y}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^2 e}{\partial y^2}\right\|_{C(\overline{\Omega})}\right)$$

$$+ h^3 \max\left(\left\|\frac{\partial^3 d}{\partial x^3}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^3 e}{\partial y^3}\right\|_{C(\overline{\Omega})}\right)$$

$$+ h^4 \max\left(\left\|\frac{\partial^4 d}{\partial x^4}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^5 d}{\partial x^5}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^4 e}{\partial y^4}\right\|_{C(\overline{\Omega})}, \left\|\frac{\partial^5 e}{\partial y^5}\right\|_{C(\overline{\Omega})}\right),$$

$$\delta = \max\left(\|d\|_{C(\overline{\Omega})}, \left\|\frac{\partial d}{\partial x}\right\|_{C(\overline{\Omega})}, \|e\|_{C(\overline{\Omega})}, \left\|\frac{\partial e}{\partial y}\right\|_{C(\overline{\Omega})}\right),$$

where $g(x, y) = c(x, y) - [(\partial d/\partial x)(x, y) + (\partial e/\partial y)(x, y)]/2$.

Proof. Rewriting Lu in the form

$$Lu = -\frac{\partial}{\partial x}\left(a(x, y)\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(b(x, y)\frac{\partial u}{\partial y}\right)$$

$$+\frac{1}{2}\left[d(x, y)\frac{\partial u}{\partial x} + \frac{\partial}{\partial x}(d(x, y)u) + e(x, y)\frac{\partial u}{\partial y} + \frac{\partial}{\partial y}(e(x, y)u)\right] + g(x, y)u,$$

and using an approach similar to that in the proof of Theorem 3.1, it can be shown that

$$\langle L_h v, w\rangle = \sum_{i=1}^{4} \mathcal{B}_h^{(i)}(v, w) + \langle g(x, y)v, w\rangle,$$

where the bilinear forms $\mathcal{B}_h^{(1)}$ and $\mathcal{B}_h^{(2)}$ satisfy (3.9)–(3.11) and

$$\mathcal{B}_h^{(3)}(v, v) = 0, \qquad |\mathcal{B}_h^{(3)}(v, w)| \le C\delta\langle -\Delta_h v, v\rangle^{1/2}\langle -\Delta_h w, w\rangle^{1/2},$$

$$|\mathcal{B}_h^{(4)}(v, w)| \le C\sigma(h)\langle -\Delta_h v, v\rangle^{1/2}\langle -\Delta_h w, w\rangle^{1/2}.$$

The remainder of the proof is analogous to that of the proof of Theorem 3.1 and is therefore omitted. □

If

$$c(x, y) - \frac{1}{2}\left[\frac{\partial d}{\partial x}(x, y) + \frac{\partial e}{\partial y}(x, y)\right] > -2\pi^2\alpha, \qquad (x, y) \in \overline{\Omega},$$

then it follows from Theorem 4.1 that for h sufficiently small, the operators $A = L_h$ and $B = -\Delta_h$ satisfy the assumptions of Lemma 2.1. However, in contrast to the selfadjoint boundary value problem, the unknown $\gamma_3 = C[\delta + \eta(h) + \sigma(h)] = O(1)$, in general, and hence a good approximation to $\overline{\tau}$ of (2.3) is not readily available. Therefore, the PR method of (3.17) is not applicable for the solution of the orthogonal spline collocation problem corresponding to nonselfadjoint L. Nevertheless, this problem can be still solved by the PMR method (3.21) with the convergence rate given by (2.16).

5. Numerical experiments. The purpose of the numerical experiments was to verify the convergence rates of the PR and PMR orthogonal spline collocation methods rather than the accuracy of orthogonal spline collocation itself. Therefore the problem (1.1), (1.2) with

$$a(x, y) = e^{xy}, \quad b(x, y) = 1 + \sin(xy), \quad c(x, y) = 0, \quad u(x, y) = 10x(1 - x)y(1 - y),$$

was discretized by orthogonal spline collocation with piecewise Hermite bicubics that led to (1.3) with $u_h = u$. Starting with $u_h^{(0)} = 0$, for different values of the stepsize $h = \frac{1}{N}$, the PR method (3.17) and the PMR method (3.21) were used to compute sucessive iterates $u_h^{(k)}$ converging to u_h. Since (3.3) is satisfied with $\alpha = 1$ and $\beta = e$, following the discussion of §3.2, the iteration parameter $\bar{\tau}$ in (3.17) was replaced with $\tilde{\tau} = 2/(1 + e)$. Table 1 presents the discrete maximum norm errors

$$\varepsilon_N^{(k)} = \max_{0 \le i, j \le N} |(u_h - u_h^{(k)})(t_i, t_j)|$$

for both the PR and PMR methods. (The errors $\varepsilon_N^{(k)}$ were computed for convenience since the $H^1(\Omega)$ norm errors require the computation of integrals.)

TABLE 1
Errors $\varepsilon_N^{(k)}$ *in the* PR *and* PMR *methods.*

k	0	2	4	6	8
$\varepsilon_5^{(k)}$ in PR	.6	$.8 \times 10^{-1}$	$.1 \times 10^{-1}$	$.2 \times 10^{-2}$	$.3 \times 10^{-3}$
$\varepsilon_5^{(k)}$ in PMR	.6	$.2 \times 10^{-1}$	$.2 \times 10^{-2}$	$.2 \times 10^{-3}$	$.3 \times 10^{-4}$
$\varepsilon_{50}^{(k)}$ in PR	.6	$.8 \times 10^{-1}$	$.1 \times 10^{-1}$	$.2 \times 10^{-2}$	$.3 \times 10^{-3}$
$\varepsilon_{50}^{(k)}$ in PMR	.6	$.2 \times 10^{-1}$	$.2 \times 10^{-2}$	$.3 \times 10^{-3}$	$.4 \times 10^{-4}$

The obtained results show that the convergence rates of both methods were independent of h which confirms the theoretical results of §3.2. The numerical results also indicate, as expected, that the PMR method converges faster than the PR method, and hence is more efficient with respect to the number of arithmetic operations since each iteration of the PMR method requires about the same number of arithmetic operations as one iteration of the PR method. It should be noted that the PMR method does not have to converge faster than the PR method, due to the different paths taken by the PMR and PR methods. However, in practice, the PMR method usually does outperform the PR method.

REFERENCES

[1] B. BIALECKI, G. FAIRWEATHER, AND K. R. BENNETT, *Fast direct solvers for piecewise Hermite bicubic orthogonal spline collocation equations*, SIAM J. Numer. Anal., 29 (1992), pp. 156–173.

[2] B. BIALECKI AND X.-C. CAI, *H^1-error bounds for C^1 piecewise Hermite bicubic orthogonal spline collocation schemes*, SIAM J. Numer. Anal., to appear.

[3] P. CONCUS AND G. H. GOLUB, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, SIAM J. Numer. Anal., 10 (1973), pp. 1103–1120.

[4] K. D. COOPER AND P. M. PRENTER, *Alternating direction collocation for separable elliptic partial differential equations*, SIAM J. Numer. Anal., 28 (1991), pp. 711–727.

[5] J. DOUGLAS, JR. AND T. DUPONT, *Collocation Methods for Parabolic Equations in a Single Space Variable*, Lecture Notes in Mathematics 385, Springer-Verlag, New York, 1974.

[6] E. G. D'YAKONOV, *On an iterative method for the solution of finite difference equations*, Dokl. Akad., Nauk SSSR, 138 (1961), pp. 522–525.

[7] W. R. DYKSEN, *Tensor product generalized* ADI *methods for separable elliptic problems*, SIAM J. Numer. Anal., 24 (1987), pp. 59–76.

[8] S. C. EISENSTAT, H. C. ELMAN, AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equation*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[9] H. C. ELMAN AND M. H. SCHULTZ, *Preconditioning by fast direct methods for nonselfadjoint nonseparable elliptic equations*, SIAM J. Numer. Anal., 23 (1986), pp. 44–57.

[10] T. A. MANTEUFFEL AND S. V. PARTER, *Preconditioning and boundary conditions*, SIAM J. Numer. Anal., 27 (1990), pp. 656–694.

[11] J. M. ORTEGA, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

[12] P. PERCELL AND M. F. WHEELER, *A C^1 finite element collocation method for elliptic equations*, SIAM J. Numer. Anal., 17 (1980), pp. 605–622.

[13] Y. SAAD AND M. H. SCHULTZ, GMRES: *Generalized minimal residual algoritheorem for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[14] A. A. SAMARSKII AND E. S. NIKOLAEV, *Numerical Methods for Grid Equations, Volume* II *Iterative Methods*, Birkhäuser-Verlag, Basel, 1989.

[15] D. M. YOUNG, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

[16] D. M. YOUNG AND K. C. JEA, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 681–703, May 1994

012

TOWARDS POLYALGORITHMIC LINEAR SYSTEM SOLVERS FOR NONLINEAR ELLIPTIC PROBLEMS*

ALEXANDRE ERN†, VINCENT GIOVANGIGLI‡, DAVID E. KEYES§, AND MITCHELL D. SMOOKE¶

Abstract. The authors investigate the performance of several preconditioned conjugate gradient-like algorithms and a standard stationary iterative method (block-line successive overrelaxation (SOR)) on linear systems of equations that arise from a nonlinear elliptic flame sheet problem simulation. The nonlinearity forces a pseudotransient continuation process that makes the problem parabolic and thus compacts the spectrum of the Jacobian matrix so that simple relaxation methods are viable in the initial stages of the solution process. However, because of the transition from parabolic to elliptic character as the timestep is increased in pursuit of the steady-state solution, the performance of the candidate linear solvers spreads as the domain of convergence of Newton's method is approached. In numerical experiments over the course of a full nonlinear solution trajectory, short recurrence or optimal Krylov algorithms combined with a Gauss–Seidel (GS) preconditioning yield better execution times with respect to the standard block-line SOR techniques, but SOR performs competitively at a smaller storage cost until the final stages. Block-incomplete factorization preconditioned methods, on the other hand, require nearly a factor of two more storage than SOR and are uniformly less effective during the pseudotransient stages. The advantage of GS preconditioning is partly attributable to the exploitation of a dominant convection direction in the examples; nevertheless, a multidomain version of GS with streamwise coupling lagged at rows between adjacent subdomains incurs only a modest penalty.

Key words. nonlinear elliptic boundary value problems, polyalgorithms, preconditioned iterative methods, computational combination

AMS subject classifications. 65F10, 65N22, 65Y20, 80A32

1. Introduction.

The impossibility of uniformly ranking linear system solvers in order of effectiveness within broad categories of large, sparse, nonsingular systems arising from implicit discretizations of nonlinear elliptic partial differential equations (PDEs) is widely appreciated. Shifts in the balance of symmetric to nonsymmetric parts of a linear operator caused by mesh refinement, translations of the spectrum through the addition of a diagonal matrix representing a transient term in the PDE, reorderings, and changes in the granularity of implicitness in preconditioners (motivated, for instance, by architectural considerations) are all capable of causing shifts in the ranking of linear solvers on the same PDE. When the character of the solution itself evolves over the course of solving a nonlinear problem, when convergence tolerances vary in the style of inexact Newton methods, and when Jacobian evaluation costs and memory limitations intrude, it is not clear that the most effective overall algorithm can rely on a single linear solver, or even a single family of solvers.

As part of an ongoing effort to expand combustion modelling capabilities, we compare several nonsymmetric iterative solvers on flame sheet problems, which lie along the natural route to the numerical simulation of multidimensional diffusion flames with detailed chemistry and complex transport combustion models [1], [2]. Diffusion (or "nonpremixed") flames, in turn, are important in the study of the interaction of heat and mass transfer with chemical reaction

*Received by the editors May 18, 1992; accepted for publication (in revised form) December 31, 1992.

†Department of Mechanical Engineering, Yale University, New Haven, Connecticut 06520-2159 (alex@biomed.med.yale.edu) and Ecole Nationale des Ponts et Chaussées, 75007 Paris, France. The work of this author was supported in part by Office of Basic Energy Sciences, Department of Energy grant DE-F602-88ER13966 and by a grant from Ecole Nationale des Ponts et Chaussées, Paris France.

‡Centre de Mathematiques Appliquées and Centre National de la Recherche Scientifique, Ecole Polytechnique, 91128, Palaiseau Cedex, France (giovangi@cmapx.polytechnique.fr).

§Insitute for Computer Applications in Science and Engineering, Mail Stop 132C, NASA-LaRC, Hampton, Virginia 23681-0001 (keyes@icase.edu). The work of this author was supported by National Science Foundation contract ECS-8957475.

¶Department of Mechanical Engineering, Yale University, New Haven, Connecticut 06520-2159 (smooke%smooke@venus.ycc.yale.edu). The work of this author was supported in part by Office of Basic Energy Sciences, Department of Energy grant DE-F602-88ER13966.

in jet turbines, commercial burners, and reactors for materials processing. The terminology "flame sheet" refers to an infinitesimally thin flame zone located at the locus of stoichiometric mixing of fuel and oxidizer in a nonpremixed flame. In this limit, corresponding to infinitely fast conversion of reactants into stable products, it is impossible to recover any information about minor or intermediate species; however, the temperature distribution inside the reaction zone can be adequately predicted by the flame sheet model for many important fuel-oxidizer combinations and configurations. Moreover, a flame sheet model adds only one field to the hydrodynamic fields that describe the underlying flow, whereas a detailed kinetics model of a hydrocarbon air flame adds as many fields as species considered in the kinetic mechanism, each with its own coupled conservation equation. Since being studied as a means of obtaining an approximate solution for use as an initial iterate for a one-dimensional detailed-kinetics computation in [3], flame sheets have been routinely employed to initialize multidimensional nonpremixed flames.

Improving the efficiency of practical combustion systems and reducing their environmental impact will require improved physical models and improved computational simulation capabilities in many respects. The computational issues include resolution, nonlinear convergence, and linear convergence. Typical combustion problems may involve dozens of species defined at each grid point, and may require resolution of curved fronts whose thickness is on the order of thousandths of the domain diameter, across which critical fields vary by an order of magnitude or more. As a result, computations generally expand to fill all available memory, and any workspace required by the solution algorithm comes directly at the expense of resolution. The power-law dependence of the species mass fractions on each other and the exponentially nonlinear dependence of these fields on the temperature (through Arrhenius-type source terms in the conservation equations) call for Newton methods with sophisticated control strategies, including adaptive continuation techniques and damping. Finally, robust Newton algorithms depend on robust solvers for large nonsymmetric, possibly indefinite linear systems.

Meanwhile, there has been a burgeoning effort to design efficient algorithms to solve large sparse systems of the form

$$Ax = b \tag{1.1}$$

such as arise from finite difference or finite element discretizations of PDEs. In the case of symmetric positive definite systems, the classical conjugate gradient (CG) algorithm in combination with an efficient preconditioning technique is a powerful method for solving (1.1). For instance, a modified incomplete Cholesky CG can solve the discretized Poisson equation to truncation error accuracy in near optimal $\mathcal{O}(N^{1+1/2d} \log N)$ operations in dimensions $d = 2$ or 3 [4]. However, in the case of nonsymmetric matrices, such as those arising from the discretization of systems with convection or reaction, CG meets a fundamental difficulty. The orthogonality of the search directions cannot be maintained by using three-term recurrence relationships; all intermediate directions are needed for the computation of the next one. The alternative of working with the system of normal equations, $A^T Ax = A^T b$, is doubly unattractive: its condition number is much worse than that of the original, and, for a reason to be mentioned later, we prefer not to depend upon the ability to form the action of the transpose.

Ideally, any generalization of CG to nonsymmetric matrices would retain the two main features of the original algorithm: the optimality of some error or residual norm over a Krylov subspace, and constant work and storage requirements at each iteration. Unfortunately, there is no CG-type algorithm that fulfills both of these requirements [5].

Among the most successful schemes satisfying the first requirement is generalized minimal residual (GMRES) [6] (with its recent pseudo-Krylov variants flexible generalized

minimal residual (FGMRES) [7] and GMRESR [8]), in which the residuals satisfy a minimization property over conveniently generated subspaces. However, in these schemes, storage grows linearly and works quadratically with the iteration number so that, in most practical situations, it is necessary to use a restarted version that may seriously degrade convergence. In a second class of approaches, the strict norm minimization is abandoned and the iterates are instead defined by a Galerkin condition. This leads to simple three-term recurrence relationships, but also to irregular convergence behavior and numerical instabilities. This may be regarded as a theoretical weakness; in practice, however, numerical experiments reveal less serious consequences than might be expected. Typical algorithms of the second category are the CG squared (CGS) [9], biconjugate gradient stabilized (Bi-CGSTAB) [10], and the quasi-minimal residual (QMR) family [11], [12].

An interesting comparison of methods for nonsymmetric problems based on the normal equations, residual minimization, and Galerkin conditions is given in [13]. Each candidate method is notably superior to the rest on some test problems and notably inferior to the rest on some others. However, linear operators derived from elliptic PDEs are not represented in [13]. Because of the breadth of this problem class, it seems incumbent on investigators in different application areas to undertake custom parametric studies before choosing production algorithms. This paper is one such study.

In §2, we briefly describe the candidate algorithms, each of which is well documented elsewhere: the "baseline" SOR along with GMRES and Bi-CGSTAB and their preconditioners. In §3, we compare the convergence behavior of classical stationary and accelerated iterations on ideal elliptic and parabolic problems, particularly displaying their scaling on the size of the timestep. Though straightforward, the needed results do not seem to be documented elsewhere, so we devote relatively more detail to this section than to the surrounding sections. Model governing equations of an axisymmetric unconfined flame sheet are described in §4 in the vorticity-velocity formulation.

The governing equations are solved using a single Newton-based solver that, in turn, employs several different linear system modules. The results are presented in §5; we emphasize performance results as opposed to physical interpretations that are available in the references. One of the conclusions is that a properly ordered block-line GS makes an excellent preconditioner; however, GS is an expressly sequential algorithm. This raises the question of whether the Gauss-Seidel updates on selected lines of the grid can be replaced with Jacobi updates to create parallel threads of execution in the preconditioner. In §6, this is answered in the affirmative, for this problem and for a related problem involving 26 species engaged in 78 chemical reactions. Section 7 concludes with a summary of observations and extensions.

2. Linear solution algorithms. We consider a total of seven linear solvers: the classical block-line SOR method, and two acceleration methods, GMRES and Bi-CGSTAB, each with the same set of three preconditioners: block-line GS, symmetric block-line GS (SGS), and a block incomplete LU decomposition (ILU). The adjective "block" refers to the coupling of several (in our test problems, *four*) fields at each point, and "line" to the coupling of all the grid points in a row of the (tensor product) grids. In our test problem of predominantly unidirectional flow in a high aspect ratio domain, we group by lines normal to the main transport direction. We always sweep in the main flow direction, except in the case of the SGS preconditioner, which includes a return sweep against the main flow.

In this work, the matrices to be inverted arise from the discretization of second-order PDEs on two-dimensional tensor product grids in a symmetry plane of an axisymmetric domain described by radial and axial coordinates. If n_c is the number of dependent variables and n_r and n_z are the number of grid points in the r and z directions, respectively, the Jacobian matrix A has the following tridiagonal block structure

$$A = \begin{bmatrix} D_1 & U_1 & & \\ L_2 & D_2 & U_2 & \\ & \ddots & \ddots & \ddots \\ & \ddots & \ddots & U_{n_z-1} \\ & & L_{n_z} & D_{n_z} \end{bmatrix}, \tag{2.1}$$

where all the blocks D, L, and U are of size $(n_c n_r) \times (n_c n_r)$. These blocks in turn have a tridiagonal block structure consisting of $n_c \times n_c$ blocks arranged in n_r rows. When a nine-point finite difference stencil is employed for the discretization, the Jacobian must allow for $9n_c^2 n_r n_z$ nonzeros. For compact algorithmic descriptions below, we recast (2.1) in the form

$$A = L + D + U, \tag{2.2}$$

where L, D, and U are the block lower, diagonal, and upper parts of A, respectively.

2.1. Primitive iterations. For use either as stand alone or preconditioning methods, we consider splittings of relaxation and incomplete factorization type. The first class requires no storage in addition to the original matrix A, whereas the second nearly doubles the memory requirements of the code in the practically important limit of large n_c. For reasons described in §4, we consistently precondition on the left, transforming (1.1) to $(B^{-1}A)x = B^{-1}b$ by means of a nonsingular approximation B to A, whose inverse action is inexpensive to apply in the usual sense that its operation cost is proportional to the first power of the number of degrees of freedom in the system.

2.1.1. GS. The GS preconditioner is

$$B_{GS} = (L + D). \tag{2.3}$$

The preconditioned product $y = B_{GS}^{-1}Ax$ is formed in two steps: solve $(L + D)z = Ux$; $y \leftarrow z + x$. The product Ax is not explicitly computed, so the factors of the decomposition of the block diagonal matrix D can be stored in place in A. Note also that the vector z can be overwritten by y.

2.1.2. SOR. GS is a special case of the SOR algorithm

$$B_{\text{SOR}} = \left(L + \frac{1}{\omega}D\right), \tag{2.4}$$

with $\omega = 1$. The vector $y = B_{\text{SOR}}^{-1}Ax$ can be computed in three steps: compute $s = [(1 - \frac{1}{\omega})I + D^{-1}U]x$; solve $(\frac{1}{\omega}I + D^{-1}L)z = s$; $y \leftarrow z + x$. The name derives from the choice $1 < \omega < 2$, which can be selected to improve the convergence rate by a full asymptotic order over GS in model problems. When used as a stationary method in typical finite-difference combustion applications [1] it is usually necessary to *under* relax SOR to bring the spectral radius of the iteration matrix below one. Even $\omega = 1$ leads to catastrophic divergence for a significant fraction of the Jacobians encountered. One advantage of the acceleration method is its freedom from senstivity to relaxation parameters, which is difficult to optimize in these multicomponent, convective problems.

2.1.3. SGS. The SGS preconditioner is

$$B_{\text{SGS}} = (L + D)D^{-1}(U + D). \tag{2.5}$$

The following four-step process forms $y = B_{\text{SGS}}^{-1}Ax$: compute $s = -LD^{-1}Ux$; solve $(L + D)z = s$; solve $(I + D^{-1}U)y = z$; $y \leftarrow y + x$. The second and third steps are done

using a forward and backward substitution, respectively. Note that the LU decomposition of the matrix D can also be stored in place in the matrix A, but one more work vector is necessary with respect to the GS preconditioner. The principal advantages of SGS over GS, preserving symmetry when A is itself symmetric and propagating boundary information more isotropically in problems with negligible flow or significant flow recirculation, are not directly relevant to our test problem, but we include SGS in the tests to provide an indirect measure of the value of downstream influence in our strongly coupled system of elliptic boundary value problems.

2.1.4. ILU. ILU factorizations of the matrix A constitute an important class of preconditioning methods. An ILU factorization of A (of level 0) consists of a lower triangular matrix $\tilde{L}$ and an upper triangular matrix $\tilde{U}$ with the same nonzero pattern as the original matrix A satisfying

$$\tilde{L}\tilde{U} = A + E, \tag{2.6}$$

where E is the deviation matrix. The LU decomposition is obtained by a modified Gaussian elimination procedure where all the corrections in the matrices $\tilde{L}$ and $\tilde{U}$ are discarded if the corresponding entry in the matrix A is zero. With this procedure, no fill-in is introduced, but we still need to store the matrices $\tilde{L}$ and $\tilde{U}$, in addition to the original matrix A. We use a block ILU factorization at the level of the n_c dependent variables, i.e., dense Gaussian elimination is done below this level.

Because *all* of the matrix operations described in §2.1 are blocked at the level of the n_c degrees of freedom defined at each point, all of the solvers treat the source-term coupling (including such chemical reaction and such heat release as may be present) of the combustion process implicitly. This is algorithmically natural and nearly universal [14] since the $n_c \times n_c$ blocks are dense, and are of highly unpredictable diagonal dominance, depending upon which reactions are active at a given grid point. Only parts of the convection and diffusion operators are left explicit. The GS, SOR, and SGS methods are implemented in block-line form along radial lines, so that only the z-directional coupling is left explicit. The ILU preconditioner treats both r- and z-directional coupling implicitly, up to the level of fill permitted. The performance of low-fill ILU is sensitive to ordering effects, but less so than that of the block-line methods. A 20 to 30% difference in iteration count for flow-aligned versus cross-flow ILU orderings on a similar flame sheet problem is reported in [15].

2.2. Acceleration methods. The GMRES method [6] computes successive approximations to x^*, the solution of (1.1), in the affine space $x_0 + K_m$, where x_0 is an initial approximation to the solution. Upon introduction of the initial preconditioned residual $r_0 = B^{-1}(b - Ax_0)$, the Krylov subspace K_m is defined by $K_m = \text{span}\{r_0, B^{-1}Ar_0, \ldots, (B^{-1}A)^{m-1}r_0\}$, and, in practical implementation, is spanned by the orthonormal Arnoldi basis generated through a Gram–Schmidt process on the elements of K_m, requiring m inner products and one preconditioned matrix-vector product at iteration m. The approximate solution x_m at step m minimizes the residual over the Krylov subspace $x_0 + K_m$, which requires the storage of the full Arnoldi basis of size m. Since the maximal dimension of the Krylov subspace is held fixed in practice, the iteration generally needs to be restarted before achieving convergence.

In contrast to GMRES, Bi-CGSTAB [10] keeps the work and storage constant per iteration by defining the iterates by a Galerkin condition rather than by a minimization property. The algorithm terminates in a number of steps equal at most to the size of the problem; however, since there is no minimization property of the residuals in the intermediate iterations, breakdowns—more precisely, division by zero or very small numbers—may occur in the convergence process. In the Bi-CGSTAB algorithm, the iterates are constructed in such a

way that the residual r_j is orthogonal with respect to a sequence of vectors $\tilde{r}_0, \ldots, \tilde{r}_{j-1}$ and, similarly, the vector $\tilde{r}_j$ is made orthogonal to $r_0, \ldots, r_{j-1}$. The jth residual can be written $r_j = P_j(A)r_0$, where P_j is a monic polynomial of degree less or equal to j. The $\tilde{r}_j$ are generated with polynomials of the form $Q_j(x) = (1 - \omega_1 x) \ldots (1 - \omega_j x)$, where the ω_j are chosen so that

$$(P_i(A)r_0, Q_j(A^T)\tilde{r}_0) = 0, \qquad i \neq j. \tag{2.7}$$

In a practical implementation, condition (2.7) is enforced without explicit reference to A^T. Bi-CGSTAB requires two preconditioned matrix-vector products with A and four inner products per iteration. In the tabulated numerical experiments, we consistently chose the standard default $\tilde{r}_0$ equal to r_0. Based on a recent personal communication from H. van der Vorst, we repeated certain experiments with $\tilde{r}_0$ equal to $(B^{-1}A)^T r_0$ with mixed benefits; see §5.

3. The effect of the pseudotransient on the linear conditioning. Though relaxation-based solvers have largely been superseded by Krylov methods for linear elliptic problems, this section motivates their use in nonlinear elliptic problems aided by pseudotransient continuation. Model elliptic and parabolic Poisson problems in two dimensions are considered because of the convergence theory available for them. We present theoretical iteration count estimates for various methods on discrete operators corresponding to $-\nabla^2$ and $\partial/\partial t - \nabla^2$ in unpreconditioned or preconditioned form. We denote either of these symmetric positive definite operators as A, and write $A = B - R$, where B is the dominant part of A to be used as a splitting matrix or preconditioner, and R is the remainder. A simple conclusion drawn from the model problems is tested experimentally in §5 in the combustion context.

Convergence theorems for either stationary or accelerated iterative methods for general systems (1.1) are few in number and generally unsatisfactory with respect to either of two weaknesses: they are frequently pessimistic, or, if tight, they depend on hypotheses about the spectrum of A or the rank of $\alpha I - A$ (for some nonzero scalar α) whose verification is more computationally complex than the solution of (1.1), itself. Because of the lack of predictability of the convergence rate of iterative methods for nonsymmetric or indefinite systems, experience within a problem class is often more useful than theory in estimating computational work to solve a particular problem. A combination of theory for model problems (for the scaling laws), and experience within a problem class (for the constants), is much better than either guide alone in estimating the computational resources (time and memory) required to solve a problem that is in some sense "near" to the problem whose behavior is well understood.

3.1. Spectrum-based convergence results for symmetric problems. The number of iterations, k, required for convergence in the sense of

$$\frac{||x_k - x^*||_M}{||x_0 - x^*||_M} \leq \epsilon,$$

where x_0 is the initial and x_k the kth iterate, and $\epsilon > 0$ a given tolerance, can be estimated as follows: for stationary methods, we have for $M = I$ that

$$k(\epsilon) \approx \frac{\log(\epsilon)}{\log(\rho(I - B^{-1}A))}. \tag{3.1}$$

For a preconditioned CG method, we have for $M = B^{-1/2}AB^{-1/2}$ that

$$k(\epsilon) \approx \frac{\log(\epsilon/2)}{\log\left(\frac{\sqrt{\kappa(M)}-1}{\sqrt{\kappa(M)}+1}\right)}. \tag{3.2}$$

In these formulae, ρ and κ are the spectral radius and the condition number of their matrix arguments, respectively, which reduce for symmetric positive definite operators to $\rho(M) \equiv \lambda_{\max}(M)$ and $\kappa(M) \equiv \lambda_{\max}(M)/\lambda_{\min}(M)$. The spectra, in turn, depend upon the mesh spacing h, and the timestep size Δt. To comprehensively compare the effectiveness of stationary and accelerated methods in pseudotransient problems, $k(\epsilon, h, \Delta t)$ should be examined as a function of h and Δt at a tolerance that tightens according to the truncation error, e.g., $\epsilon \sim h^2$ for second-order discretizations. Here we compare several methods on a fixed grid with a fixed tolerance as Δt varies.

We consider the SGS and ILU(0) preconditioners, which can be split symmetrically around the model symmetric A defined above, but which may also be defined as one-sided preconditioners in the nonsymmetric case. As baseline cases, we also consider unpreconditioned CG iteration and the stationary GS iteration. To examine the behavior of (3.1) and (3.2) we need spectral data that can be obtained experimentally from applying sparse eigensolvers to large-dimensional discrete systems or estimated at virtually no expense from Fourier techniques. The techniques we employ were developed in [16] and subsequently applied in [17] and [18]. In [16], it is argued that the distribution of the spectrum of the periodic boundary condition problem of mesh interval h predicts the distribution of the spectrum of the Dirichlet problem of mesh interval $2h$, provided that the zero eigenvalue (corresponding to the constant eigenvector), and the next pair of eigenvalues (corresponding to the univariate modes), are removed from the bottom of the periodic spectra. In the case of periodic boundary conditions, A and the various preconditioners B all share the same set of eigenvectors, and hence the spectra $\lambda(I - B^{-1}A)$, and $\lambda(B^{-1/2}AB^{-1/2})$ are trivially computed from the individual spectra $\lambda(A)$ and $\lambda(B)$.

3.2. The unpreconditioned periodic problem. Slightly extending [16] in notation that conforms to it where possible, consider $A = (h^2/\Delta t)I - \nabla_h^2$, where $-\nabla_h^2$ is the usual uniform grid five-point operator scaled to have four on the diagonal on a periodic square domain of $n + 1$ grid points on a side. Because of the periodicity, the number of cells is the same as the number of points, i.e., the mesh interval h is $1/(n + 1)$. We take the coordinate indices in the standard range $0 \le j, k \le n$, interpreting indices outside this range in terms of their modulus with respect to n, so that a row of $Au = f$ becomes

$$\left(4 + \frac{h^2}{\Delta t}\right) u_{jk} - u_{j-1,k} - u_{j+1,k} - u_{j,k-1} - u_{j,k+1} = h^2 f_{jk}. \tag{3.3}$$

The $(n + 1)^2$ eigenvectors are

$$u_{jk}^{(s,t)} = e^{i(j\theta_s + k\phi_t)}, \quad 0 \le s, t \le n,$$

where $\theta_j \equiv j2\pi h$ and $\phi_k \equiv k2\pi h$. Substituting this expression into (3.3), the corresponding eigenvalues are readily determined:

$$\begin{aligned}
\lambda_{st}(A) &= 4(1 + \gamma^2) - e^{-i\theta_s} - e^{i\theta_s} - e^{-i\phi_t} - e^{i\phi_t} \\
&= 4(1 + \gamma^2) - 2(\cos\theta_s + \cos\phi_t) \\
&= \frac{h^2}{\Delta t} + 4 - 2(\cos\theta_s + \cos\phi_t) \\
&= \frac{h^2}{\Delta t} + 4\left(\sin^2\frac{\theta_s}{2} + \sin^2\frac{\phi_t}{2}\right),
\end{aligned}$$

where $\gamma^2 \equiv h^2/(4\Delta t)$. The largest eigenvalue is $\lambda_{\max} \approx h^2/\Delta t + 8$ at $\theta_s = \phi_t \approx \pi$ and the smallest, excluding $s = 0$ or $t = 0$, is $\lambda_{\min} = h^2/\Delta t + 8\sin^2(\pi h) \approx h^2/\Delta t + 8\pi^2 h^2$.

The condition number is therefore

$$\kappa(A) \approx \frac{h^2/\Delta t + 8}{h^2/\Delta t + 8\pi^2 h^2},$$

from whence it follows that $\lim_{\Delta t \to 0} \kappa(A) = 1$ and $\lim_{\Delta t \to \infty} \kappa(A) \approx (\pi h)^{-2}$. Values of $\kappa(A)$ between these saturation limits are plotted as a function of Δt at $h^{-1} = 64$, 128, and 256 as the solid curves in Fig. 1(a). For a Dirichlet problem, the Poisson problem limit would be the classical $\kappa(A) \approx (\pi(h/2))^{-2}$, justifying in this instance the claim that the periodic case condition number predicts the Dirichlet case condition number of a grid twice as coarse. To apply to the Dirichlet problem, the curves in Fig. 1(a) should be labelled $h^{-1} = 32$, 64, and 128.

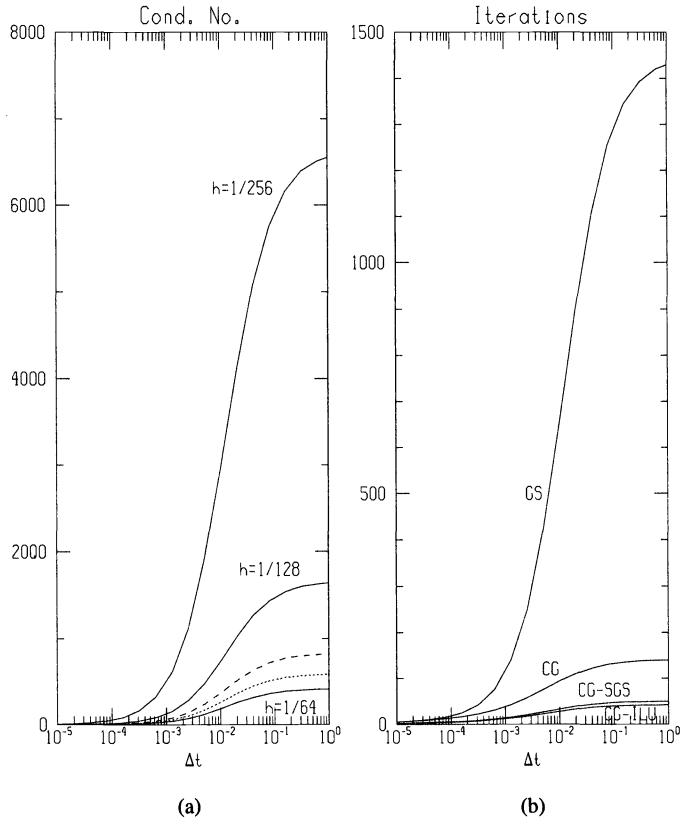


FIG. 1. (a) *Condition number of the parabolic Poisson problem with periodic boundary conditions for the unpreconditioned case (solid curves) at three different h, and for the* SGS-*preconditioned (dashed curve) and* ILU-*preconditioned (dotted curve) cases for* $h^{-1} = 256$, *as a function of* Δt. (b) *Iteration count estimates for the parabolic Poisson problem on a grid with* $h^{-1} = 64$ *with a convergence tolerance on the error of* 10^{-6} *in norms convenient to particular accelerator/preconditioner combinations for stationary* GS, *unpreconditioned* CG, CG-SGS, *and* CG-ILU, *as a function of* Δt.

Note that if Δt is of the order of h^2, $\lambda_{\min}$ becomes independent of h to leading order, and all methods converge in a number of iterations independent of h. (Of course, in this case, many outer timesteps will be necessary.) If Δt is of the order of h, $\lambda_{\min}$ goes as h to leading order. Finally, if Δt is large, independent of h, $\lambda_{\min}$ behaves like h^2. In the course of the numerical results reported in §5, Δt sweeps through the entire range of $\mathcal{O}(h^2)$ to $\mathcal{O}(1)$, governed locally

by nonlinear convergence criteria. In this and many other practical problems, therefore, the preferred linear solver may also vary over the course of the nonlinear trajectory.

3.3. The SGS preconditioned periodic problem. From the additive decomposition $A = L + D + U$ and $B_{\mathrm{SGS}} = (L + D)D^{-1}(U + D)$, we derive upon substitution of the $u_{jk}^{(s,t)}$ into $B_{\mathrm{SGS}}u = \lambda(B_{\mathrm{SGS}})u$ that

$$\begin{aligned}\lambda_{st}(B_{\mathrm{SGS}}) &= [4(1+\gamma^2) - e^{-i\theta_s} - e^{-i\phi_t}] \cdot \frac{1}{4(1+\gamma^2)} \cdot [4(1+\gamma^2) - e^{i\theta_s} - e^{i\phi_t}] \\ &= 4(1+\gamma^2) - 2(\cos\theta_s + \cos\phi_t) + \frac{1+\cos(\theta_s - \phi_t)}{2(1+\gamma^2)}.\end{aligned}$$

Since B_{SGS} and A are simultaneously diagonalized, we can easily evaluate

$$\lambda_{st}(B_{\mathrm{SGS}}^{-1/2} A B_{\mathrm{SGS}}^{-1/2}) = \lambda_{st}(B_{\mathrm{SGS}}^{-1}A) = \frac{4(1+\gamma^2) - 2(\cos\theta_s + \cos\phi_t)}{4(1+\gamma^2) - 2(\cos\theta_s + \cos\phi_t) + \frac{1+\cos(\theta_s-\phi_t)}{2(1+\gamma^2)}}.$$

This expression can be explored analytically for its maximum and minimum in (s, t), but it is simplest for present purposes to extract the condition number by looping over all eigenpairs. A plot of the condition number of $B_{\mathrm{SGS}}^{-1}A$ over a range of Δt at $h^{-1} = 256$ is given as the dashed curve in Fig. 1(a).

3.4. The ILU preconditioned periodic problem. For a symmetric problem, we write $B_{\mathrm{ILU}(0)}$ as the product of a lower triangular factor, a diagonal factor, and the transpose of the lower triangular factor, and we require that the elements of the product match the elements of the original where possible. Thus,

$$A = \tilde{L}\tilde{D}^{-1}\tilde{L}^T - E,$$

where E is the deviation matrix. For the constant coefficient operator in question, $\tilde{L}$ has 1's in the same off-diagonal locations as A and a to-be-determined d on the diagonal, and $\tilde{D}$ is a diagonal matrix of all d's. Multiplying out and equating to A yields

$$d + \frac{2}{d} = 4(1+\gamma^2)$$

as the equation for d, whence $d = 2[(1+\gamma^2) + \sqrt{(1+\gamma^2)^2 - 1/2}]$. This leads straightforwardly to

$$\begin{aligned}\lambda_{st}(B_{\mathrm{ILU}}) &= [d - e^{-i\theta_s} - e^{-i\phi_t}] \cdot \frac{1}{d} \cdot [d - e^{i\theta_s} - e^{i\phi_t}] \\ &= 4(1+\gamma^2) - 2(\cos\theta_s + \cos\phi_t) + \frac{\cos(\theta_s - \phi_t)}{(1+\gamma^2) + \sqrt{(1+\gamma^2)^2 - 1/2}}.\end{aligned}$$

As above, we can easily evaluate

$$\lambda_{st}(B_{\mathrm{ILU}}^{-1}A) = \frac{4(1+\gamma^2) - 2(\cos\theta_s + \cos\phi_t)}{4(1+\gamma^2) - 2(\cos\theta_s + \cos\phi_t) + \frac{\cos(\theta_s-\phi_t)}{(1+\gamma^2)+\sqrt{(1+\gamma^2)^2-1/2}}}.$$

A plot of the condition number of $B_{\mathrm{ILU}}^{-1}A$ over a range of Δt at $h^{-1} = 256$ is given as the dotted curve in Fig. 1(a).

The condition number data of the previous three subsections is employed in (3.2) to generate the curves labelled CG in Fig. 1(b), which predicts iteration counts for a fixed reduction in a convenient preconditioned energy norm of the error over a range of Δt at a fixed grid size of $h^{-1} = 64$.

3.5. GS stationary iterations on the periodic problem. The curve labelled GS in Fig. 1(b) is plotted from (3.1) on the basis of the following derivation for $|\lambda(I - B_{GS}^{-1}A)| = |\lambda(B_{GS}^{-1}R)|$ in the symmetric case for which $A = L + D + L^T$ and $B_{GS} = D + L$.

$$\lambda_{st}(I - B_{GS}^{-1}A) = \lambda_{st}((D+L)^{-1}L^T) = \frac{-e^{i\theta_s} - e^{i\phi_t}}{4(1+\gamma^2) - e^{-i\theta_s} - e^{-i\phi_t}}.$$

Multiplying by the complex conjugate,

$$|\lambda_{st}(I - B_{GS}^{-1}A)|^2 = \frac{1 + \cos(\theta_s - \phi_t)}{8(1+\gamma^2)^2 - 4(1+\gamma^2)(\cos\theta_s + \cos\phi_t) + 1 + \cos(\theta_s - \phi_t)},$$

whence $\rho(I - B_{GS}^{-1}A)$ is easily found.

In comparing different curves in Fig. 1(b), it must be borne in mind that changing the method or preconditioner changes the norm in which the error is measured; thus, one should not attempt to infer in detail the cost to arrive at a particular solution, but only the trends. These plots indicate that if Δt is constrained to be small by nonlinear considerations, ordinary relaxation will be competitive.

4. Test problems. In this section, we briefly describe the flame sheet problem used to evaluate the performance of the various linear algebra solvers discussed in §2. The numerical solution of a flame sheet model presents a twofold interest. First, as mentioned above, flame sheet problems are on the natural route to the numerical solution of important multidimensional diffusion flames. Second, the governing equations are highly nonlinear and the dependent variables are strongly coupled in the physical domain and on its boundaries so that their solution constitutes a formidable test for nonlinear elliptic solvers.

The flame sheet governing equations consist of the conservation of total mass, momentum, and a conserved scalar equation. The temperature and major species profiles are recovered from the conserved scalar, as described in [1]. The flow field equations are formulated in terms of velocity and vorticity coupled together with a conserved scalar equation. As motivated in [19], a vorticity-velocity formulation allows replacement of the first-order continuity equation with additional second-order equations. Whereas the streamfunction-vorticity formulation also accomplishes the same replacement in two dimensions, vorticity-velocity is extensible to three and allows more accurate formulation of boundary conditions in a numerically compact way. Furthermore, convective terms in off-diagonal blocks that exert a strong influence in a streamfunction-vorticity formulation disappear.

The governing equations prior to nondimensionalization are

$$\nabla^2 v_r = \frac{\partial\omega}{\partial z} - \frac{1}{r}\frac{\partial v_r}{\partial r} + \frac{v_r}{r^2} - \frac{\partial}{\partial r}\left(\frac{v\cdot\nabla\rho}{\rho}\right), \tag{4.1}$$

$$\nabla^2 v_z = -\frac{\partial\omega}{\partial r} - \frac{1}{r}\frac{\partial v_r}{\partial z} - \frac{\partial}{\partial z}\left(\frac{v\cdot\nabla\rho}{\rho}\right), \tag{4.2}$$

$$\frac{1}{r^2}\nabla\cdot\left(r^3\nabla\frac{\mu\omega}{r}\right) = \rho v_r\frac{\partial\omega}{\partial r} + \rho v_z\frac{\partial\omega}{\partial z} - \frac{\rho v_r}{r}\omega + \overline{\nabla}\rho\cdot\nabla\frac{v^2}{2} - \overline{\nabla}\rho\cdot g, \tag{4.3}$$

$$\frac{1}{r}\nabla\cdot\left(r\rho D\nabla S\right) = \rho v_r\frac{\partial S}{\partial r} + \rho v_z\frac{\partial S}{\partial z}, \tag{4.4}$$

where $v = (v_r, v_z)$ is the velocity vector with radial and axial components v_r and v_z, respectively, ρ the density, $\omega = \partial v_r/\partial z - \partial v_z/\partial r$ the normal component of the vorticity, g the gravity vector, μ the viscosity, S the conserved scalar, D a diffusion coefficient, ∇^2 a shorthand for $\partial^2/\partial r^2 + \partial^2/\partial z^2$ (not the full cylindrical Laplacian), $\nabla\cdot(\alpha_r, \alpha_z)$ is $\partial\alpha_r/\partial r + \partial\alpha_z/\partial z$, and the

components of $\overline{\nabla}\beta$ are $(\partial\beta/\partial z, -(\partial\beta/\partial r))$. The diffusion coefficient D and the viscosity μ depend on the temperature through a power law [1]. In our computations, typical values for the Prandtl and Reynolds numbers were 0.75 and 160, respectively. As boundary conditions on an open domain with entrainment permitted, we employ along the axis of symmetry ($r = 0$)

$$v_r = 0, \quad \frac{\partial v_z}{\partial r} = 0, \quad \omega = 0, \quad \frac{\partial S}{\partial r} = 0, \tag{4.5}$$

at the exit ($z \to \infty$)

$$v_r = 0, \quad \frac{\partial v_z}{\partial z} = 0, \quad \frac{\partial \omega}{\partial z} = 0, \quad \frac{\partial S}{\partial z} = 0, \tag{4.6}$$

along the inlet ($z = 0$)

$$v_r = 0, \quad v_z = v_z^0, \quad \omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r}, \quad S = S^0, \tag{4.7}$$

and along the entrainment boundary ($r = R_0$)

$$\frac{\partial v_r}{\partial r} = 0, \quad \frac{\partial v_z}{\partial r} = 0, \quad \frac{\partial \omega}{\partial r} = 0, \quad S = 0. \tag{4.8}$$

The inlet consists of an inner cylindrical fuel jet and an outer coflowing annular oxidizer jet. A schematic of this simple, extensively experimentally investigated configuration is given in Fig. 2. The inlet profile of the conserved scalar, $S^0(r)$, is a slightly rounded step function that blends room temperature reservoirs of fuel and oxidizer by means of a narrow Gaussian in temperature centered at R_I (see Fig. 2).

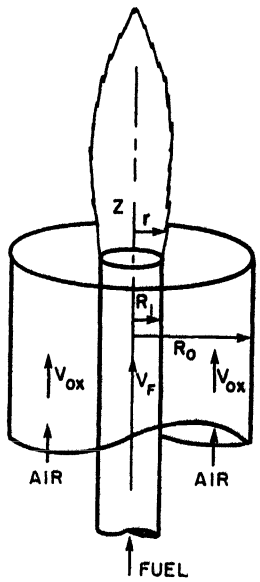


FIG. 2. *Schematic of the physical configuration.*

The PDEs (4.1)–(4.4) together with the boundary conditions (4.5)–(4.8) are discretized on a two-dimensional tensor-product grid. A solution is first obtained on an initial coarse grid.

Additional mesh points are then inserted and the coarse grid solution is interpolated onto the finer grid to yield a new solution starting estimate. A modified damped Newton iteration

$$\Delta U^n = -\lambda^n J(U^n)^{-1} F(U^n), \qquad n = 0, 1, \ldots, \tag{4.9}$$

with convergence tolerance $\|\Delta U^n\|_2 < 10^{-5}$, is used to solve the set of discretized equations $F(U) = 0$. The Jacobian is inverted at each Newton step through an inner iteration. It is important to base convergence of the linear iterative method as nearly as possible on the magnitude of $\|\Delta U\|_2$, which controls the outer Newton iteration. If our linear convergence criterion is too loose or too stringent, we may suffer divergence of the outer iteration or excessive CPU time, respectively. Due to the ill conditioning of the Jacobian, the norm of the linear residual may differ by orders of magnitude from the norm of the correction ΔU. In the Krylov methods, we base convergence on the norm of the left-preconditioned residual, since it scales as $\|\Delta U\|_2$. In the SOR method, the magnitude of the update to the Newton correction is already conveniently available.

All of the results presented below are obtained using an absolute linear convergence tolerance equal to one-tenth of the Newton tolerance. By this procedure, excessive CPU consumption during the pseudotransient phase is avoided while still bringing back sufficient precision in the Newton correction. To illustrate the difficulties encountered when using the right-preconditioned residual as the convergence criterion, we reran selected cases with the same absolute convergence tolerance applied to the right-preconditioned residual. For instance, the coarse-grid Bi-CGSTAB/GS run required 64% more CPU time when right-preconditioned convergence was based on the same absolute tolerance as the left-preconditioned method. The Newton convergence history was virtually unchanged. This unnecessarily stringent criterion could be partially alleviated with a mixed absolute-relative tolerance allowing termination at the earlier of the absolute criterion based on the Newton tolerance *or* a relative reduction of 10^6 in linear residual. The same test required only 33% more CPU time when based on the mixed tolerance.

Due to the nonlinearity of the original problem, a pseudotransient continuation process is used to produce a parabolic-in-time problem and bring the starting estimate on a given grid into the convergence domain of the steady Newton method. The original nonlinear elliptic problem is cast into a parabolic form by appending a pseudotransient term $\partial U/\partial t$ to the original set of algebraic equations $F(U) = 0$, and a fully implicit scheme solves (again with Newton's method)

$$\mathcal{F}(U^{n+1}) = F(U^{n+1}) + \frac{U^{n+1} - U^n}{\Delta t^{n+1}} = 0, \tag{4.10}$$

where Δt^{n+1} is the $(n+1)$st timestep. The timestep is adaptively increased as the convergence domain of the steady Newton method is approached [20] so that the solution process on a given grid may be conceived as having three main stages: an initial "deep" transient phase, a "medium" transient phase, and the final steady Newton iteration. The boundary between the first two phases is blurred by the gradual and not necessarily monotonic buildup of Δt. Typically, in driving detailed-chemistry flame solutions towards grid independence, an outer iteration in which the grid is adapted with respect to truncation error estimates is wrapped around each pseudotransient-to-steady-state cycle. Asymptotically, as the mesh spacing approaches zero, the interpolant of the converged solution on one grid lies in the convergence domain of Newton's method on the next finer grid [21].

5. Numerical results. In this section, we present several numerical experiments obtained on a Multiflow Trace 14/300 computer for the solution of the test problem described in the previous section. We compared three categories of methods for solving the linear systems

arising at each Newton step: a stationary method (block-line SOR with a relaxation parameter initially set at 0.85, adjusted downward as necessary), a short-term recurrence Krylov (Bi-CGSTAB) and an optimal Krylov (GMRES). Due to memory constraints, GMRES was actually run in a nonoptimal restarted mode with 20 Arnoldi vectors. The choice of 20, justified below, represents a balance of convergence rate and memory considerations. Both Krylov methods were combined with the three different left preconditioners: block-line GS, block-line SGS, and block-point ILU. We used GS as a preconditioner rather than the somewhat more costly SOR because of our experience that relaxation methods are much less sensitive to ω as preconditioners than they are as stationary iterative methods, as also reported in [22]. In addition to varying the iterative method/preconditioner combination, we considered a coarse 41×41 and a fine 81×81 tensor-product grid. In presenting the results, we distinguish between early timesteps of small size and later larger timesteps to observe performance variations in the parabolic and elliptic limits.

Coarse and fine grids adapted to the coflow geometry are shown in Fig. 3, and the converged solution for a methane-air flame sheet is shown in Fig. 4. The axisymmetric flow is locally tangent to the streamfunction contours, and the mass flux it carries in a circumferential ring is inversely proportional to the local contour spacing. The failure of the contours to close on the open outer boundary is evidence of entrainment of ambient oxidizer. In engineering practice, the outer boundary condition of a zero gradient on the vorticity may be replaced with its definition in terms of the velocity derivatives, which we have observed to yield less entrainment of fluid across the outer boundary. The temperature field plot in Fig. 4(b), obtained from the solution for the conserved scalar S as described in [1], varies from an ambient temperature of 298°K to approximately 2000°K in a distance of millimeters. Interpretative discussions of similar results for flame sheet, reduced kinetics, and full kinetics flame models may also be found in [1], [2], and references therein.

5.1. General comparisons. The algorithmic performance results obtained on the coarse 41×41 grid are summarized in Table 1. We found that a minimum of 850 adaptively increased timesteps had to be taken to bring the initial estimate into the convergence domain of the steady Newton method. The initial timestep was set at 10^{-8} and was allowed to increase adaptively [20]. By the time a switchover to the steady-state form of the equations could be undertaken, Δt had typically reached $O(1)$. The first line of Table 1 presents the memory requirement in double-precision words for each method. The extra overhead of the Krylov space is reflected in the GMRES columns, and it is seen that the use of an ILU preconditioner approximately doubles the storage requirement with respect to the standard relaxation method. The second line shows the CPU time needed per inner linear iteration averaged along the pseudotransient iterations; these numbers are normalized by the corresponding SOR time in line 3. Lines four–six show the total amount of CPU time spent on the linear algebra solves for the first 5, 10, and 20 timesteps, respectively. At this stage of the transient iteration, the timestep is small so that a relaxation method is quite competitive with a Krylov method preconditioned with GS, whereas the use of other preconditioners (SGS or ILU) tends to be more expensive.

It is clear from the last three lines of Table 1, representing the total CPU time spent on the pseudotransient iteration, the steady Newton method, and the whole algorithm, respectively, that the best overall results are obtained by combining either Bi-CGSTAB or GMRES(20) with the GS preconditioner and, from a storage point of view, the former algorithm is preferable. Relaxation methods meet with difficulty when the timestep is increased and are ineffective in the fully elliptic phase of the iteration. The asterisks in the SOR column indicate that linear convergence on the steady-state system was eventually achieved with a relaxation factor ω of 0.3, instead of the previously sufficient 0.85. The combination of GMRES(20) with SGS stagnates due to loss of orthogonality in the Krylov vectors.

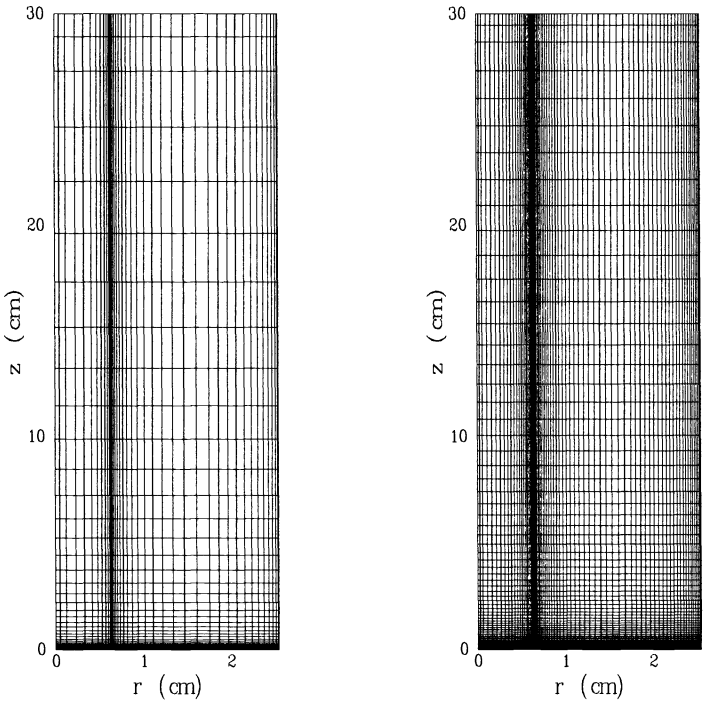


FIG. 3. *The grids.* (a) *a coarse grid, showing concentration of mesh lines in the high-activity regions of the flow domain and* (b) *a self-similar fine grid.*

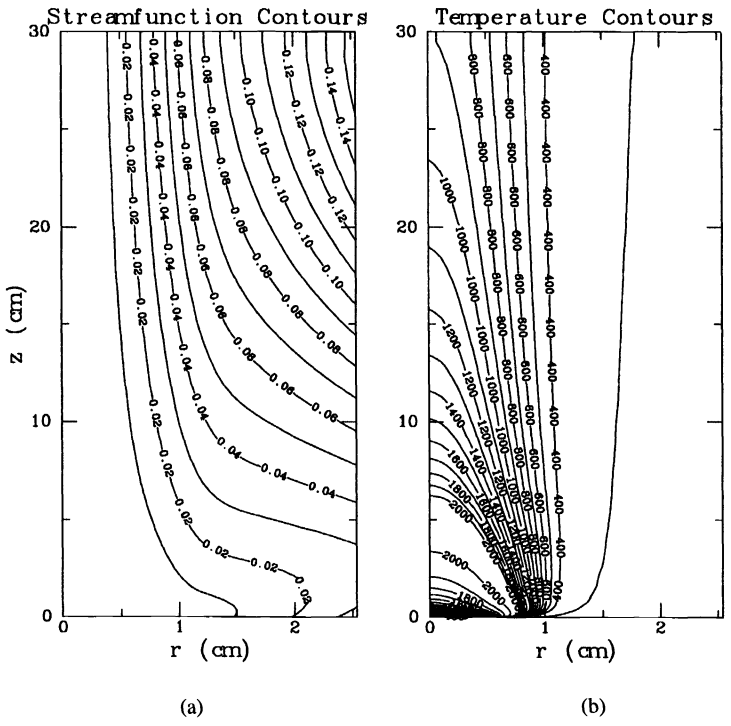


FIG. 4. *The converged solution.* (a) *variable-density Stokes streamfunction and* (b) *temperature.*

The use of an ILU preconditioner is not very attractive for these problems because, in the coarse grid case, most of the CPU time is spent in the pseudotransient continuation process to bring the iterate into the convergence domain of the steady Newton method. The decrease of inner iterations needed to invert the Jacobian matrix with either the SGS or ILU preconditioners is not sufficient to compensate for their higher cost per iteration. This results in an increase of the overall CPU time by a factor of 1.4 and 2.1 for SGS and ILU, respectively. Finally, the use of ILU preconditioning nearly doubles the memory requirements in problems already dominated by Jacobian storage. In detailed chemistry combustion models, where the dominant term in the storage, due to the Jacobian, scales like n_c^2, available memory limits the choice of n_c, which may be as high as 40 in current practice [1].

TABLE 1
Performance results on the coarse grid.

Algorithm	SOR	Bi-CGSTAB			GMRES(20)		
Preconditioner	–	GS	SGS	ILU	GS	SGS	ILU
work array (Mw)	0.31	0.35	0.35	0.60	0.45	0.46	0.70
time/it (min) $*10^{-3}$	2.83	6.56	16.6	26.6	4.27	11.7	20.7
time/SOR-time	1.0	2.3	5.9	9.4	1.5	4.1	7.3
time/5 steps (min)	0.13	0.09	0.21	0.33	0.09	0.18	0.32
time/10 steps (min)	0.25	0.20	0.40	0.62	0.18	0.35	0.62
time/20 steps (min)	0.48	0.42	0.78	1.22	0.36	0.73	1.20
time/transient (min)	156	121	162	246	117	154	243
time/steady (min)	94*	9	18	24	9	–	28
Total time (min)	250*	130	180	270	126	–	271

We discuss next the results obtained on the finer 81×81 tensor-product grid. Although the interpolated coarse grid solution lay in the domain of convergence of the steady Newton method on the finer grid, we found it more efficient overall to take several timesteps before starting the steady Newton iteration. After timesteps are taken, the Jacobian matrices at the first and second Newton steps are less expensive to invert so that an optimal balance exists between the CPU time spent in the parabolic and elliptic phases. In our computations, ten timesteps between 10^{-6} and 10^{-5} represented an excellent compromise. Numerical results are summarized in Table 2 using the same notation as in Table 1. Once again, the shortest overall execution time is obtained by using a Krylov method preconditioned with GS, and a short-term recurrence may be then preferable, rather than an optimal method, from a storage point of view. The relaxation method compares very favorably during the deep transient steps, but is unable to solve the elliptic linear systems with the original relaxation factor of 0.85. The asterisked execution times were obtained with a reduced factor of 0.65. With intermediate values of 0.75 and 0.70 for ω, SOR was able to take only the first steady Newton step and the first two steady Newton steps, respectively. As in the coarse grid case, the use of either SGS or ILU preconditioning does not present any advantage and it is interesting to note that GMRES(20)/ILU even stagnates as opposed to the much simpler GS and SGS preconditioners. The convergence history of all the above Krylov methods is illustrated in Fig. 5 for both grid cases by representing the residual norm versus CPU time during the inner iterations for the first steady Newton step. The characteristic staircase pattern of restarted GMRES and the highly nonmonotonic convergence history of Bi-CGSTAB are evident.

We have observed above that 20 Krylov vectors are not sufficient to prevent GMRES from stagnating in certain steady Newton cases. Our uniform choice of 20 vectors in the reported results stems from numerical experiments with as few as 5 and as many as 30 vectors, using the GS preconditioning. With Krylov spaces of dimensions 5 and 10, even the coarse grid problems stagnated. The same difficulty was encountered on the fine grid with dimension 15. In the other limit, allowing as many as 30 vectors led to fewer restart cycles during the

TABLE 2
Performance results on the fine grid.

Algorithm	SOR	Bi-CGSTAB			GMRES(20)		
Preconditioner	–	GS	SGS	ILU	GS	SGS	ILU
work array (Mw)	1.22	1.35	1.38	2.32	1.77	1.80	2.74
time/it (min) $*10^{-2}$	1.17	2.88	6.47	10.6	1.70	5.25	8.38
time/SOR-time	1.0	2.5	5.5	9.1	1.5	4.5	7.2
time/5 steps (min)	1.04	0.93	1.63	2.05	0.84	1.78	1.96
time/10 it (min)	1.83	1.64	2.92	3.63	1.46	3.40	3.53
time/transient (min)	4.1	3.5	5.0	8.3	3.5	5.8	8.3
time/steady (min)	104.9*	25.0	49.1	59.3	26.3	48.6	–
Total time (min)	109.0*	28.5	54.1	67.6	29.8	54.4	–

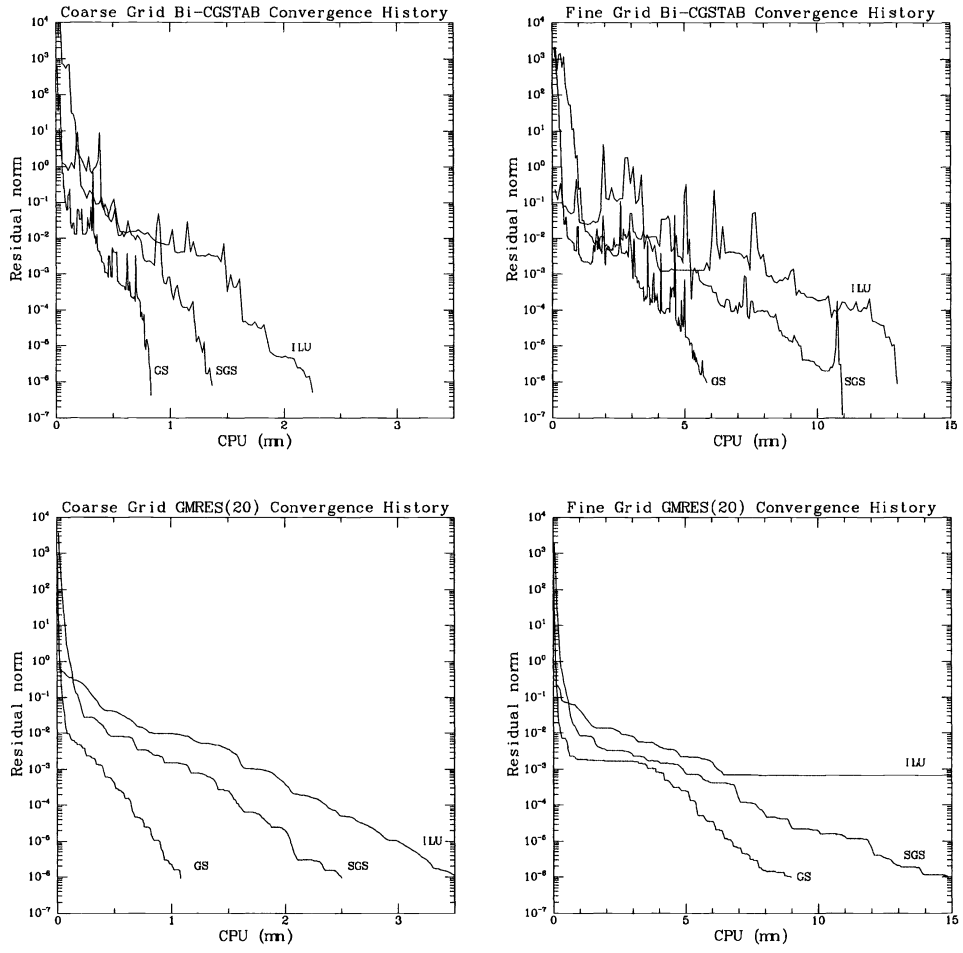


FIG. 5. *Plots of preconditioned residual norm as a function of execution time for* Bi-CGSTAB *and* GMRES(20) *using each of three preconditioners on coarse and fine grids for the first steady Newton step* ($\Delta t = \infty$).

steady Newton iteration with overall CPU savings of 26% and 2% on fine and coarse grids respectively, but at a cost of approximately 15% more in storage. It therefore appears that

special care should be taken in selecting the Krylov dimension to avoid both stagnation in the steady Newton iteration and excessive storage requirements in the pseudotransient stage.

Following a suggestion of Van der Vorst, we reran the Bi-CGSTAB/GS cases on both coarse and fine grids with $\tilde{r}_0 = (B_{GS}^{-1}A)^T r_0$ to attempt to observe a decrease in iterations often associated with this practice. We found decreases in iterations relative to $\tilde{r}_0 = r_0$ in the steady-state Newton iteration for both coarse and fine grids, the savings in CPU time being as high as 18%. In contrast, the convergence improvements were marginal and the execution times greater when this device was employed in the pseudotransient phase. The associated CPU time penalty was as high as 18%. This is mainly attributable to the extra cost of forming and factoring a second set of diagonal blocks of the Jacobian for use in the application of the transpose of the preconditioner. Their cost is not amortized over many subsequent iterations since the pseudotransient linear systems converge fairly rapidly.

We should also mention that the use of the flexible variant of GMRES, namely FGMRES [7], preconditioned alternatively by lower and upper triangles, did not meet with success. In fact, the alternate use of these two preconditioners seemed to uncondition the system for this test problem and the results were much worse than with a lower triangle preconditioner (GS) alone. Finally, we point out that for flame sheet problems, the cost of evaluating the Jacobian does not represent a significant number of linear iterations, and the best overall strategy for these problems is to re-evaluate the Jacobian at each timestep. For instance, in the fine grid case, the evaluation of the Jacobian takes as much time as 4.7 Bi-CGSTAB/GS iterations. Consequently, the savings on Jacobian evaluations from keeping the Jacobian fixed for several timesteps does not compensate for the resulting increase in the number of linear iterations. This contrasts with our experience in larger combustion systems, such as a 29-species methane-air flame, in which a Jacobian evaluation costs the equivalent of 14 Bi-CGSTAB iterations, so that it is preferable to keep it fixed for several timesteps, especially during the initial deep transient phase.

5.2. Detailed comparisons of Bi-CGSTAB/GS and GMRES/GS. According to the results of the previous section, the two most effective algorithms over the course of the entire solution history are Bi-CGSTAB/GS and GMRES(20)/GS. These algorithms never fail to converge and deliver total CPU times that are within 2% of each other on the two different grids. Bi-CGSTAB accomplishes this with approximately 30% less memory than does GMRES(20).

Table 3 breaks down the counts of Newton steps and Krylov iterations and the CPU time over each of four easily demarcated phases of the overall two-grid computation. As summarized in the left-most columns, these phases consist of: 850 adaptively chosen transient Newton steps on a coarse grid of 40 subintervals on each side, followed by a steady Newton iteration on the same grid, followed by interpolation onto a grid formed by doubling the refinement in each coordinate direction and 10 adaptively chosen timesteps on this new grid, followed by a steady-state iteration on the fine grid. The count of Newton iterations includes all of the correction vectors ΔU that are accepted by the algorithm, sometimes after damping. (Because of the control structure of the nonlinear algorithm, this is not the same thing as the total number of linear systems (4.9) of size $n_c \cdot n_r \cdot n_z$ solved. Many of these Newton corrections are provisional "look ahead" vectors that are ultimately discarded.) The counts of preconditioned Krylov iterations (of either Bi-CGSTAB or GMRES type) are the totals over all calls to the linear solver. This is the same as the number of calls to the subroutine that evaluates matrix-vector products for GMRES; for Bi-CGSTAB, the matrix-vector subroutine is called twice in each iteration. Likewise, the subroutine that evaluates vector inner products is called four times for each tabulated Bi-CGSTAB iteration. It is not possible directly to infer the number of inner products from number of GMRES iterations because of the ever-expanding space of direction vectors against which GMRES orthogonalizes each iterate. However, each

full restart cycle of 20 Krylov vectors requires $20 * 21/2 = 210$ inner products, so we can upper bound the number of inner products per GMRES iteration at 10.5.

TABLE 3
Performance of Bi-CGSTAB/GS *and* GMRES/GS *by grid and timestepping phases.*

Problem		Bi-CGSTAB/GS	GMRES/GS
Coarse grid transient	Newton its.	1,662	1,664
	Krylov its.	12,550	19,661
	Time (min)	120.5	116.7
Coarse grid steady	Newton its.	7	7
	Krylov its.	1,595	2,724
	Time (min)	9.4	9.2
Fine grid transient	Newton its.	14	14
	Krylov its.	57	86
	Time (min)	3.5	3.5
Fine grid steady	Newton its.	4	4
	Krylov its.	1,102	2,006
	Time (min)	25.0	26.3
Total	Newton its.	1,687	1,689
	Krylov its.	15,304	24,477
	Time (min)	158.4	155.7

It would appear that the nonlinear convergence history is independent of the linear solver, since the total number of Newton steps is nearly identical. In fact, a detailed iteration-by-iteration comparison reveals this *not quite* to be the case. The convergence tolerance for the Krylov methods is loose enough to allow slightly differently shaped correction vectors (whose linear residuals satisfy the same Euclidean bound) to be returned to the Newton method. Appreciable discrepancies (on the order of a factor of two) in nonlinear residual norm between the Bi-CGSTAB- and GMRES-driven solvers occur only in the last few Newton iterations, after the norm has dropped a millionfold. However, the number of Newton iterations to complete convergence is not affected.

In the transient portions of the solution trajectory, when the diagonal dominance of the Jacobian is enhanced by the timestepping, the number of Krylov vectors required per Newton step is one to two orders of magnitude less than the number required during the steady-state portions for both Bi-CGSTAB and for GMRES. In the transient fine-grid computation, for instance, Bi-CGSTAB requires approximately 4.1 iterations per Newton step, and GMRES approximately 6.1. On the other hand, in the steady coarse-grid computation, Bi-CGSTAB requires approximately 276 iterations per Newton step, and GMRES approximately 502. (Because "look-ahead" steps are frequently discarded during the steady Newton stage of the computation, the average numbers of Bi-CGSTAB and GMRES iterations per linear system solved are closer to half these totals.)

For a fixed state of the physical system, the Jacobian constructed on the fine grid is more dominated by the elliptic, second-derivative terms of (4.1)–(4.4) than by the first-derivative terms. The tables do not contain a control experiment in which the identical physical state is discretized on each of the two grids, but we can observe that Bi-CGSTAB is on average equal to or slightly better than GMRES in terms of CPU time on the fine grid, whereas GMRES is slightly better than Bi-CGSTAB on the coarse grid.

Comparing the iteration counts of Bi-CGSTAB and GMRES, we see that the former is 62.5% of the latter, averaged over a complete computation, ranging from about 55% to about 66% for different phases of the computation. This translates into the Bi-CGSTAB solver making 25% more matrix-vector product calls than GMRES. On the other hand, over a complete calculation, GMRES makes just under three times as many calls to the inner product

routine as Bi-CGSTAB. Given the readily apparent differences between the distribution of work of the two linear solvers, it is interesting that their CPU times match very closely within each phase.

6. A multidomain preconditioner. In this section, we reexamine the block-line GS preconditioner with a view toward parallelism. Computational combustion is driven in this direction by both memory requirements and execution time requirements, even for two-dimensional problems. As is widely appreciated, Newton–Krylov methods for finite-difference discretizations are easily and efficiently parallelized by domain decomposition, except, possibly, for two tasks: inner products and preconditioning steps with internal sequentiality or extensive nonlocal data dependencies. Apart from these tasks, Newton–Krylov solvers consist mainly of residual evaluations, Jacobian evaluations, and sparse matrix-vector products, in which the worst data dependencies can be confined to nearest-neighbor type. Inner products of distributed vectors involve global synchronized communication. Possibilities for recasting Krylov algorithms to form several inner products simultaneously within a vector global reduction operation have been explored by many authors with views towards stability and computational complexity, and are beyond the scope of this article. However, the sequentiality of the block-line GS preconditioner has been addressed through the following simple modification.

The preconditioned product $y = B_{GS}^{-1}Ax$ is still formed in two steps as in §2.1.1, but for multidomain GS we replace L with an $\hat{L}$ in the factor to be inverted: solve $(\hat{L} + D)z = Ux + (L - \hat{L})x$; $y \leftarrow z + x$. The operator $\hat{L}$ is just L with the off-diagonal row blocks that contain the coupling of each subdomain to the one upstream set to zero. In our application, a subdomain consists of a collection of contiguous rows of gridpoints oriented normal to the predominant flow direction. For the case of two subdomains on the coarse grid, for instance, the downstream subdomain spanning rows $j = 22$ through $j = 41$ takes its upstream data at iteration k from the data along row $j = 21$ after iteration $k - 1$.

We have previously experimented with a similar "poor man's" type of domain decomposition preconditioner in the context of block-ILU subdomain preconditioners in [23]. Though its effectiveness does not scale to the fine parallel granularity limit in elliptic systems, it is a simple and practical device at coarse granularity, motivated by the following elementary observation. Let A be a banded matrix, written in block 2×2 form

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{6.1}$$

The A_{11} block corresponds to subdomain 1 and A_{22} to subdomain 2, with coupling blocks A_{12} nonzero only in its lower left corner and A_{21} only in its upper right. Let B be a conformally blocked diagonal preconditioner

$$B = \begin{bmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{bmatrix}. \tag{6.2}$$

The preconditioned operator is

$$B^{-1}A = \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ A_{22}^{-1}A_{21} & I \end{bmatrix} \equiv I + R. \tag{6.3}$$

In the context of a naturally ordered two-dimensional PDE discretization with $\mathcal{O}(n)$ mesh intervals on a side, the dimension of $B^{-1}A$ is $\mathcal{O}(n^2)$ and the rank of R is $\mathcal{O}(n)$. Hence CG on a symmetrizable $B^{-1}A$ will converge in at most $\mathcal{O}(n)$ iterations [24], which is acceptable relative to other comparably parallelizable methods as n becomes large.

Tables 4 and 5 display the results of rerunning the Bi-CGSTAB and GMRES(20) tests with the modified preconditioner. The first columns show the same timing data as in Table 3. The next three columns display the results of complete computations carried out on the same serial machine as the single domain computation.

TABLE 4
Performance of the multidomain preconditioner with Bi-CGSTAB *(timings in minutes).*

Problem	One subdomain	Two subdomains	Four subdomains	Eight subdomains	8:1 ratios
Coarse grid transient	120.5	122.2	121.6	126.7	1.05
Coarse grid steady	9.4	10.1	9.7	11.0	1.17
Fine grid transient	3.5	3.5	3.5	3.5	1.00
Fine grid steady	25.0	28.0	29.2	27.5	1.10
Total	158.4	163.8	164.0	168.7	1.07

TABLE 5
Performance of the multidomain preconditioner with GMRES(20) *(timings in minutes).*

Problem	One subdomain	Two subdomains	Four subdomains	Eight subdomains	8:1 ratios
Coarse grid transient	116.7	121.1	118.2	123.0	1.05
Coarse grid steady	9.2	9.3	9.7	11.0	1.19
Fine grid transient	3.5	3.5	3.5	3.5	1.00
Fine grid steady	26.3	23.5	23.3	22.8	0.87
Total	155.7	157.4	154.7	160.3	1.03

At the maximum granularity of eight subdomains, the line GS preconditioning is commenced independently in each of eight strips that are, respectively, 5- and 10-grid cells wide in the streamwise direction on the coarse and fine grids. As can be observed from the ratios in the final column, comparing the cases of one and eight subdomains, the penalty in CPU time paid by this modification is very modest: 7% under Bi-CGSTAB and only 3% under GMRES. It has become conventional in reporting parallel efficiencies to distinguish between the "numerical efficiency," η_{num}, of a multithreaded algorithm relative to the best single-threaded algorithm and the "machine efficiency," η_{mach}, of the multiprocessor implementation of the multithreaded algorithm. In this nomenclature, the final column lists the reciprocals of the numerical efficiency of the parallelized preconditioning. We conclude that modest granularity parallel preconditioners are available for these problems at a very small price in terms of convergence degradation relative to an excellent serial preconditioner.

It is perhaps surprising that there are a few nonmonotonicities in the convergence behavior of both linear solvers as the number of severed j-lines increases. Moreover, on the fine grid during the final steady Newton iterations, the convergence of GMRES is actually monotonic in the unexpected direction for the first three mesh bisections shown. However, GMRES(20) stagnates when the preconditioner is further decoupled into 16 subdomains. We do not fully understand the circumstances under which using data that is less than the best available (i.e., moving away from GS towards Jacobi) will enhance convergence, but we can offer the following observations concerning the plausibility of encountering such behavior.

First, this problem is multicomponent. In some regions of the flow field, the strongest couplings of one degree of freedom to all others occur *not* between unknowns of the same type at neighboring points in space, but between unknowns of different type (e.g., velocity and vorticity) at the same point in space. Furthermore, in pseudotransient phases with very small timestep, the strongest coupling of each degree of freedom is to its own image at the previous timestep. Therefore, weakening spatial dependencies does not always make a leading-order difference in the update formula for a given degree of freedom.

Second, the linear operators are not symmetric positive definite; recall that SOR used as a stationary iteration actually requires underrelaxation for convergence throughout the entire solution trajectory. The modification of GS described above is a form of underrelaxation.

Third, we do not iterate to machine precision within each linear solution process; therefore, there is an interaction between the linear and nonlinear residuals. When the preconditioner changes, the shape of the incompletely converged Newton correction changes, which results in a different Jacobian matrix and right-hand side vector at the next Newton step. Our numerical experiments show that for finer granularity preconditioners more inner iterations are needed to solve the first steady Newton step; on the other hand, further *nonlinear* progress is made in the sense that the norm of the nonlinear residual is further reduced by the resulting step. The next Newton step then generally requires fewer linear iterations for its computation. The result is an oscillation in the linear iteration history for the decoupled problems around the linear iteration history for the single-domain problems. Threshhold effects of this oscillation along the course of the steady nonlinear iterations account for the variations in cumulative linear iterations as the number of subdomains is increased.

7. Conclusions. In this study, we compared a standard relaxation method and two CG-like algorithms combined with three preconditioner options in the solution of linear systems arising in axisymmetric flame sheet simulation problems. Both traditional (SOR) and recent (Bi-CGSTAB) solvers occupy practical niches. Though SOR cannot handle the Jacobians of the steady Newton steps on the coarse grid as early as the Krylov methods in the present problem, it has been successfully employed in streamfunction-vorticity [1], primitive variable [2], and vorticity-velocity [19] formulations both with and without chemical reaction by prolonging the pseudotransient stage and by under-relaxation. By handling fully implicitly the intra-point coupling, block-SOR can be regarded as the most economical splitting matrix for problems in which the dominant coupling is through the source terms or backwards to the previous iterate through the implicit time-differencing. The Bi-CGSTAB algorithm combined with a GS left-preconditioner gives the best execution time and does not break down in our experience. Bi-CGSTAB/GS permits passage to steady Newton iteration at an earlier stage than SOR. GMRES/GS yields nearly identical execution times without concern for breakdowns, but at a higher storage overhead in problems in which memory may be a premium.

With an eye towards future applications, it should be noted that either Bi-CGSTAB or GMRES can be employed as inner iterative loops of Newton's method in matrix-free variants with preconditioners that might rely on evaluation and storage of only selected portions of the Jacobian, or only infrequent evaluation of the Jacobian. It is to preserve this possibility on problems with more expensive Jacobian evaluations that we avoid linear solvers with an explicit reliance on matrix-vector products with the transpose of the Jacobian.

For coarse-granularity parallel computation in which each processor is responsible for a different subdomain, we have demonstrated that a number of the off-diagonal blocks of the GS preconditioner coupling adjacent subdomains may be discarded with minor penalty, leaving a block-Jacobi-like structure to the outermost iteration. This approach has been implemented in [3] and in [25]. In the latter full-scale implementation of a 26-species, 78-step reaction mechanism for a methane-air flame, an overall efficiency of 82% is reported on a six-processor IBM ES/3090 600J. GS still retains its usefulness as a subdomain preconditioner. For finer granularity parallelism, we would expect to combine it into a global preconditioner based on hierarchically coarsened grids. Simple, two-level hierarchies have been extensively tested under the rubric of "domain decomposition" methods. For some parallel performance results on model two-dimensional problems with convection and adaptive refinement, see [26].

Finally, on fine-grained distributed parallel architectures with differential rates for communication involving local and global data dependencies, the trade-off between the generally

greater number of matrix-vector products in Bi-CGSTAB and the generally greater number of inner products in GMRES may play a more significant role than CPU and total storage considerations in selecting among linear solvers.

REFERENCES

[1] M. D. SMOOKE, R. E. MITCHELL, AND D. E. KEYES, *Numerical solution of two-dimensional axisymmetric laminar diffusion flames*, Combust. Sci. and Tech., 67 (1989), pp. 85–122.

[2] Y. N. XU, *Numerical Calculations of an Axisymmetric Laminar Diffusion Flame with Detailed and Reduced Reaction Mechanisms*, Ph.D. thesis, Yale University, New Haven, CT, 1991.

[3] D. E. KEYES AND M. D. SMOOKE, *Flame sheet starting estimates for counterflow diffusion flame problems*, J. Comput. Phys., 72 (1987), pp. 267–288.

[4] I. GUSTAFSSON, *Modified incomplete Cholesky (MIC) methods*, in Preconditioning Methods: Analysis and Applications, D. J. Evans, ed., Gordon and Breach, New York, 1983.

[5] V. FABER AND T. A. MANTEUFFEL, *Necessary and sufficient conditions for the existence of a conjugate gradient method*, SIAM J. Numer. Anal., 21 (1984), pp. 352–362.

[6] 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.

[7] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, Univ. Minnesota Supercomputer Inst. Report 91/279, 1991.

[8] H. A. VAN DER VORST AND C. VUIK, *GMRESR: A Family of Nested GMRES Methods*, Technological University of Delft, Report 91-80, the Netherlands, 1991.

[9] P. SONNEVELD, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[10] H. A. VAN DER VORST, *Bi-CGSTAB: a more smoothly converging variant of CG-S for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[11] 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.

[12] R. W. FREUND, *A transpose-free quasi-minimum residual method for non-Hermitian linear systems*, RIACS Tech. Report 91.18, NASA-Ames Research Center, Moffett Field, CA, 1991.

[13] 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.

[14] E. S. ORAN AND J. P. BORIS, *Numerical Simulation of Reactive Flow*, Elsevier, New York, 1987.

[15] G. BADER AND E. GEHRKE, *Solution of Flame Sheet Problems on Transputer Networks*, in Flow Simulation on Supercomputers, Notes on Numerical Fluid Mechanics, Vol. 38, E. H. Hirschel, ed., Vieweg Verlag, Braunschweig, Germany, 1993.

[16] T. F. CHAN AND H. C. ELMAN, *Fourier analysis of iterative methods for elliptic boundary value problems*, SIAM Rev., 31 (1989), pp. 20–49.

[17] T. F. CHAN, C-C. J. KUO, AND C. TONG, *Parallel elliptic peconditioners: Fourier analysis and performance on the Connection Machine*, Comput. Phys. Comm., 53 (1989), pp. 237–252.

[18] T. F. CHAN, *Fourier analysis of relaxed incomplete factorization preconditioners*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 668–680.

[19] A. ERN AND M. D. SMOOKE, *Velocity-vorticity formulation for three-dimensional steady compressible flows*, J. Comput. Phys., 72 (1993), pp. 58–71.

[20] M. D. SMOOKE, J. A. MILLER, AND R. J. KEE, *Solution of premixed and counterflow diffusion flame problems by adaptive boundary value methods*, in Numerical Boundary Value ODEs, U. M. Ascher and R. D. Russell, eds., Birkhäuser-Verlag, Basel, Switzerland, 1985.

[21] M. D. SMOOKE AND R. M. M. MATTHEIJ, *On the solution of nonlinear two-point boundary value problems on successively refined grids*, Appl. Numer. Math., 1 (1985), pp. 463–487.

[22] A. ECDER, *Krylov Methods in Transport Modeling*, Ph.D. thesis, Dept. of Mechanical Engineering, Yale University, New Haven, CT, 1992.

[23] D. E. KEYES, *Domain decomposition methods for the parallel computation of reacting flows*, Comput. Phys. Commun., 53 (1989), pp. 181–200.

[24] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[25] M. D. SMOOKE AND V. GIOVANGIGLI, *Numerical modeling of axisymmetric laminar diffusion flames*, Impact Comput. Sci. Engrg., 4 (1992), pp. 46–79.

[26] W. D. GROPP AND D. E. KEYES, *Parallel performance of domain-decomposed preconditioned Krylov methods for PDEs with locally uniform refinement*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 128–145.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 704–712, May 1994

013

ITERATIVE SOLUTION OF THE EIGENVALUE PROBLEM IN HOPF BIFURCATION FOR THE BOUSSINESQ EQUATIONS*

HANS D. MITTELMANN†, K.-T. CHANG‡, D. F. JANKOWSKI‡, AND G. P. NEITZEL§

Abstract. Some of the most challenging eigenvalue problems arise in the stability analysis of solutions to parameter-dependent nonlinear partial differential equations. Linearized stability analysis requires the computation of a certain purely imaginary eigenvalue pair of a very large, sparse complex matrix pencil. A computational strategy, the core of which is a method of inverse iteration type with preconditioned conjugate gradients, is used to solve this problem for the stability of thermocapillary convection. This convection arises in the float-zone model of crystal growth governed by the Boussinesq equations. The results obtained complete the stability picture augmenting the energy stability results [Mittelmann, et al., *SIAM J. Sci. Statist. Comput.*, 13 (1992), pp. 411–424] and recent experimental results. Here a real eigenvalue of a Hermitian eigenvalue problem had to be determined.

Key words. eigenvalue problem, thermocapillary convection, Hopf bifurcation, linear stability, inverse iteration

AMS subject classifications. 76E15, 76D99, 65F15

1. Introduction.

The float-zone crystal-growth process is a containerless method for producing high-quality electronic material in which a rod of the material to be refined is passed through a type of heater, producing a zone of molten material that is held in place by surface-tension forces (see Fig. 1). The requirement that surface-tension forces alone support the weight of the molten zone makes the process unsuited for use with certain materials (notably gallium arsenide) in terrestrial environments. Consequently, there has been a great deal of interest in exploring the microgravity environment of space to grow larger crystals of electronic material using the float-zone method [4], [14], [19].

Along with the reduction in weight provided by a microgravity environment comes a reduction in any convection in the melt induced by buoyancy. At one time, it was believed that buoyancy-induced convection was responsible for the appearance of *striations* observed in float-zone-grown material. If this were the case, then one might also expect to produce *better* material in a microgravity environment. Associated with the float-zone process, however, is another type of convection that will not vanish in space, namely, *thermocapillary convection*, driven by temperature-induced surface-tension gradients along the free surface of the melt. In fact, it has been widely speculated that the *instability* of this convective mode is responsible for the appearance of the observed *striations*. The desire to utilize the unique environment of space coupled with these observations has led to a significant body of research associated with the stability of thermocapillary convection in models of the float-zone process.

Hydrodynamic stability theory is concerned with determining the conditions under which a certain flow, called the *basic state*, will remain stable or become unstable due to the inevitable presence of unknown perturbations. In general, these perturbations are governed by nonlinear partial differential equations. For a general reference on stability theory we refer to [6]. Linear-stability theory assumes the perturbations to be infinitesimally small and neglects the nonlinear terms in comparison with their linear counterparts. This theory is local in nature and results in a criterion that guarantees growth of these small disturbances. Typically, an

*Received by the editors May 18, 1992; accepted for publication (in revised form) January 4, 1993. This work was supported by the Air Force Office of Scientific Research grant AFOSR-90-0080 (HDM) and by Microgravity Science and Applications Division of National Aeronautics and Space Administration grant NAG-3-1221.

†Department of Mathematics, Arizona State University, Tempe, Arizona 85287-1804 (mittelmann@math.la.asu.edu).

‡Department of Mechanical and Aerospace Engineering, Arizona State University, Tempe, Arizona 85287-6106.

§The George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332-0405.

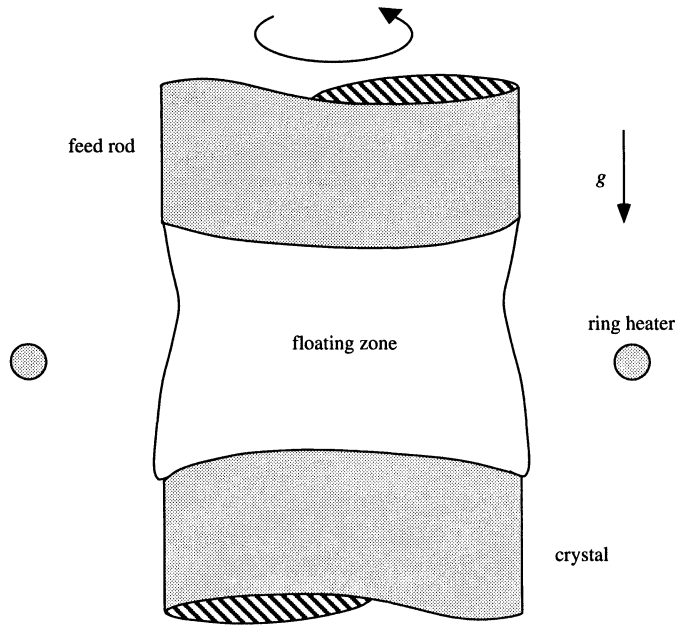


FIG. 1. *Schematic of the float-zone crystal-growth process.*

externally controllable dimensionless parameter, say R, is selected and linear theory yields a value R_L such that $R > R_L$ is a sufficient condition for instability.

Energy stability theory, on the other hand, adopts a global approach by examining the behavior of a generalized integral disturbance energy. Unlike linear stability theory, energy-stability theory provides a value R_E such that $R < R_E$ is a sufficient condition for stability of a given basic state to disturbances of *arbitrary amplitude*. This technique is equivalent to a stability analysis utilizing a Lyapunov function. The application of either theory gives rise, in general, to an eigenvalue problem.

If R_E and R_L should coincide, a rigorous stability bound is obtained. However, this is usually not the case and the proximity of R_E to R_L is a function of the physical mechanism that gives rise to the instability. Two such mechanisms for which R_E and R_L may be expected to be relatively close to each other are buoyancy and thermocapillarity.

Experimentally, it has been shown by Preisser, Schwabe, and Scharmann [13], among others, that thermocapillary convection in a model of the float-zone process undergoes a transition from steady to oscillatory convection when a dimensionless parameter known as the *Marangoni number* exceeds a particular value, the other parameters held fixed. The geometry employed by Preisser et al is termed a *half-zone* because it is meant to simulate the lower half of an actual float-zone. It consists of a pair of coaxial, solid cylindrical rods, oriented vertically, with a bridge of liquid material suspended between them. The rods are heated differentially, with the upper rod being at a higher temperature than the lower one. Buoyancy, therefore, plays a stabilizing role in the experiment since the liquid is stably stratified due to temperature in the axial direction. The basic state of thermocapillary convection that results consists of a single toroidal eddy with motion on the free surface in the direction from the hot cylinder toward the cold one.

Motivated by the above work, Shen et al. [16] undertook a stability analysis of a half-zone of $O(1)$ aspect ratio, employing energy-stability theory rather than linear theory. Their results,

computed primarily for Prandtl number Pr = 1, compared favorably with the experimental results of Preisser et al. However, their analysis had made the simplifying assumption of permitting only axisymmetric disturbances, while the oscillations observed by Preisser et alwere clearly *nonaxisymmetric* and for a material with significantly larger Prandtl number.

Computations for general disturbances were undertaken by Neitzel et al. [11]. Fortunately, experiments performed by Velten, Schwabe, and Scharmann [18] for KCl provided results for unit Prandtl numbers that eliminated the need to attempt the more difficult calculations for larger Prandtl numbers. These newer energy-theory results are in excellent agreement with the experimentally determined onset Marangoni numbers in order of magnitude, although the azimuthal structure emerging from the energy theory does not (and should not necessarily) agree with that observed experimentally.

This work seeks to complete the stability picture for the finite half-zone with a nondeformable free surface by calculating linear-stability limits for this basic state. The degree of closeness of the linear- and energy-theory results provides a bound for the region of parameter space *possibly* subject to nonlinear instability. The results show that the energy bound is less than the linear bound, but both are of the same order. The result would be consistent with subcritical Hopf bifurcation. To determine if, in fact, this is the case, additional computations would have to be done (cf. [6]). This may be done in future work. Since, furthermore, both bounds do not correspond to axisymmetric modes, *symmetry* of the basic state appears to be *broken.*

It is important to note that a difficult application could be analyzed with respect to its linearized stability without computing all or even a large number of eigenvalues of the linearization. Ultimately, in the neighborhood of the stability bound and for the finest discretization used, a method will be employed that yields just the eigenvalues of interest. Still, the compute-intensive part of this method is the solution of a linear system with the linearization as a complex system matrix. Through the use of a preconditioned conjugate gradient-type method for a related real but large system, this could be accomplished.

Since this work is as much a contribution to solving the specific class of application problems as to the numerical treatment of the algebraic subproblem, §2 contains an outline of the stability analysis as applied to the thermocapillary convection problem, while in §3 the numerical problem and the procedure used for its solution are described. Finally, some numerical results are presented in §4.

2. Linear-stability analysis. The basic state of thermocapillary convection in a half-zone with nondeformable free surface is identical to that employed and described in [16] and [11]. The axisymmetric Boussinesq equations are discretized in stream-function/vorticity form using finite differences on an equally spaced grid in the r–z plane. The computed basic state consists of a single toroidal thermocapillary eddy with flow on the free surface directed toward the bottom, cold cylinder. Isotherms become increasingly deformed from their conductive profiles as the Prandtl number of the melt is increased. Details of the numerical procedure and a discussion of the results obtained may be found in [16] and [11].

The stability analysis of the basic-state velocity, $U(x)$, temperature, $T(x)$, and pressure, $P(x)$, fields begins in the usual fashion by assuming there exists a solution to the Boussinesq equations of the form

$$q(x,t) = Q(x) + q'(x,t),$$

where q refers to any flow quantity (i.e., velocity, temperature, or pressure), a capital letter denotes the basic state, and a prime is used to denote a disturbance. Substitution of the solution into the Boussinesq equations and linearization in disturbance quantities leads to the *linearized disturbance equations* (dropping primes):

$$\text{(2.1)}\quad \mathrm{Re}[u_t + Uu_r + uU_r + Wu_z + wU_z] = -p_r + \left[\frac{1}{r}(ru)_r\right]_r + \frac{1}{r^2}u_{\varphi\varphi} + u_{zz} - \frac{2}{r^2}v_\varphi,$$

$$\text{(2.2)}\quad \mathrm{Re}\left[v_t + Uv_r + Wv_z + \frac{Uv}{r}\right] = -\frac{1}{r}p_\varphi + \left[\frac{1}{r}(rv)_r\right]_r + \frac{1}{r^2}v_{\varphi\varphi} + v_{zz} + \frac{2}{r^2}u_\varphi,$$

$$\text{(2.3)}\quad \mathrm{Re}[w_t + Uw_r + uW_r + Ww_z + wW_z] = -p_z + \frac{\mathrm{Gr}}{\mathrm{Re}}\theta + \left[\frac{1}{r}(rw)_r\right]_r + \frac{1}{r^2}w_{\varphi\varphi} + w_{zz},$$

$$\text{(2.4)}\quad \mathrm{Ma}[\theta_t + U\theta_r + uT_r + W\theta_z + wT_z] = \frac{1}{r}[r\theta_r]_r + \frac{1}{r^2}\theta_{\varphi\varphi} + \theta_{zz},$$

$$\text{(2.5)}\quad (ru)_r + v_\varphi + (rw)_z = 0.$$

In (2.1)–(2.5) we have scaled velocities by $\gamma\Delta T/\mu$, pressure by $\gamma\Delta T/R$, temperature by ΔT, and time by $R\mu/(\gamma\Delta T)$, where $\Delta T = T_H - T_C$, γ is the (positive) rate of decrease of surface tension with respect to temperature, and μ is the coefficient of dynamic viscosity of the liquid in the zone. The disturbance temperature is denoted by θ.

The dimensionless parameters appearing in (2.1)–(2.5) are

$$\begin{aligned}
&\text{Reynolds number} && \mathrm{Re} = \tfrac{\gamma R\Delta T}{\mu\nu},\\
&\text{Grashof number} && \mathrm{Gr} = \tfrac{g\alpha R^3\Delta T}{\nu^2},\\
&\text{Marangoni number} && \mathrm{Ma} = \tfrac{\gamma R\Delta T}{\mu\kappa},
\end{aligned}$$

where $\nu = \mu/\rho$ is the kinematic viscosity, ρ is the mean density, α is the coefficient of volumetric expansion, and g is the gravitational acceleration. The Prandtl number, Pr, is given by $\mathrm{Pr} = \mathrm{Ma}/\mathrm{Re}$.

The boundary conditions which complete the specification of the problem are that

$$\text{(2.6a–d)}\quad u = v = w = \theta = 0, \qquad z = 0, \Gamma,$$

$$\text{(2.7a–d)}\quad u = w_r + \theta_z = v_r - v/r + \theta_\varphi = \theta_r + \mathrm{Bi}\theta = 0, \qquad r = 1,$$

in addition to the requirement that all flow quantities remain bounded at $r = 0$. The quantity $\Gamma = H/R$ is the dimensionless *aspect ratio* of the zone. The additional parameter appearing in the free-surface heat-transfer condition (2.7d) is the Biot number, defined as

$$\mathrm{Bi} = hR/k,$$

where h is a heat-transfer coefficient and k is the thermal conductivity of the liquid.

In view of the linearity of (2.1)–(2.5) and the fact that the coefficients depend only on r and z, we make use of the Floquet theory and normal modes to write all flow quantities as

$$\text{(2.8)}\quad q(r, \varphi, z, t) = q^*(r, z)\exp(\sigma t + \mathrm{im}\varphi),$$

where $\sigma = \sigma_R + i\sigma_I$ is the complex growth rate and m is restricted to be an integer. Again, we refer to [6] for more details. Marginal stability corresponds to the condition $\sigma_R = 0$. The

form (2.8) is now substituted into (2.1)–(2.5) and the boundary conditions. A discrete version of the resulting problem is a complex, generalized eigenvalue problem of the form

$$Ax = \sigma Bx, \tag{2.9}$$

where x is the vector of unknown velocity, temperature, and pressure values at the nodes of the appropriate grid.

The corresponding eigenvalue problem from the energy-theory analysis of this basic state was, at worst, complex-Hermitian (in addition to being indefinite and sparse), whereas (2.9) has no such symmetry. An additional complication, which also existed as part of the energy-stability analysis is the fact that the basic-state velocity and temperature fields depend upon the stability parameter, Ma (equivalently, Re). For the energy-theory calculations, this required an additional level of iteration to obtain the energy limit, Ma_E. Since we formulate the problem with σ as the eigenvalue, our procedure is to fix the Prandtl, Grashof, and Biot numbers as well as the azimuthal wavenumber m and calculate the eigenvalue of system (2.9) with largest *real* part, call it σ^*, for various values of the Marangoni number. The Marangoni number Ma* corresponding to $\sigma_R^* = 0$ is the value above which infinitesimal disturbances of azimuthal wavenumber m will grow. The linear-stability limit, $\mathrm{Ma}_L(\mathrm{Pr}, \mathrm{Gr}, \mathrm{Bi})$, is therefore given by

$$\mathrm{Ma}_L(\mathrm{Pr}, \mathrm{Gr}, \mathrm{Bi}) = \min_m \mathrm{Ma}^*(\mathrm{Pr}, \mathrm{Gr}, \mathrm{Bi}; m). \tag{2.10}$$

3. The numerical method. As shown above, the determination of the linear stability bounds Ma_L requires the solution of the following eigenvalue problem

$$Ax = \sigma Bx, \tag{3.1a}$$

where A, B are $N \times N$ matrices, $N = k + l$, which are partitioned as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \qquad B = \begin{pmatrix} B_{11} & 0 \\ 0 & 0 \end{pmatrix} \tag{3.1b}$$

$A_{11}, B_{11} \in C^{k,k}$, $A_{22} \in C^{l,l}$, where k is the number of velocity/temperature unknowns and l the number of pressure unknowns. The matrices are derived from linearizing the boundary value problem for the Boussinesq equations at the solution (basic state) for specific values of the parameters (Pr, Gr, Bi). A is complex and non-Hermitian while B_{11} is taken as a multiple of the identity matrix. If the real parts of all eigenvalues are negative, the basic state is stable, and for increasing Ma that value Ma* must be found where for the first time an eigenvalue, to be called the critical eigenvalue in the following, crosses the imaginary axis. Here, it is expected that this corresponds to a simple Hopf bifurcation point [5]; that is, there will be exactly one complex-conjugate eigenvalue pair with nonzero imaginary part that crosses the imaginary axis with nonzero speed.

There is a sizable literature on the numerical computation of Hopf bifurcation points; instead of attempting to give a necessarily rather incomplete listing of works, we refer to a recent collection of articles on bifurcation problems [9] in which several papers address this issue.

Generalized eigenvalue problems of the form (3.1), on the other hand, particularly if they arise from applications as is the case here, have been studied thoroughly, and many contributions have been made to their numerical solution. Again, we confine references to just one recent survey work [7], which also includes an extensive bibliography.

Frequently, Hopf bifurcation points are detected during a continuation process. For this, in general, all eigenvalues of (3.1) are computed, a procedure that is prohibitively expensive

for large problems ($N > 10^4$). After the detection of a Hopf point, its precise location may be determined through characterizing extended systems; see [17], [3], and the references therein. This technique has proven to be useful for problems of moderate size, while it is of limited use for very large problems.

From the literature dealing with the computation of Hopf points in flow problems, [2] shall be quoted exemplarily. In this work, a relatively sophisticated combination of numerical techniques was applied to a nontrivial hydrodynamic stability problem. One feature common to the numerical approaches in [2] and many other works is the direct solution of the occurring linear systems of equations. This feature still limits the size of the problems. The dimensions of the eigenvalue problems solved are a few thousand. On the other hand, in [16] the iterative solution of these problems permitted (in-core) solution of problems of dimension $10^4 - 10^5$. From the experience with the related but different computations of the energy stability bounds in [16], there was reason to hope that, also for the determination of the linearized stability limits, a method of inverse iteration type would yield the desired results.

If it was known in advance with which imaginary part β the critical eigenvalue crosses the imaginary axis, then inverse iteration with shift $i\beta$ would allow detection of this Hopf point at least when the computation is started in the stable range, not too far from the critical parameter value and continuation in this parameter is employed. It is clear that to safeguard the obtained results, computations must be done with different values of β. Alternatively, generalizations of the numerical algorithm, i.e., the Arnoldi method, must be used to compute more than one eigenvalue at a time as was done for medium-fine discretizations. On coarse meshes all eigenvalues could be computed using the QZ method. Then, Arnoldi [15] was used to compute 10–15 eigenvalues near that of largest real part. Roughly, the computational work was greater than that for the inverse iteration used subsequently by a factor equal to the number of eigenvalues computed.

Let s denote the shift that is usually anticipated to be purely imaginary with, say, positive imaginary part β. The following form of inverse iteration was successfully applied to the present problem.

$$(A - sB)\delta x_k = (\sigma_k B - A)x_k, \tag{3.2a}$$

$$\sigma_{k+1} = \frac{(x_k + \delta x_k)_i}{\sigma_k [x_k]_i}, \tag{3.2b}$$

$$x_{k+1} = \frac{x_k + \delta x_k}{[x_k + \delta x_k]_i}, \tag{3.2c}$$

where $[x]_i$ denotes the component of the largest modulus of a vector $x \in C^n$. Here, x_0 was initially, i.e., for the first Marangoni number used, chosen as a random vector and σ_0 was chosen as 1. For a fixed shift, convergence of both the eigenvalue and eigenvector approximations will generally be linear with a factor

$$\left| \frac{s - \sigma^*}{s - \sigma_n} \right| < 1, \tag{3.3}$$

where σ^*, σ_n are the nearest and the next nearest eigenvalue to s. Also, faster convergence for the σ_k could be obtained by iterating with approximate left eigenvectors and using the generalized Rayleigh quotient, cf; e.g., [7].

The essential computational work involved in the proposed method is the solution of the linear system in (3.2a). The matrix $C = A - sB$ is complex and non-Hermitian. First, an equivalent real system of order $2N$ was formed for the matrix

$$\overline{C} = \begin{pmatrix} \mathrm{Re}(C) & -\mathrm{Im}(C) \\ \mathrm{Im}(C) & \mathrm{Re}(C) \end{pmatrix}.$$

This system was then solved using the conjugate gradient method for the normal equations [12]. For this, the matrix $\overline{C}$ was first scaled by multiplying the matrices A_{ij} in (3.1b) with appropriate powers of the discretization parameters to balance $\overline{C}$. Then, additionally, a diagonal scaling was used such that the columns of $\overline{C}$ had unit norm. Different preconditionings may well lead to a more efficient solution method; several, however, were tried without yielding a substantial reduction in work. These included outer-inner iterations utilizing the partitioning in (3.1b) as, for example, in [1] for the Stokes problem. Here, however, $A_{11} - sB_{11}$ is neither Hermitian nor definite. An incomplete LU decomposition was used to precondition the inner iterations. Nevertheless, a higher efficiency is to be expected from an adaptive multilevel approach such as hierarchical bases [1]. Instead, the finite difference discretization given above was used on a fixed grid so that the results could be compared to the energy stability results of [11] where the analogous discretization was used. Studies with various grid sizes suggest that the error of the numerical results is well below 1%.

Instead of the normal equations, the system with $\overline{C}$ could have been solved directly by suitable generalizations of conjugate gradients such as biconjugate gradients, biconjugate gradients squared, generalized minimal residuals, etc. No preconditioners could easily be found that made these methods sufficiently efficient for the cases to be solved. While $\overline{C}$ will not generally be exactly singular if $i\beta \neq \mathrm{Im}(\sigma^*)$, it will be nearly singular. It must also be noted that the values of $\mathrm{Im}(\sigma^*)$ were in the range $10^{-2} - 10^{-1}$.

4. Numerical results. Algorithm (3.2) was applied to the eigenvalue problem (3.1) for aspect ratio $\Gamma = 1$ and the parameter values $\mathrm{Pr} = 1$, $\mathrm{Gr} = 0$, and $\mathrm{Bi} = 0$ for which experimental results are given in [18]. For coarse discretizations on square grids, up until about $\delta r = \delta z = \frac{1}{20}$, the results were checked with a complex QZ routine. Then, the approximate values of Ma^*, σ^*, and the trend for decreasing grid size were known. It was not difficult to find suitable guesses for the shift s and a Marangoni number somewhat smaller than the expected Ma^*. Continuation in Ma with a bisection or secant method on $\mathrm{Re}(\sigma^*)$ was used to determine Ma^*. To check the validity of the stability bound obtained on the finest mesh, the computations were repeated for various shifts.

The critical eigenvalues computed were in the range .02i–.03i. The complex linear systems (3.2a) had total dimension N where $l = (1 + \frac{1}{\delta r})(1 + \frac{1}{\delta z})$, $k = \frac{4}{\delta r \delta z} + \frac{fac}{\delta z}$ in (3.1b) and $3 \leq fac \leq 5$ depending on m. The number of conjugate gradient iterations was less than N but $O(N)$ indicating the need for a better preconditioner. While a more complete discussion of results is given in [10], it should be noted that the least-stable mode from energy theory, here $m = 1$, is not the same as the most unstable mode from the linear theory. In Fig. 2 the present linear stability bounds and our earlier energy bounds are graphed. In fact, the linear bound is obtained for $m = 2$

$$\mathrm{Ma}_E \approx 1685, \qquad \mathrm{Ma}_L \approx 2484.$$

For this bound $\delta r = \delta z = \frac{1}{59}$ resulted in an order $N = 17,709$ for the complex eigenvalue problem (3.1).

As in the case of exact comparability of experimental and computational models, the experimentally determined stability limits that should be between the energy and linear stability limits, are, in fact, close to the energy limits. While no definite statement can be made, these results would be consistent with subcritical Hopf bifurcation with *breaking* of the *axisymmetry* of the basic state. Clearly, there is additional nontrivial numerical computation needed to complete the physical understanding of this problem.

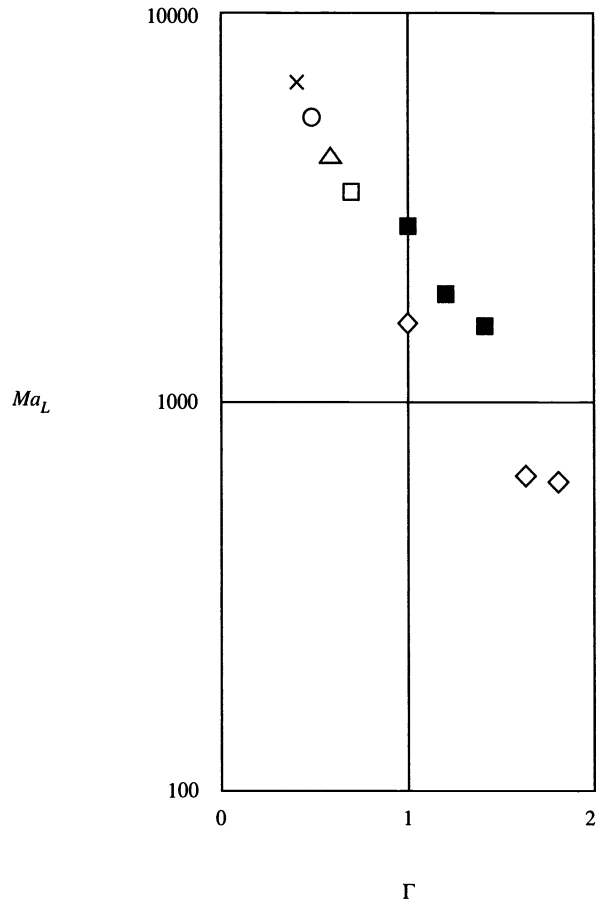


FIG. 2. *Comparison of linear-stability results (solid symbols) for* Gr = 0, Bi = 0 *with those of energy-stability theory. Azimuthal wavenumber is indicated by symbol type: cross,* $m = 5$; *circle,* $m = 4$; *triangle,* $m = 3$; *square,* $m = 2$; *diamond,* $m = 1$.

The computations were performed on a 4-processor Stardent 3000 in the Advanced Research Computing Facility of the Department of Mathematics at Arizona State University; the QZ computations were done on a CRAY2 at the NASA Ames Research Center.

REFERENCES

[1] R. E. BANK, B. D. WELFERT, AND H. YSERENTANT, *A class of iterative methods for solving saddle point problems*, Numer. Math., 56 (1990), pp. 645–666.

[2] K. N. CHRISTODOULOU AND L. E. SCRIVEN, *Finding leading modes of a viscous free surface flow: An asymmetric generalized eigenproblem*, J. Sci. Comput., 3 (1988) pp. 355–405.

[3] B. DEDIER, D. ROOSE, AND P. VAN ROMPAY, *Interaction between fold and Hopf curves leads to new bifurcation phenomena*, in Continuation Techniques and Bifurcation Problems, H. D. Mittelmann and D. Roose, eds., 1990, pp. 171–186.

[4] A. EYER, H. LEISTE, AND R. NITSCHE, *Floating zone growth of silicon under microgravity in a sounding rocket*, J. Crystal Growth, 71 (1985), pp. 173–182.

[5] E. HOPF, *Abzweigung einer periodischen Lösung von einer stationären Lösung eines Differentialsystems*, Ber. Verh. Sachs. Akad. Wiss. Leipzig. Math. -Nat. Kl.95, 1 (1943), pp. 3–22.

[6] G. IOSS AND D. D. JOSEPH, *Elementary Stability and Bifurcation Theory*, Springer-Verlag, New York, Berlin, Heidelberg, 1990.

[7] W. KERNER, *Large-scale complex eigenvalue problems*, J. Comput. Phys., 85 (1989), pp. 1–85.

[8] H. D. MITTELMANN, C. LAW, D. F. JANKOWSKI, AND G. P. NEITZEL, *A large sparse and indefinite generalized eigenvalue problem from fluid mechanics*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 411–424.

[9] H. D. MITTELMANN AND D. ROOSE, EDS., *Continuation Techniques and Bifurcation Problems*, ISNM 92, Birkhäuser-Verlag, Basel, 1990.

[10] G. P. NEITZEL, K.-T. CHANG, D. F. JANKOWSKI, AND H. D. MITTELMANN, *Linear-stability theory of thermocapillary convection in a model of the float-zone crystal growth process*, Phys. Fluids A, 1992. Submitted.

[11] G. P. NEITZEL, C. C. LAW, D. F. JANKOWSKI, AND H. D. MITTELMANN, *Energy stability of thermocapillary convection in a model of the float-zone crystal-growth process. Part 2, Non-axisymmetric disturbances*, Phys. Fluids A, 3 (1991), pp. 2841–2846.

[12] C. C. PAIGE AND M. A. SAUNDERS, LSQR: *An algorithm for sparse linear equations and sparse least squares* ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[13] F. PREISSER, P. SCHWABE, AND A. SCHARMANN, *Steady and oscillatory thermocapillary convection in liquid columns with free cylindrical surface*, J. Fluid Mech., 126 (1983), pp. 545–567.

[14] R. RUPP, G. MÜLLER, AND G. NEUMANN, *Three-dimensional time dependent modelling of the Marangoni convection in zone melting configurations for GaAs*, J. Crystal Growth, 97 (1989), pp. 34–41.

[15] Y. SAAD, *Variations of Arnoldi's method for computing equivalents of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.

[16] Y. SHEN, G. P. NEITZEL, D. F. JANKOWSKI, AND H. D. MITTELMANN, *Energy stability of thermocapillary convection in a model of the float-zone crystal-growth process*, J. Fluid Mech., 217 (1990), pp. 639–660.

[17] A. SPENCE, K. A. CLIFFE, AND A. D. JEPSON, *A note on the calculation of paths of Hopf bifurcations*, in Continuation Techniques and Bifurcation Problems, H.D. Mittelmann and D. Roose, eds., 1990, pp. 125–131.

[18] R. VELTEN, D. SCHWABE, AND A. SCHARMANN, *The periodic instability of thermocapillary convection in cylindrical liquid bridges*, Phys. Fluids A, 3 (1991), pp. 267–279.

[19] J.-J. XU AND S. H. DAVIS, *Convective thermocapillary instabilities in liquid bridges*, Phys. Fluids, 27 (1984), pp. 1102–1107.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 713–735, May 1994

014

PRECONDITIONED, ADAPTIVE, MULTIPOLE-ACCELERATED ITERATIVE METHODS FOR THREE-DIMENSIONAL FIRST-KIND INTEGRAL EQUATIONS OF POTENTIAL THEORY*

K. NABORS†, F. T. KORSMEYER‡, F. T. LEIGHTON§, AND J. WHITE†

Abstract. This paper presents a preconditioned, Krylov-subspace iterative algorithm, where a modified multipole algorithm with a novel adaptation scheme is used to compute the iterates for solving dense matrix problems generated by Galerkin or collocation schemes applied to three-dimensional, first-kind, integral equations that arise in potential theory. A proof is given that this adaptive algorithm reduces both matrix-vector product computation time and storage to order N, and experimental evidence is given to demonstrate that the combined *preconditioned, adaptive, multipole-accelerated* (PAMA) method is nearly order N in practice. Examples from engineering applications are given to demonstrate that the accelerated method is substantially faster than standard algorithms on practical problems.

Key words. boundary-element methods, Laplace's equation, potential theory

AMS subject classifications. 65N38, 45L10

1. Introduction. Mixed first- and second-kind surface integral equations with $\frac{1}{r}$ and $\frac{\partial}{\partial n}\frac{1}{r}$ kernels are generated by a variety of three-dimensional engineering problems. For such problems, Nyström-type algorithms cannot be used directly, but an expansion for the unknown, rather than for the entire integrand, can be assumed and the product of the singular kernel and the unknown integrated analytically. Combining such an approach with a Galerkin or collocation scheme for computing the expansion coefficients is a general approach, but leads to dense matrix problems. In this paper, we focus on accelerating such techniques for purely first-kind integral equations of potential theory, and present an overlapping-block preconditioned Krylov-subspace iterative algorithm for solving the associated dense matrix problem, where a modified multipole algorithm with a novel adaptation scheme is used to compute the iterates. This approach follows along lines originally suggested in [1].

In the section that follows, we review Galerkin and collocation algorithms applied to the first-kind integral equation, as well as Krylov-subspace iterative algorithms for solving the generated dense matrix problem. Our approaches to applying a fast multipole algorithm to this problem are described in §3, and a simplified complexity analysis for our adaptive scheme is given. In §4, we present more general proofs of the adaptive multipole algorithm linear computational growth. The preconditioning strategy for accelerating the Krylov-subspace method convergence is described in §5, and experimental results gained from using the method to analyze a variety of structures derived from engineering problems are presented in §6. Finally, conclusions are given in §7.

2. Formulation. Consider the first-kind integral equation for a single-layer surface density, hereafter referred to as a charge density, generated by solution of the exterior Dirichlet problem in a multiply-connected domain. (For a second-kind formulation of this problem see

*Received by the editors June 22, 1992; accepted for publication (in revised form) February 12, 1993.This work was supported by the Defense Advanced Research Projects Agency contract N00014-91-J-1698, Office of Naval Research contract N00014-90-J-1085, the National Science Foundation contract MIP-8858764 A02, Federal Bureau of Investigation contract J-FBI-88-067, and grants from Digital Equipment Corporation and I.B.M.

†Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (ksn@rle-vlsi.mit.edu); (white@rle-vlsi.mit.edu).

‡Department of Ocean Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (xmeyer@chf.mit.edu).

§Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139 (ftl@math.mit.edu).

[2].) The charge density σ satisfies the integral equation

$$\psi(x) = \int_S \sigma(x') \frac{1}{\|x - x'\|} da', \qquad x \in S, \tag{1}$$

where S is a two-dimensional surface in $\mathbf{R}^3$, $\psi(x)$ is a given surface potential, da' is the incremental surface area, x, $x' \in \mathbf{R}^3$, and $\|x\|$ is the usual Euclidean length of x given by $\sqrt{x_1^2 + x_2^2 + x_3^2}$. More compactly, we denote (1) by

$$\psi = \mathcal{L}\sigma. \tag{2}$$

The charge density σ can be related to electrostatic capacitances and forces or fluid velocities for the case of potential flow. Electrostatic capacitances are useful figures of merit for designers of electronic packaging [3]; microelectromechanical system designers are interested in electrostatic forces [4]; and ocean vehicle designers are interested in potential flow [5]. For these engineering problems, 0.1%–1% accuracy is typically sufficient, and therefore low-order schemes are in common use.

To compute an approximation to σ, one can generally consider an expansion of the form

$$\sigma(x) \approx \sum_{i=1}^{N} q_i \theta_i(x), \tag{3}$$

where $\theta_1(x), \ldots, \theta_N(x) : \mathbf{R}^3 \to \mathbf{R}$ are a set of not necessarily orthogonal expansion functions, and $q_1, \ldots, q_N$ are the unknown expansion coefficients. Typically, $\theta_1(x), \ldots, \theta_N(x)$ represent an approximate discretization of the surface S, where each θ_i is nonnegative, of compact support, and satisfies a normalization condition

$$\int_S \theta_i(x') da' = 1. \tag{4}$$

The expansion coefficients are then determined by requiring that they satisfy a Galerkin or collocation condition of the form

$$Pq = \overline{p}, \tag{5}$$

where $P \in \mathbf{R}^{N \times N}$ and $\overline{p}, q \in \mathbf{R}^N$. In the case of a Galerkin condition,

$$P_{ij} = \langle \theta_j, \mathcal{L}\theta_i \rangle \tag{6}$$

and

$$\overline{p}_i = \langle \theta_i, \psi \rangle, \tag{7}$$

where $\langle f, g \rangle \equiv \int_S f(x')g(x')da'$. For the collocation condition,

$$P_{ij} = \langle \delta(x_j), \mathcal{L}\theta_i \rangle \tag{8}$$

and

$$\overline{p}_i = \langle \delta(x_i), \psi \rangle, \tag{9}$$

where $\langle \delta(x), u \rangle \equiv u(x)$, and $x_1, \ldots, x_N$ are the collocation points [6].

Remark. If the θ_i's are nonnegative, then for both the Galerkin and collocation methods, P is a positive matrix, and for the Galerkin method, P is symmetric and positive definite. The normalization condition implies that for pairs of θ_i's whose support is widely separated, the associated off-diagonals of P approach $\frac{1}{r}$, where r is the separation distance.

Example. The approach used in many engineering applications is to approximate the surface S with N planar quadrilateral and/or triangular panels over which σ is assumed to be uniform. The expansion functions are then

$$\theta_i(x) = \begin{cases} \frac{1}{a_i} & \text{if } x \text{ is on panel } i, \\ 0 & \text{otherwise,} \end{cases} \tag{10}$$

where a_i is the area of panel i. The Galerkin scheme for determining the expansion coefficients yields

$$P_{ij} = \frac{1}{a_i\, a_j} \int_{\text{panel}_i} \int_{\text{panel}_j} \frac{1}{\|x - x'\|}\, dada', \tag{11}$$

and for the collocation method

$$P_{ij} = \frac{1}{a_i} \int_{\text{panel}_j} \frac{1}{\|x_i - x'\|}\, da'. \tag{12}$$

There are closed-form expressions for the integral in (12) [5], but closed-form expressions for the integral in (11) are known only in special cases [3].

The dense linear system of (5) can be solved to compute expansion coefficients from a right-hand side derived from (7) or (9). If Gaussian elimination is used to solve (5), the number of operations is order N^3. Clearly, this approach becomes computationally intractable if the number of expansion functions exceeds several hundred. Instead, consider solving the linear system (5) using a Krylov-subspace method such as generalized minimal residual (GMRES)[7]. Such methods have the general form of Algorithm 2.1.

ALGORITHM 2.1. General Krylov-subspace algorithm for solving (5).
Set q^0 to some initial guess at the solution.
Compute the initial residual and Krylov vector, $r^0 = p^0 = \overline{p} - Pq^0$.
Determine the value of a cost function (e.g., $\|r^0\|$.)
Set $k = 0$.
repeat {
  if (cost < *tolerance*), return q^k as the solution.
  $p^{k+1} = Pp^k$.
  Choose α's and β in
    $q^{k+1} = \sum_{j=0}^{k} \alpha_j p^j + \beta p^{k+1}$
  to minimize the cost function.
  Set $k = k + 1$.
}

The dominant costs of Algorithm 2.1 are in calculating the N^2 entries of P using (6) or (8) before the iterations begin and in performing N^2 operations to compute Pp^k on each iteration. Described below are modified and adaptive multipole algorithms that avoid forming most of P and reduce the cost of forming Pp^k to order N operations. This does not necessarily imply that each iteration of a GMRES-style algorithm can be computed with order N operations. If the number of GMRES iterations required to achieve convergence approaches N, then to perform the minimization in each GMRES iteration will require order N^2 operations. This

problem is avoided through the use of a preconditioner, also described below, that in practice reduces the number of GMRES iterations required to achieve convergence to well below N for large problems.

Remark. As the Galerkin matrix P is symmetric and positive definite, it is tempting to replace a Krylov-subspace method suitable for nonsymmetric problems with the conjugate gradient algorithm. Unfortunately, this is not recommended because if a multipole algorithm is used to approximately compute Pp^k, this is equivalent to using an iterative method to solve $\tilde{P}q = \overline{p}$, where $\tilde{P}$ is the multipole algorithm's *not necessarily symmetric*, but sparse, approximation to P.

3. Multipole algorithms. In the case of collocation, computing the dense matrix-vector product Pq is equivalent to evaluating the potential at N collocation points, $\{x_1, \ldots, x_N\}$, due to a charge density described by $\sum_{i=1}^{N} q_i \theta_i(x)$. It is possible to avoid forming P, and to substantially reduce the cost of computing Pq, using the fast multipole algorithm [8], [9]. The fast multipole algorithm uses a hierarchical partitioning of the problem domain and careful application of multipole and local expansions to accurately compute potentials at N points due to N charged particles in order N operations.

In particular, in the fast multipole algorithm, potentials due to clusters of charges are represented by truncated multipole expansions. These expansions have the general form

$$\psi(r, \theta, \phi) \approx \sum_{n=0}^{l} \sum_{m=-n}^{n} \frac{M_n^m}{r^{n+1}} Y_n^m(\theta, \phi), \tag{13}$$

where l is the expansion order, r, θ, and ϕ are the spherical coordinates with respect to the multipole expansion's origin (usually the center of the charge cluster), $Y_n^m(\theta, \phi)$'s are the surface spherical harmonics, and the M_n^m's are the multipole coefficients [10].

Multipole expansions can be used to efficiently evaluate the potential due to a cluster of charges at any point where the distance between the evaluation point and the cluster center is significantly larger than the radius of the cluster. A dual optimization is possible using local expansions. That is, for a cluster of evaluation points, the potential due to charges whose distances from the cluster center are significantly larger than the radius of the cluster can be combined into a local expansion at the cluster center. Then, this local expansion can be used to efficiently compute potentials at evaluation points in the cluster. A local expansion has the form

$$\psi(r, \theta, \phi) \approx \sum_{n=0}^{l} \sum_{m=-n}^{n} L_n^m Y_n^m(\theta, \phi) r^n, \tag{14}$$

where l is the order of the expansion, r, θ, and ϕ are the spherical coordinates of the evaluation location with respect to the expansion center, and the L_n^m's are the local expansion coefficients.

For a more complete introduction to the three-dimensional fast multipole algorithm, refer to the references. In the remainder of this section, we focus instead on the several aspects of the fast multipole algorithm that must be modified to create an efficient algorithm for computing the matrix-vector products associated with using an iterative scheme to solve the discretized integral equations of potential theory. For such problems, the charge is a surface density given by $\sum_{i=1}^{N} q_i \theta_i(x)$, rather than a set of point charges, and this mildly complicates the procedure for computing multipole expansion coefficients. In addition, using an iterative algorithm to solve the discretized integral equation implies that the multipole algorithm is used many times to compute potentials, but in iteration computation, only the coefficients of the charge density expansion functions change. This implies that efficiency can be improved if quantities with only geometric dependencies are computed once and stored. Finally, the spatial nonuniformity

associated with surface discretizations can be efficiently handled using an adaptive algorithm different from that given in [11].

3.1. Computing multipole coefficients. In general, the coefficients of a multipole expansion for a charge density σ in a volume V are given by

$$M_n^m = \int_V \sigma(\rho, \alpha, \beta)\rho^n Y_n^{-m}(\alpha, \beta)dV, \tag{15}$$

where ρ, α, and β represent position in spherical coordinates. In the case where the charge density is given by $\sum_{i=1}^{N} q_i\theta_i(x)$, substitution in (15) leads to

$$M_n^m = \sum_{i=1}^{N} q_i \int_V \theta_i(\rho, \alpha, \beta)\rho^n Y_n^{-m}(\alpha, \beta)dV. \tag{16}$$

Note that the relationship between the multipole coefficients and the density expansion coefficients is linear, assuming that the geometric quantities are given.

In general, the multipole coefficients can be computed easily using quadrature formulas; the integrand in (16) contains no singularities. It is also possible to derive a closed form expression for the integral in (16) for the case of the piecewise-constant expansion function in (10).

THEOREM 3.1. *Let Q be any triangular or quadrilateral region of a plane in $\mathbf{R}^3$. There exists a closed form expression for*

$$\int_Q \rho^n Y_n^{-m}(\alpha, \beta)da, \tag{17}$$

where ρ, α, β are the spherical coordinates of points on the panel surface.

Proof. Formulas for shifting and rotating spherical harmonics are already well known, having been derived for problems in quantum angular momentum [12]. It is therefore sufficient to demonstrate a closed form expression for (17) in the case where the coordinate system origin is coincident with the panel centroid, and the coordinate system x_1-x_2 plane is coplanar with the panel.

To begin, note that by definition,

$$Y_n^m(\phi, \theta) = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}(\cos\theta)e^{im\phi}. \tag{18}$$

Substituting into (17) and noting that the coordinate system assumption implies that $\cos\beta = 0$ for any point on the panel surface,

$$\int_Q \rho^n Y_n^{-m}(\alpha, \beta)da = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}(0) \int_Q \rho^n e^{im\phi}da. \tag{19}$$

For $k = n - |m|$ odd, $P_n^{|m|}(0) = 0$ and therefore (19) evaluates to zero. For $k = n - |m|$ even, $\rho^n e^{im\phi}$ can be expanded in terms of x_1 and x_2 as

$$\rho^n e^{im\phi} = \sum_{j=0}^{\frac{n-|m|}{2}} \binom{\frac{n-|m|}{2}}{\frac{n-|m|}{2} - j} \sum_{k=0}^{|m|} (-\mathrm{sgn}(m)i)^{|m|-k} \binom{|m|}{|m|-k} x_1^{2j-k} x_2^{n-(2j+k)}. \tag{20}$$

The terms

$$\mathcal{I}_{j,k} \equiv \int_Q x_1^j x_2^k da \tag{21}$$

are the moments of the panel for which analytic formulas can be found in [5]. Using the moments and combining (20) and (19) leads to

$$M_n^m = K_n^m \sum_{j=0}^{\frac{n-|m|}{2}} \binom{\frac{n-|m|}{2}}{\frac{n-|m|}{2}-j} \sum_{k=0}^{|m|} (-\mathrm{sgn}(m)i)^{|m|-k} \binom{|m|}{|m|-k} \mathcal{I}_{2j-k,n-(2j+k)}, \tag{22}$$

where

$$K_n^m = \sqrt{\frac{(n-|m|)!}{(n+|m|)!}} P_n^{|m|}(0). \qquad \square \tag{23}$$

3.2. Matrix representation. As mentioned above, the iterative algorithm described here uses the fast multipole algorithm to compute matrix-vector products. This implies that the algorithm is used many times without a change in geometry, and this fact can be exploited to improve algorithm efficiency. As is made clearer below, the fast multipole algorithm involves constructing multipole expansions from charge density expansions, shifting and combining multipole expansions, converting multipole expansions and charge density expansions into local expansions, shifting local expansions, and evaluating the potential due to charge density expansions, multipole expansions, or local expansions. However, each of the construction, translation, and evaluation operations is a linear function of the expansion coefficients, and therefore can be represented as a matrix whose elements are functions of geometry alone [13].

From the above observation we have the following theorem.

THEOREM 3.2. *The fast multipole algorithm computes an approximation to the potential due to a charge density that is a linear function of the density expansion coefficients. In addition, it is possible to represent this linear transformation directly in terms of translation matrices whose entries are functions only of geometry.*

The fast multipole algorithm produces an approximation to Pq, denoted $\tilde{P}q$, which is a linear function of q. That is, the fast multipole algorithm is directly analogous to a sparse representation of the dense matrix P. This has several important consequences. When the multipole algorithm is used to compute matrix-vector products in an iterative method, the iterative method's convergence is controlled by the properties of $\tilde{P}$, not by how well $\tilde{P}$ approximates P. Also, the translation matrices need be computed only once, then used repeatedly until the iteration converges.

3.3. The modified multipole algorithm. Mostly to establish notation, we give a modified multipole algorithm that closely follows the development in [8]. The algorithm allows for the charge density to be described in terms of expansion functions, and also exploits Theorem 3.2 by exclusively using translation matrices.

To begin, we consider the hierarchical domain partitioning. Let the root cube be the smallest cube containing the problem domain. More precisely, the root cube is the smallest cube that contains all the collocation points and for which x outside the cube implies $\theta_i(x) = 0$ for all i. The hierarchy is then just a recursive eight-way sectioning of this root cube.

DEFINITION 3.1. *Cube hierarchy. The cube containing the problem domain is referred to as the level* 0, *or root, cube. Then, the volume of the cube is subdivided into eight equally sized child cubes, referred to as level-*1 *cubes, and each has the level-*0 *cube as its parent. The collocation points are distributed among the child cubes by associating a collocation point*

*with a cube if the point is contained in the cube. Each of the level-*1 *cubes is then subdivided into eight level-*2 *child cubes and the collocation points are again distributed. The result is a collection of* 64 *level-*2 *cubes and a* 64*-way partition of the collocation points. This process is repeated to produce D levels of cubes and D distributions of collocation points starting with an* 8*-way partition and ending with an* 8^D*-way partition. The depth D is chosen so that the maximum number of* θ_i*'s whose support intersects any finest level cube is less than a selected constant (see Definition* 3.11).

The terms below are used to concisely describe the modified multipole algorithm.

DEFINITION 3.2. *Evaluation points of a cube. The collocation points within the cube.*

DEFINITION 3.3. *Nearest neighbors of a cube. Those cubes that have a corner in common with the given cube.*

DEFINITION 3.4. *Second-nearest neighbors of a cube. Those cubes that are not nearest neighbors but that have a corner in common with a nearest neighbor of the given cube.*

Note that there are at most 124 nearest and second-nearest neighbors of a cube, excluding the cube itself.

DEFINITION 3.5. *Interaction set of a cube. The set of cubes that are either the second nearest neighbors of the given cube's parent, or are children of the given cube's parent's nearest neighbors, excluding nearest or second-nearest neighbors of the given cube.*

There is a maximum of 189 cubes in an interaction set. Roughly half of the cubes are from a level one coarser than the level of the given cube; the rest are on the same level.

For the jth cube on level d: its parent's index on level $d-1$ is denoted $F(d, j)$; its set of $d+1$ level children is denoted $C(d, j)$; its set of of interaction cubes is denoted $I(d, j)$; the set of cube j and cube j's nearest and second-nearest neighbors is denoted $N(d, j)$; the vector of multipole expansion coefficients representing the charge density in the cube is denoted $M_{d,j}$; the vector of local expansion coefficients for the cube is denoted $L_{d,j}$; the vectors of the cube's charge density expansion coefficients and collocation point potentials are denoted $q_{d,j}$ and $p_{d,j}$, respectively. The matrix that represents the conversion of a multipole expansion from the level $\tilde{d}$ cube $\tilde{j}$ center to a local expansion at the level d cube j center is denoted $M2L(d, j, \tilde{d}, \tilde{j})$. The $L2L$, $M2M$, and $Q2P$ translation matrices are similarly specified. Finally, the matrix that maps $q_{\tilde{d},\tilde{j}}$ to $M_{d,j}$ is denoted $Q2M(d, j, \tilde{d}, \tilde{j})$, and the $L2P$ matrix is similarly specified.

ALGORITHM 3.1. The Modified Multipole Algorithm.

/* THE CONSTRUCTION PHASE: Computes multipole expansions at the finest level. */

For each level D cube $j = 1$ to 8^D

$M_{D,j} = Q2M(D, j, D, j)q_{D,j}$

/* THE UPWARD PASS: Computes multipole expansions. */

For each level $d = D-1$ to2

For each level d cube $j = 1$ to 8^d

$M_{d,j} = \sum_{\tilde{j}\in C(d,j)} M2M(d, j, d+1, \tilde{j})M_{d+1,\tilde{j}}$

/* THE INTERACTION PHASE: Converts multipole expansions to local expansions. */

For each level $d = 2$ to D

For each level d cube $j = 1$ to 8^d

$L_{d,j} = \sum_{\tilde{d},\tilde{j}\in I(d,j)} M2L(d, j, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$

/* THE DOWNWARD PASS: Transfers and accumulates local expansions. */

For each level $d = 3$ to D

For each level d cube $j = 1$ to 8^d

$L_{d,j} = L_{d,j} + L2L(d, j, d-1, F(d, j))L_{d-1,F(d,j)}$

/* THE EVALUATION PHASE: Evaluates the potential. */
For each level D cube $j = 1$ to 8^D
$p_{D,j} = L2P(D, j)L_{D,j} + \sum_{\tilde{j} \in N(D,j)} Q2P(D, j, D, \tilde{j})q_{D,j}$

3.4. The adaptive multipole algorithm. Almost certainly, the surfaces of a given three-dimensional problem will not fill the volume of the root cube, and this will necessarily lead to a nonuniform distribution of the support of the θ_i's. It is possible to derive an adaptive multipole algorithm from Algorithm 3.1 by breaking up the problem domain nonuniformly. In this case, when the cube hierarchy is created, cubes are not subdivided if they intersect the support of fewer than some limiting number of θ_i's [11]. However, for the kinds of charge density expansion functions in common use, such an approach can sometimes require more computation than a nonadaptive algorithm [14]. A more effective approach in this setting, one which is guaranteed to use fewer operations than a nonadaptive algorithm, is to avoid translation from charge density expansion coefficients to multipole expansions, and to avoid forming local expansions, whenever such representations are inefficient. For example, consider computing $M_{D,j}$ from $Q2M(D, j, D, j)q_{D,j}$. If the number of entries in vector $q_{D,j}$ is smaller than the number of multipole coefficients in $M_{D,j}$, which for an lth-order multipole expansion is $(l+1)^2$, then $q_{D,j}$ is a more efficient representation of the charge distribution, and that representation can be propagated through the algorithm instead of $M_{D,j}$. Note that using such an approach requires some rather straight-forward bookkeeping and a few easily derived translation operators [13]. In particular, multipole coefficient to evaluation point and charge to local expansion coefficient translation operators, denoted $M2P$ and $Q2L$, respectively, are required.

A second optimization for the nonuniform case, one that is guaranteed to reduce the operation count during the upward pass and also improve accuracy slightly, is to exploit the fact that there is no need to construct a multipole expansion for a cube with only one nonempty child. This is made clear in the following remark.

Remark. Suppose the charge density in a given cube is entirely contained in one of the cube's descendants. Then the potential due to charge in the given cube is at least as accurately represented using the descendant's multipole expansion about the descendant's center, as by a shifted version of the descendant's multipole expansion about the given cube's center.

A similar optimization can be used to improve the efficiency of the downward pass. That is, there is no advantage to creating a local expansion for a cube with only one nonempty child. Instead, the multipole expansions associated with the members of a cube's interaction set can be translated directly to the cube's only nonempty child.

To describe an adaptive algorithm that exploits the above optimizations, the following additional definitions are used.

DEFINITION 3.6. *Adaptive cube. Any nonempty finest level cube or a coarser level cube that has more than one nonempty child.*

DEFINITION 3.7. *Adaptive child. An adaptive child of a given cube is any adaptive cube descendent which has no ancestors that are adaptive cube descendents of the given cube. The set of adaptive children of level d cube with index j is denoted* $C^A(d, j)$.

Note that a cube need *not* be adaptive to have adaptive children. However, if a nonempty cube is not an adaptive cube, $|C^A(d, j)|$, which denotes the number of elements in $C^A(d, j)$, is precisely one.

DEFINITION 3.8. *Adaptive parent. The adaptive parent of a given adaptive cube is the unique adaptive cube for which the given adaptive cube is an adaptive child. The adaptive parent of a level d adaptive cube j is denoted by* $F^A(d, j)$.

Note that an adaptive parent can be separated from an adaptive child by an arbitrary number of levels.

DEFINITION 3.9. *Adaptive interaction set. The adaptive interaction set of a cube is the set of all members of the cube interaction set, except any nonadaptive cube member is replaced with its adaptive child. The adaptive interaction set is denoted* $I^A(d, j)$.

ALGORITHM 3.2. Adaptive Multipole Algorithm.

/* THE CONSTRUCTION PHASE: Computes multipole expansions at the finest level. */

For each level D nonempty cube j

if $(size(q_{D,j}) > (l+1)^2)$ $M_{D,j} = Q2M(D, j, D, j)q_{D,j}$.

/* THE UPWARD PASS: Computes multipole expansions. */

For each level $d = D-1$ to 2

For each level d *adaptive* cube j

if $(size(q_{d,j}) > (l+1)^2)$ {

For each level $\tilde{d}$ cube $\tilde{j} \in C^A(d, j)$

if $(size(q_{\tilde{d},\tilde{j}}) > (l+1)^2)$ $M_{d,j} = M_{d,j} + M2M(d, j, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$

else $M_{d,j} = M_{d,j} + Q2M(d, j, \tilde{d}, \tilde{j})q_{\tilde{d},\tilde{j}}$

}

/* THE INTERACTION PHASE: Converts multipole expansions to local expansions. */

For each level $d = 2$ to D

For each level d cube j for which $|I^A(d, j)| > 0$

if $(|C^A(d, j)| > 1)$ $\hat{d} = d$ and $\hat{j} = j$.

else $\hat{d}, \hat{j}$ is the only member of $C^A(d, j)$.

if $(size(p_{d,j}) > (l+1)^2)$ {

For each $\tilde{d}, \tilde{j} \in I^A(d, j)$

if $(size(q_{\tilde{d},\tilde{j}}) > (l+1)^2)$ $L_{\hat{d},\hat{j}} = L_{\hat{d},\hat{j}} + M2L(\hat{d}, \hat{j}, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$.

else $L_{\hat{d},\hat{j}} = L_{\hat{d},\hat{j}} + Q2L(\hat{d}, \hat{j}, \tilde{d}, \tilde{j})q_{\tilde{d},\tilde{j}}$.

}

else {

For each $\tilde{d}, \tilde{j} \in I^A(d, j)$

if $(size(q_{\tilde{d},\tilde{j}}) > (l+1)^2)$ $p_{d,j} = p_{d,j} + M2P(d, j, \tilde{d}, \tilde{j})M_{\tilde{d},\tilde{j}}$.

else $p_{d,j} = p_{d,j} + Q2P(d, j, \tilde{d}, \tilde{j})q_{\tilde{d},\tilde{j}}$.

}

/* THE DOWNWARD PASS: Transfers and accumulates local expansions. */

For each level $d = 3$ to D

For each level d *adaptive* cube j

$\tilde{d}, \tilde{j} = F^A(d, j)$

if $(size(p_{d,j}) > (l+1)^2)$

if $(\tilde{d} > 1)$ $L_{d,j} = L2L(d, j, \tilde{d}, \tilde{j})L_{\tilde{d},\tilde{j}}$

else if $(size(p_{\tilde{d},\tilde{j}}) > (l+1)^2)$

if $(\tilde{d} > 1)$ $p_{d,j} = L2P(d, j, \tilde{d}, \tilde{j})L_{\tilde{d},\tilde{j}}$

/* THE EVALUATION PHASE: Evaluates the potential. */

For each level D nonempty cube j

if $(size(p_{D,j}) > (l+1)^2)$ $p_{D,j} = L2P(D, j)L_{D,j}$

$p_{D,j} = p_{D,j} + \sum_{\tilde{j} \in N(D,j)} Q2P(D, j, D, \tilde{j})q_{D,\tilde{j}}$

Since the adaptive algorithm is derived by reducing or avoiding operations in a nonadaptive algorithm, the theorem below follows directly.

THEOREM 3.3. *Algorithm* 3.2 *always uses fewer operations than Algorithm* 3.1.

3.5. Complexity analysis. It is clear from the description of both the nonadaptive and adaptive multipole algorithms that they can require more than order N operations if expansion functions with arbitrary support are used to represent the charge density. Although there are techniques that use sets of expansion functions whose support is the entire surface, we will insist on the following two conditions.

DEFINITION 3.10. *Compactness condition. The problem domain and $\theta_1, \ldots, \theta_N$ are such that there exists a cube hierarchy of depth D for which the number of cubes intersecting the support of each θ_i is bounded by K_{ct}, independent of N.*

DEFINITION 3.11. *Bounded overlap condition. The problem domain and $\theta_1, \ldots, \theta_N$ are such that there exists a cube hierarchy of depth D for which the number of θ_i's whose support intersects each finest-level cube i is bounded by K_{tc}, independent of N.*

Previous approaches to bounding the computation of adaptive multipole algorithms assumed an a priori bound on on the maximum number of levels in a cube hierarchy, denoted $\tilde{D}$ [11]. Such an assumption is loosely justified by the notion that given a machine precision, there is a smallest representable finest-level cube. The precision assumption has the suspicious consequence that in any cube hierarchy, each cube has a bounded number, in fact $\tilde{D}$, ancestors. Given this, simple examples can lead to seeming contradictions. Consider the surface S to be a flat square plate of unit area, and suppose the surface charge density on the plate is represented with expansion functions corresponding to small square panels over which the charge density is assumed constant (see (10)). If a $\sqrt{N} \times \sqrt{N}$ array of such square panels is used to discretize the plate, then the finest-level cube size required to satisfy the compactness and bounded overlap conditions must be order $\sqrt{N}$ in diameter. But this requires a partitioning depth of order $\log N$, violating the a priori bound on the depth implied by the precision assumption.

Nevertheless, the multipole algorithm completes in order N operations for the square plate example. For that example, and for typical problems associated with the discretization of two-dimensional surfaces in three-dimensional domains, a perhaps more relevant assumption is the hierarchical contraction condition given below.

DEFINITION 3.12. *Hierarchical contraction condition. The problem domain and $\theta_1, \ldots, \theta_N$ are such that the number of nonempty cubes at level $d-1$ is less than γ times the number of nonempty cubes at level d, where γ is strictly less than one and independent of N.*

The importance of the hierarchical contraction condition is made clear in the following lemma.

LEMMA 3.4. *If the hierarchical contraction condition is satisfied, then the number of nonempty cubes, summed over all the levels, is bounded by*

$$N * K_{ct} * \beta, \tag{24}$$

where $\beta \equiv \frac{1}{1-\gamma}$ is independent of N.

Proof. If the hierarchical contraction condition is satisfied, then, by definition, the total number of nonempty cubes summed over all the levels is

$$\sum_{i=0}^{D} N * K_{ct} * \gamma^i < N * K_{ct} * \sum_{i=0}^{\infty} \gamma^i = N * K_{ct} * \beta. \tag{25}$$ □

Given that a problem satisfies the hierarchical contraction condition, it is easily shown that Algorithm 3.2 completes in order N operations. It is also possible to prove such a result without this assumption, and we return to the general case in §4.

THEOREM 3.5. *Given the hierarchical contraction condition, if the problem domain and $\theta_1, \ldots, \theta_N$ are such that there exists a cube hierarchy of depth D for which the compactness and bounded overlap conditions above are satisfied, then the nonadaptive modified multipole*

algorithm (if empty cubes are ignored), and the adaptive multipole algorithm, complete in order N operations.

Proof. To prove the theorem, operations are counted for each of the five steps of Algorithm 3.1, assuming that empty cubes are ignored. Then by Theorem 3.3, the result is a also a bound on the number of operations for Algorithm 3.2.

Construction phase. Each N θ_i contributes to $(l+1)^2$ terms in the multipole expansion of at most K_{ct} cubes.

$$\text{Cost} \leq N * K_{ct} * (l+1)^2.$$

The upward pass. By Lemma 3.4, the total number of nonempty cubes summed over all the levels is bounded by $N * K_{ct} * \beta$. In the upward pass, each nonempty cube at each level is associated with one $M2M$ translation to its nonempty parent, so there is a total of no more than $N * K_{ct} * \beta$ $M2M$ operations, each of which costs $(l+1)^4$ operations.

$$\text{Cost} \leq N * K_{ct} * \beta * (l+1)^4.$$

The interaction phase. Again by Lemma 3.4, the total number of nonempty cubes summed over all the levels is bounded by $N * K_{ct} * \beta$. Each nonempty member of a nonempty cube interaction set is associated with one $M2L$ translation, and there are no more than 189 nonempty members in a cube interaction set. Therefore, in the interaction phase, there is a total of no more than $N * K_{ct} * \beta * 189$ $M2L$ operations, each of which costs $(l+1)^4$ operations.

$$\text{Cost} \leq N * K_{ct} * \beta * (l+1)^4 * 189.$$

The downward pass. Again by Lemma 3.4, the total number of nonempty cubes is bounded by $N * K_{ct} * \beta$, and as each nonempty cube is associated with one $L2L$ translation from its nonempty parent, there is a total of no more than $N * K_{ct} * \beta$ $L2L$ operations, each of which costs $(l+1)^4$ operations, in the downward pass.

$$\text{Cost} \leq N * K_{ct} * \beta * (l+1)^4.$$

The evaluation phase. For each collocation point in a finest-level cube j, the contribution of the local expansion to the potential, which costs $(l+1)^2$ operations to evaluate, must be added to the potential due to the charge density associated with the fewer than $125 * K_{tc}$ θ_i's whose support intersects cube j.

$$\text{Cost} \leq N * (125 * K_{tc} + (l+1)^2).$$

Adding the costs leads to an order N bound on the total number of operations,

$$\text{Total Cost} \leq N * \left[\left((K_{ct}+1) * (l+1)^2\right) + \left(K_{ct} * \beta * 191 * (l+1)^4\right) + (125 * K_{tc})\right]. \quad \square \tag{26}$$

Note that for the flat square plate with a square panel discretization example mentioned above, $\gamma = 0.25$, and therefore $\beta = \frac{4}{3}$.

One last aspect of the multipole algorithm in this context must be considered. For efficiency, it is assumed that all the translation matrices will be computed once and stored, so they can be rapidly reapplied during an iterative matrix solution algorithm. This suggests that the question of the required memory be addressed. Since each application of the multipole algorithm uses every translation matrix element, the theorem below follows easily.

THEOREM 3.6. *If the multipole algorithm completes in order N operations, then storing the translation matrices requires order N storage.*

4. General theorem for Algorithm 3.2. In §3, it is claimed that the number of operations in Algorithm 3.2 is order N. However, to simplify the proof of the bounds on the upward pass, the downward pass and the interaction phase of Algorithm 3.2, a hierarchical contraction condition was assumed. In this section, we give more general proofs of these bounds.

To simplify notation, results in this section will be given in terms of the number of nonempty finest-level cubes, denoted M. Using the notation of §3, it is clear that $M \leq (N * K_{ct})$.

To bound the number of operations in the upward and downward passes of Algorithm 3.2, we make use of a graph that can represent either pass. The graph tree structure leads naturally to the bounds using results from elementary graph theory (see, for example, [15]).

DEFINITION 4.1. *Adaptive upward-pass graph. Let each adaptive cube in each level of the cube hierarchy correspond to a node in a graph, and insert an edge between pairs of nodes if one of the associated adaptive cubes is an adaptive child of the other. The resulting graph is the adaptive upward-pass graph.*

Since an edge connects every adaptive child cube to its parent, there is a one-to-one correspondence between the edges in the adaptive upward-pass graph and the $M2M$ or $Q2M$ translation operations in the upward pass of Algorithm 3.2. Similarly, there is also a one-to-one correspondence between the edges in the adaptive upward-pass graph and the $L2L$ or $L2P$ translation operations in the downward pass of Algorithm 3.2.

LEMMA 4.1. *The adaptive upward-pass graph is a tree with M leaves, or childless nodes, and every nonleaf node in the tree except the root has at least three edges connected to it. That is, each nonleaf node except the root has degree greater than two.*

Proof. A graph that is connected and has e edges and n nodes with $e = n - 1$ is a tree. The adaptive upward-pass graph is connected because any two nodes in the graph correspond to two cubes that are parts of at least one larger cube. A path from any particular node to another node can always be found passing through the node corresponding to the larger cube or directly between the two nodes of given cubes if one of the two given cubes is contained in the other.

Furthermore, every node represents an adaptive child cube and therefore has a single edge connecting it to the node corresponding to its adaptive parent, except the node corresponding to the level 0 cube. Since this level 0 node has no such edge, $e = n - 1$. Thus the adaptive upward-pass graph is a tree, and the level 0 node may be taken as its root.

The definition of an adaptive cube implies that each nonroot and nonfinest-level adaptive cube must have an adaptive parent and at least two adaptive children. Therefore, every associated nonroot and nonleaf node in the adaptive upward-pass graph has degree greater than two. The leaves of the upward-pass graph obviously correspond to the M nonempty finest-level cubes since these adaptive cubes have no children. □

The bounds are a direct consequence of the upward-pass graph tree structure as summarized by Lemma 4.1.

THEOREM 4.2. *The total number of $M2M$ or $Q2M$ operations in the upward pass, as well as the total number of $L2L$ or $L2P$ operations in the downward pass, is bounded by $2 * M$, where M is the number of nonempty finest-level cubes.*

Proof. Since there is a one-to-one correspondence between each edge in the adaptive upward-pass graph and either upward-pass or downward-pass translation operations, the theorem can be proved by demonstrating that the upward-pass graph has no more than $2 * M$ edges. Since the sum of the degrees of any graph's nodes is equal to twice the number of edges in the graph (each edge connects exactly two nodes), Lemma 4.1 implies

$$M + 3(n - M - 1) + 2 \leq 2e, \tag{27}$$

in the case of an upward-pass graph with n nodes and e edges. The first term is the sum of the leaf-node degrees, the second corresponds to the sum of minimum nonleaf node degrees, and the last is the minimum degree of the root. Since $e = n - 1$,

$$e \leq 2(M - 1). \qquad \square \tag{28}$$

The following theorem addresses the computational complexity of the interaction phase in Algorithm 3.2. The interaction phase uses $M2L$, $Q2L$, $M2P$, and $Q2P$ operations to translate information between cubes in each interaction set. We now prove that the total number of these interaction operations is of the order of the number of nonempty finest-level cubes. To avoid somewhat less interesting complications that arise when an interaction set contains cubes on two different levels, the theorem is proved using the definition of an interaction set given in the original description of the fast multipole algorithm [8]. To avoid confusion, we refer to this set as the regular interaction set.

DEFINITION 4.2. *Regular interaction set of a cube. Those cubes that are children of the given cube's parent's nearest and second-nearest neighbors, excluding nearest or second-nearest neighbors of the given cube.*

Unlike the interaction set given in Definition 3.5, the regular interaction set of a cube has many more members (875 rather than 189) but all the members are cubes from the same level as the given cube. Clearly, using the regular interaction set in Algorithm 3.2 only increases the required computation, so the upper bounds on computation derived below are also upper bounds on the interaction phase of Algorithm 3.2.

A second useful definition describes one approach to merging regular interaction sets of a cube and its descendents.

DEFINITION 4.3. *Generalized interaction set of a cube. Those cubes at the same level as the given cube that are either members of the given cube's regular interaction set or that contain descendents who are members of the regular interaction set of one of the given cube's descendents. For cubes with no descendents, that is, those cubes on the finest level of a hierarchy, the generalized interaction set is defined by assuming that the hierarchy extends to an additional finer level.*

The generalized interaction set of a cube can also be described as the union of the cube's regular interaction set with the cube's nearest and second-nearest neighbors. Or, equivalently, the generalized interaction set of a cube contains the children of the cube's parent, excluding the given cube, and the children of the parent's nearest and second-nearest neighbors. Therefore, the generalized interaction set for a cube contains no more than 999 members, all at the level of the given cube, and a cube is in no more than 999 generalized interaction sets (cubes near the sides or corners of the problem domain may have fewer members in their interaction sets).

THEOREM 4.3. *Regardless of the distribution or number of levels in the cube hierarchy, the number of interaction operations in the interaction phase of Algorithm* 3.2 *is bounded by*

$$1998 * M, \tag{29}$$

where M is the number of nonempty finest-level cubes.

Proof. Given a nonempty cube at the Dth, or finest, level in the cube hierarchy, there is at most one interaction operation associated with each of up to 999 members of the given cube's generalized interaction set. And as each nonempty cube is a member of at most 999 generalized interaction sets, each cube is associated with at most 999 interaction operations. This may seem to be an unnecessarily generous count, as on this finest level there are only interaction operations between a nonempty cube and the nonempty members in the cube's *regular* interaction set. However, for the purposes of establishing an inductive argument, we

consider the possibility of an interaction operation associated with any member of a cube's *generalized* interaction set.

Now consider cubes at level $D-1$. Suppose that for each of the nonempty $D-1$ level cubes, all but one of the cube's nonempty Dth level children are emptied. We denote the number of emptied cubes as p_D, where p_D is also equal to the difference between the number of nonempty cubes at level D and the number of nonempty cubes at level $D-1$. And as the p_D Dth level cubes are emptied in a way that does not completely empty any originally nonempty $D-1$ level cubes, no $D-1$ level or coarser interaction operations will be eliminated or added. In addition, each of the emptied cubes has no more than 999 Dth level cubes in its generalized interaction set, and each emptied cube is contained in at most 999 Dth level generalized interaction sets. Therefore, by emptying the p_D Dth level cubes, fewer than $1998 * p_D$ Dth level interaction translation operations will be eliminated.

Once the p_D cubes have been emptied, each nonempty $D-1$ level cube contains exactly one nonempty child. As a result, given a nonempty $D-1$ level cube, there is at most one interaction operation associated with each member of the given cube's generalized interaction set. This follows from the fact that each nonempty member of a given $D-1$ level cube's generalized interaction set *either* is in the given cube's regular interaction set, *or* contains a single nonempty child that is in the generalized interaction set of the given cube's only nonempty child, *but not both*. If the former is true, the associated interaction operation is between $D-1$ level cubes, and if the latter is true, the associated interaction operation is between Dth level cubes.

The above argument establishes that emptying p_D Dth level cubes, and eliminating the associated interaction operations, results in a set of $D-1$ level cubes which have the property that given a nonempty $D-1$ level cube, there is at most one interaction operation associated with each member of the given cube's generalized interaction set. This was the only property about the Dth level cubes used in the above argument, so the argument can be reapplied to levels $D-1$, $D-2$, through to the coarsest level. In inductively applying the above argument at level $d-1$, p_d nonempty d level cubes will be emptied, and, in doing so, no more than $1998 * p_d$ interaction operations will be eliminated. In addition, since only nonempty cubes will be emptied, each emptied cube must correspond to at least one of M nonempty finest-level cubes, and therefore $\sum_{d=1}^{D} p_d \leq M$. Finally, as there are no interaction operations on the coarsest level, all the interaction operations are eliminated by continuing the induction to the coarsest level. Therefore, the total number of interaction operations in the interaction phase of Algorithm 3.2 is bounded by

$$\sum_{d=1}^{D} 1998 * p_d \leq 1998 * M. \qquad \square \tag{30}$$

5. Preconditioning the iterative method. In general, the convergence of a Krylov-subspace method can be significantly accelerated by preconditioning if there is an easily computed approximation to the problem's inverse. We denote the approximation to P^{-1} in (5) by $\tilde{C}$, in which case preconditioning is equivalent to solving

$$P\tilde{C}x = \overline{p} \tag{31}$$

for the unknown vector x, from which the vector of expansion coefficients is determined by $q = \tilde{C}x$. Clearly, if $\tilde{C}$ is precisely P^{-1}, then (31) is trivial to solve, but $\tilde{C}$ will be very expensive to compute.

In [16] and [17], it was suggested that a good approximation to P^{-1} could be derived by solving overlapping subproblems. Such an approach fits naturally with the hierarchical

multipole algorithm because the preconditioner can be constructed and applied in a cube-by-cube fashion. First, $\tilde{C}$ is computed by inverting a sequence of reduced P matrices, one associated with each finest-level cube, as in Algorithm 5.1 below.

ALGORITHM 5.1. Forming $\tilde{C}$.
For each finest-level nonempty cube j
/* Form P^j with the Q2P matrices of cube j's nearest neighbors. */
For each $\tilde{j} \in N(D, j)$
For each $\tilde{k} \in N(D, j)$
$P^j_{\tilde{j},\tilde{k}} = Q2P(D, \tilde{j}, D, \tilde{k})$
Compute $\tilde{C}^j = (P^j)^{-1}$.
For each θ_i whose support intersects cubes in $N(D, j)$
if θ_i's support does *not* intersect cube j **delete** the associated row from $\tilde{C}^j$.

The matrix P^j is a block matrix, and each entry $P^j_{\tilde{j},\tilde{k}}$ is a matrix, not a scalar. In general, since the number of collocation points contained in the cubes in $N(D, j)$ will not necessarily be equal to the number of expansion functions whose support intersects the cubes in $N(D, j)$, P^j is not square. Therefore, forming $(P^j)^{-1}$ should be interpreted to imply a generalized inverse.

By comparing Algorithm 5.1 with Algorithm 3.2, it is clear that P^j uses only those elements of the full P matrix that are already required in Algorithm 3.2, and therefore the computational cost in computing the preconditioner is only in inverting small P^j matrices. Then computing the product $P\tilde{C}x^k$, which would be used in a Krylov-subspace method applied to solving (31), is accomplished in two steps. First, the preconditioner is applied to form $q^k = \tilde{C}x^k$ using Algorithm 5.2 below. Then, Pq^k is computed using Algorithm 3.2.

ALGORITHM 5.2. Forming $q = \tilde{C}x$.
For each finest-level nonempty cube j
For each θ_i whose support intersects cube j
For each θ_k whose support intersects cubes in $N(D, j)$
Add $\tilde{C}^j_{i,k}x_k$ to q_i.

The cost of the preconditioner is linked to the cost of the multipole algorithm, provided the collocation points are distributed among the expansion functions in a reasonable manner. The statement is made precise below.

THEOREM 5.1. *If the collocation points are distributed so that associated with each expansion function there is a unique collocation point contained in the expansion function's support, and the expansion functions satisfy the bounded overlap and compact support conditions, then computing the preconditioner requires order N operations.*

Proof. If the collocation points satisfy the distribution condition in Theorem 5.1, and the expansion functions satisfy the bounded overlap condition, then each of the dimensions of each P^j is bounded by $K_{tc} * 125$. From the compact support condition, there are at most $K_{ct} * N$ nonempty cubes. Therefore, computing the preconditioner costs $N * K_{ct} * (K_{tc} * 125)^3$ operations. □

6. Experimental results. In this section, results from computational experiments in solving (1) are presented. The experiments are conducted using FASTCAP [13], a three-dimensional electrostatic analysis program that uses an implementation of the preconditioned GMRES algorithm with adaptive multipole acceleration (PAMA). The program uses the piecewise-constant panel expansion functions given in (10) and a panel centroid collocation scheme. First, idealized examples of potential flow problems are examined to allow a

controlled investigation of aspects of the algorithm, and then results from realistic engineering applications are shown to exhibit the practicality of the method.

6.1. Spheres in potential flow. The potential due to a unit sphere in an infinite fluid translating at unit velocity along the x_3-axis is given by

$$\psi(x) = -\frac{x_3}{2\|x\|^3}. \tag{32}$$

The charge density σ satisfying (1) for the potential in (32) can be determined analytically. To derive the formula, Green's theorem is written twice, once for the known ψ on the sphere's outside, and once for a ψ' (some as yet unknown potential) on the sphere's inside. Summing the two expressions gives

(33)

$$2\pi(\psi(x) + \psi'(x)) + \iint_S [(\psi(x') - \psi'(x'))\nabla G(x, x') - G(x, x')\nabla(\psi(x') - \psi'(x'))] \cdot \hat{n} da' = 0, \quad x \in S,$$

where $\hat{n}$ is the inward directed surface normal and

$$G(x, x') = \frac{1}{\|x - x'\|}. \tag{34}$$

If we let σ satisfy

$$\sigma(x) = \frac{1}{4\pi}\nabla(\psi(x) - \psi'(x)) \cdot \hat{n}, \tag{35}$$

then (1) may be obtained from (33), if ψ' is such that $\psi - \psi'$ vanishes on S; namely,

$$\psi'(x) = -\frac{1}{2}x_3. \tag{36}$$

By substitution of (32) and (36) in (35),

$$\sigma(x) = \frac{-3}{8\pi}x_3, \qquad x \in S. \tag{37}$$

Figure 1 is a plot of a shaded sphere, where the shading corresponds to the charge density given in (37).

Having the solution for the translating sphere problem in closed form allows us to demonstrate the convergence of the complete algorithm. Figure 2 shows that the convergence rate will approach $\frac{1}{N}$ if the tolerance on the convergence of the iterative method is small enough, and if the order to which terms are retained in the spherical harmonic expansions is high enough. The definition of the integrated error is the following summation over the N panels

$$\mathcal{E} = \sum_{i=1}^{N} |q_i - a_i\sigma(x_i)|, \tag{38}$$

where a_i is the area of panel i and $|\frac{q_i}{a_i} - \sigma_i(x)|$ is the error between the computed and exact solution at panel i's collocation point. The results in the figure using lower-order expansions and a larger tolerance for convergence of the iterative method show how these parameters limit the accuracy of the computed solution given a particular spatial discretization.

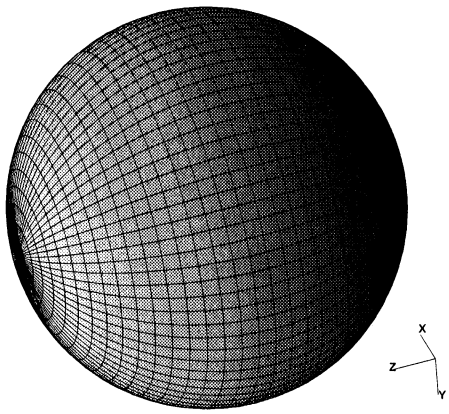


FIG. 1. *The single sphere discretized by* 2592 *panels translating in an infinite fluid. The shading corresponds to the density strength* $\sigma(x)$. *The dark shading at the pole of the sphere is a plotting artifact.*

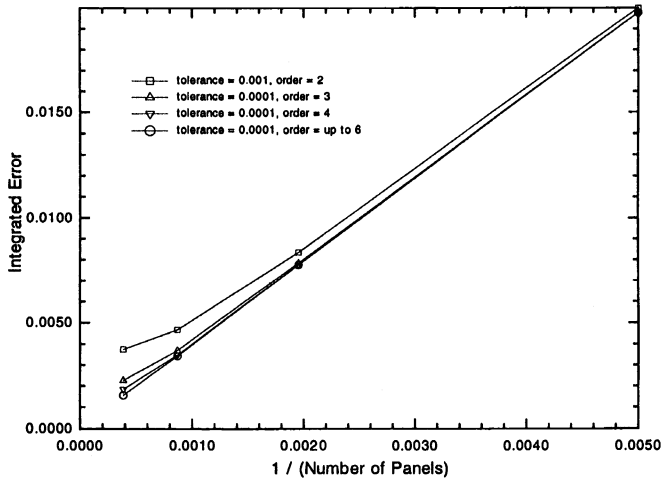


FIG. 2. *Convergence study for the single sphere translating in an infinite fluid. Tolerance refers to the convergence criterion for the iterative method, and order refers to the highest term retained in the spherical harmonic expansions.*

To investigate the advantages of using the adaptive multipole-accelerated (AMA) algorithm and the preconditioner, a more complicated case of two spheres, shown in Fig. 3, is considered. In this case, we do not have a closed form solution, so a Dirichlet problem is contrived by applying the known potential of the single sphere case to each of the two spheres. The difference in the cost of computing Pq directly and with the adaptive multipole algorithm is shown in Fig. 4. Here the operation count is the number of multiply-add operations required to compute Pq once, assuming that the entries in P and the multipole translation matrices have been precomputed. As the graph in Fig. 4 clearly demonstrates, the cost of computing Pq with the adaptive multipole algorithm increases linearly with the number of panels, and for the case where second-order expansions are used, is faster than direct computation with as few as 500 panels.

The effect of using the preconditioner to solve the two-sphere problem is shown in Fig. 5. As is clear from the figure, the number of iterations required is proportional to $\sqrt{N}$ without preconditioning, but curiously decreases slightly with N with preconditioning. Note

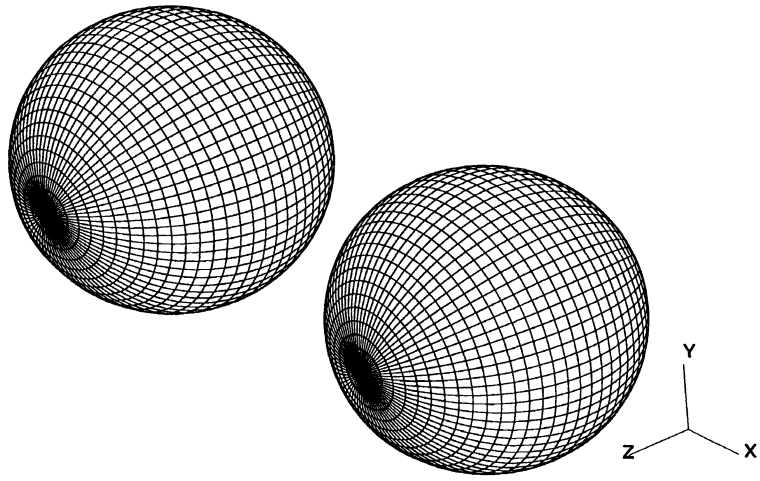


FIG. 3. *The two-sphere case, each discretized by* 2592 *panels. Here, a fictitious potential is applied. The sphere centers are three radii apart.*

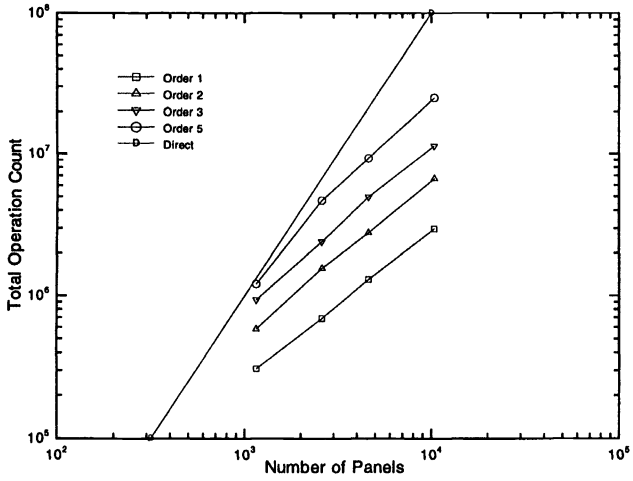


FIG. 4. *Operation counts for computing the iterates for the fictitious Dirichlet problem of two spheres, where the discretization is refined.* Direct *refers to the standard (order* (N^2)*) application of the matrix to a vector,* Order 1 *refers to the adaptive multipole algorithm with expansions to order* 1.

that the number of levels in the cube hierarchy directly affects what is used as a preconditioner. As Fig. 5 shows, the smaller the number of levels used, the larger the spatial extent of the preconditioner; the result is a reduction in the number of iterations required to achieve convergence.

6.2. Capacitance extraction. In this section we present results using the complete algorithm to solve for the capacitance matrix associated with a collection of m ideal conductors embedded in a uniform lossless dielectric medium. The capacitance matrix is defined as the $m \times m$ matrix that relates conductor potentials to the integral of conductor charge density. The jth column of the capacitance matrix can be computed by solving for the surface charge density on each of the conductors when the jth conductor is at unit potential, and all the other

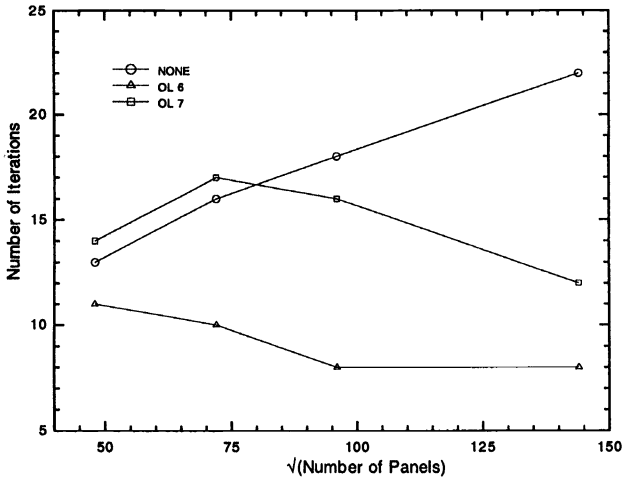


FIG. 5. *Iterations required for convergence to a tolerance of* 0.001 *in the solution of the fictitious Dirichlet problem of two spheres, as the discretization is refined.* None *refers to no preconditioning,* OL *refers to use of the overlapping-block preconditioner, and the associated number,* 6 *or* 7, *indicates the number of levels in the cube hierarchy.*

conductors are at zero potential. Note that all of the capacitance calculations are the result of using second-order multipole expansions and a GMRES relative convergence tolerance of 0.01.

The complete algorithm is nearly as accurate as the direct factorization method on complex problems, such as the 2×2 woven bus structure in Fig. 6. In Table 1, the capacitances computed using the two methods are compared using coarse, medium, and fine discretizations of the woven bus structure, also shown in Fig. 6. Note that even the coupling capacitance C_{12} between conductors one and two, which is forty times smaller than the self-capacitance C_{11}, is computed nearly as accurately with the complete algorithm as with direct factorization.

TABLE 1

Capacitance values (in pF) illustrating the accuracy of the PAMA *algorithm for the complicated geometry of Fig.* 6.

Method	Problem					
	Woven1 1584 panels		Woven2 2816 panels		Woven3 4400 panels	
	C_{11}	C_{12}	C_{11}	C_{12}	C_{11}	C_{12}
Direct	251.6	−6.353	253.2	−6.446	253.7	−6.467
PAMA	251.8	−6.246	253.3	−6.334	253.9	−6.377

On complex capacitance extraction problems, the computational cost of using the complete algorithm is roughly proportional to the product of the number of conductors, m, and the number of panels n. This is experimentally verified by computing the capacitances of the2×2 woven bus structure in Fig. 6, with progressively finer discretizations. In Fig. 7, the execution times required to compute these capacitances are plotted as a function of mn, and as the graph demonstrates, the execution time does grow nearly linearly.

To demonstrate the effectiveness of various aspects of the PAMA iterative algorithm on a range of problems, the execution times required to compute the capacitances of four different examples using four different methods are given in Table 2. The 2×2 woven bus example is described above, and the 5×5 woven bus example is the obvious extension. The via

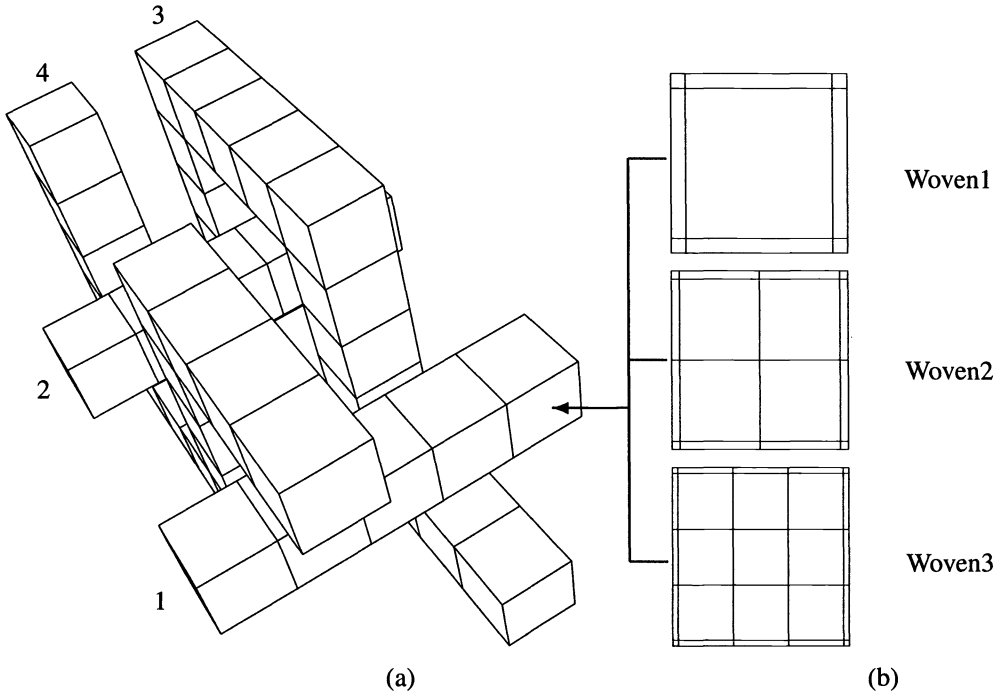


FIG. 6. *The* 2 × 2 *woven bus problem: bars have* 1*m* × 1*m cross sections. The three discretizations are obtained by replacing each square region in* (a) *with the corresponding set of panels in* (b).

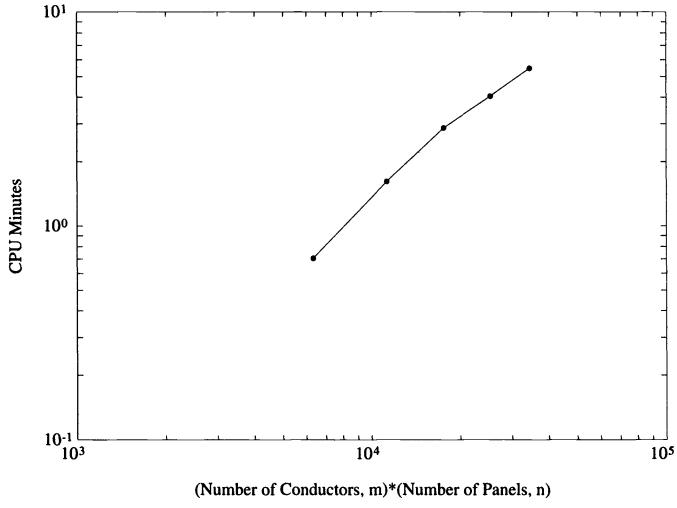


FIG. 7. *Execution time as a function of mn for the* PAMA *algorithm applied to solving progressively finer discretizations of the* 2 × 2 *woven bus problem.*

example, shown in Fig. 8, models a pair of connections between integrated circuit pins and a chip-carrier, and the diaphragm example, shown in Fig. 9, is a model for a microsensor [18].

From Table 2, it can be seen that using the AMA algorithm can improve execution time by a factor of two over using the multipole-accelerated (MA) algorithm alone, and combining the preconditioner with the AMA algorithm can reduce the execution time by as much as a factor

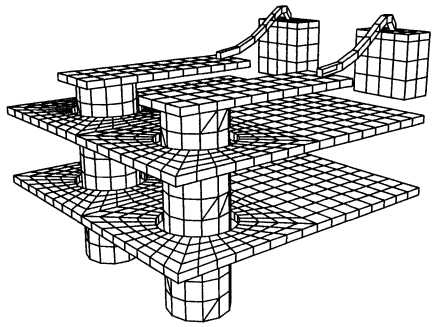


FIG. 8. *Two signal line vias passing through conducting planes.*

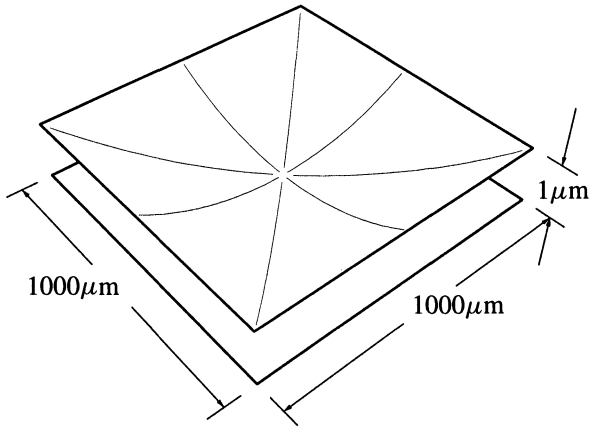


FIG. 9. *A schematic illustration of the diaphragm problem. The gap between the two plates is* 0.02 *μm at the center.*

TABLE 2

CPU times in minutes on an IBM RS6000/540 *for the nonadaptive* MA, AMA, *and* PAMA *algorithms. Times in parentheses are extrapolated.*

Method	2 × 2 woven bus 4400 panels	Via 6185 panels	Diaphragm 7488 panels	5 × 5 woven bus 9630 panels
Direct	185	(490)	(890)	(1920)
MA	6.0	11	8.7	42
AMA	3.3	4.7	5.9	23
PAMA	2.3	3.2	1.3	11

of seven. The improvement due to the adaptive algorithm is small because it is being compared to an MA algorithm that is already somewhat adaptive; empty cubes being ignored. Exploiting empty cubes is easy to implement, and makes a significant difference. For the largest problem, the 5 × 5 woven bus, more than 252,000 out of 262,000 cubes used to partition the problem domain are empty. A truly nonadaptive MA algorithm would therefore be *twenty-five* times slower than the MA algorithm used here for comparison.

The reduction in execution time afforded by the AMA algorithm over the nonadaptive MA stems from more efficiently computing the Pq product on each iteration of the GMRES algorithm, and using the preconditioner reduces execution time by reducing the number of

iterations required to achieve convergence. Because of various program overheads, comparing total execution times can hide the sometimes dramatic effect the preconditioner can have on GMRES convergence. To show the impact of the preconditioner more directly, the norm of the residual, $\|\overline{p} - Pq^k\|$, is plotted in Fig. 10 as a function of iteration for the various algorithms applied to solving the diaphragm problem. As is evident in the figure, the nonadaptive MA and AMA algorithms converge nearly identically, as expected, but the residuals computed with the PAMA algorithm decrease considerably faster. It is this rapid convergence that easily offsets the disadvantage that preconditioned iterates are slightly more expensive to compute than unpreconditioned iterates.

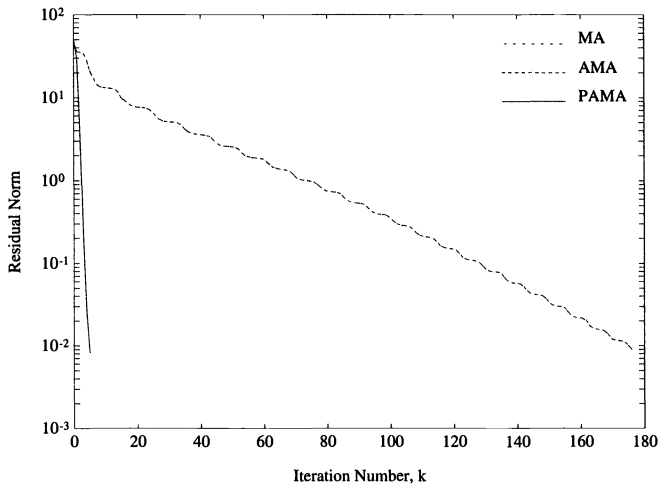


FIG. 10. *The* GMRES *residual norms for the linear system solution corresponding to the diaphragm problem (Fig.* 9*) with the top conductor at unit potential. As is evident here, the* PAMA *algorithm converges significantly more rapidly than the unpreconditioned* AMA *or* MA *algorithms.*

7. Conclusions. In this paper, a PAMA approach to solving first-kind surface integral equations with a $\frac{1}{r}$ kernel is described, and the method is shown to be effective for several engineering problems. A novel adaptive fast multipole algorithm is given and is proved to require order N computation and order N storage. Also, experimental evidence is given to demonstrate that in practice the combined algorithm is nearly order N. Note that the derivation and results are for a collocation scheme, the extension to a Galerkin scheme is straightforward though somewhat more cumbersome to implement.

Acknowlegments. The authors would like to thank Stephen Vavasis for his suggestions about approaches to preconditioning, Don Baltus for his suggestions about using graphs to represent the multipole algorithm, and Songmin Kim for his help in linking the program with the solid modeler PATRAN. The authors would also like to thank David Ling and Albert Ruehli of the I.B.M. T. J. Watson Research Center for their helpful suggestions about capacitance calculations and Dick Yue for his help in understanding potential flow problems.

REFERENCES

[1] V. ROHKLIN, *Rapid solution of integral equation of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[2] A. GREENBAUM, L. GREENGARD, AND G. B. MCFADDEN, *Laplace's Equation and the Dirichlet–Neumann Map in Multiply Connected Domains*, Tech. rep., Courant Institute, New York University, NY, March 1991.

[3] A. E. RUEHLI AND P. A. BRENNAN, *Efficient capacitance calculations for three-dimensional multiconductor systems*, IEEE Trans. Microwave Theory and Techniques, 21 (1973), pp. 76–82.

[4] S. D. SENTURIA, R. M. HARRIS, B. P. JOHNSON, S. KIM, K. NABORS, M. A. SHULMAN, and J. K. White, *A computer-aided design system for microelectromechanical systems* (memcad), IEEE J. Microelectromechanical Systems, 1 (1992), pp. 3–13.

[5] J. N. NEWMAN, *Distributions of sources and normal dipoles over a quadrilateral panel*, J. Engrg. Math., 20 (1986), pp. 113–126.

[6] R. KRESS, *Linear Integral Equations*, Springer-Verlag, Berlin, 1989.

[7] 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.

[8] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[9] L. GREENGARD, *The Rapid Evaluation of Potential Fields in Particle Systems*, M.I.T. Press, Cambridge, MA, 1988.

[10] E. W. HOBSON, *The Theory of Spherical and Ellipsoidal Harmonics*, Chelsea, New York, 1955.

[11] 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.

[12] J. J. SAKURAI, *Modern Quantum Mechanics*, Addison-Wesley, Reading, MA, 1985.

[13] K. NABORS AND J. WHITE, *Fastcap: A multipole accelerated* 3-D *capacitance extraction program*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 10 (1991), pp. 1447–1459.

[14] K. NABORS, S. KIM, AND J. WHITE, *Fast capacitance extraction of general three-dimensional structures*, IEEE Trans. Microwave Theory and Techniques, 40 (1992), pp. 1496–1506.

[15] N. DEO, *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[16] S. A. VAVASIS, *Preconditioning for boundary integral equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 905–925.

[17] K. NABORS, S. KIM, J. WHITE, AND S. D. SENTURIA, *An adaptive multipole algorithm for* 3-D *capacitance calculation*, in Proc. Internat. Conf. Computer Design, Cambridge, MA, October 1991.

[18] B. JOHNSON, S. KIM, S. D. SENTURIA, AND J. WHITE, MEMCAD *capacitance calculations for mechanically deformed square diaphragm and beam microstructures*, in Proc. Transducers 91, San Francisco, CA, June 1991.

SIAM J. SCI. COMPUT.
Vol. 15, No. 3, pp. 736–754, May 1994

015

ITERATIVE SVD-BASED METHODS FOR ILL-POSED PROBLEMS*

C. R. VOGEL† AND J. G. WADE‡

Abstract. Very large matrices with rapidly decaying singular values commonly arise in the numerical solution of ill-posed problems. The singular value decomposition (SVD) is a basic tool for both the analysis and computation of solutions to such problems. In most applications, it suffices to obtain a partial SVD consisting of only the largest singular values and their corresponding singular vectors. In this paper, two separate approaches—one based on subspace iteration and the other based on the Lanczos method—are considered for the efficient iterative computation of partial SVDs. In the context of ill-posed problems, an analytical and numerical comparison of these two methods is made and the role of the regularization operator in convergence acceleration is explored.

Key words. ill-posed problems, regularization, singular value decomposition, subspace iteration, Lanczos method

AMS subject classifications. 65J20, 65F10, 65F15

1. Introduction.

An operator equation

$$\mathcal{A}(f) = g, \qquad f \in \mathcal{H}_1, \quad g \in \mathcal{H}_2 \tag{1.1}$$

is said to be ill posed in the sense of Hadamard if it is *not* the case that for each $g \in \mathcal{H}_2$, there exists a unique $f \in \mathcal{H}_1$ for which $\mathcal{A}(f) = g$, and the solution f is stable with respect to perturbations in g and the operator $\mathcal{A}$. These equations arise in a number of important applications, particularly in remote sensing and parameter estimation. See [3] and [7] for examples.

Arguably, most practical linear ill-posed problems have a formulation (1.1) where $\mathcal{H}_1$, $\mathcal{H}_2$ are separable Hilbert spaces and $\mathcal{A} : \mathcal{H}_1 \to \mathcal{H}_2$ is a compact linear operator. An important tool for both the analysis of such equations and the computation of their solutions is the singular value decomposition (SVD) (See [10] for a thorough discussion of computational methods and applications of the SVD in the finite dimensional case). If the spaces $\mathcal{H}_1$, $\mathcal{H}_2$ are infinite dimensional, the SVD is also commonly known as a singular system for the operator $\mathcal{A}$ (see [18] for details in the infinite dimensional case). In the nondegenerate infinite dimensional case, the singular values cluster at zero. This accounts for the severe ill conditioning of the finite dimensional approximations

$$Af = g, \qquad A : R^n \to R^m. \tag{1.2}$$

Particularly for applications in several space dimensions, m and n may be extremely large.

To obtain accurate approximate solutions to (1.2), some sort of regularization is required. Regularization can be viewed as replacing the ill-posed problem (1.1) by a "nearby" well-posed problem whose solution has certain desired features, such as smoothness. Most of the commonly used regularization methods for (1.2) have a very natural interpretation in terms of

*Received by the editors May 18, 1992; accepted for publication (in revised form) May 24, 1993.

†Department of Mathematical Sciences, Montana State University, Bozeman, Montana 59717 (vogel @math.montana.edu). This research was supported in part by National Science Foundation grant DMS-9106609 and by the Center for Interfacial Microbial Process Engineering at Montana State University, a National Science Foundation-sponsored Engineering Research Center, and the Center's Industrial Associates.

‡Department of Mathematics and Statistics, Bowling Green State University, Bowling Green, Ohio 43403-0221 (gwade@athena.bgsu.edu). This research was supported in part by Air Force Office of Scientific Research grant AFOSR-90-0091, National Science Foundation grants DMS-8818530 and DMS-8704169, and by Department of Energy contract SK966-19. Part of this work was carried out while this author was a visitor at the University of Southern California, Los Angeles, California, and at the Institute for Scientific Computation, Texas A&M University, College Station, Texas.

the SVD of $\mathcal{A}$. These methods filter out singular components of the solution corresponding to small singular values of $\mathcal{A}$ while retaining those singular components corresponding to large singular values. The SVD provides additional useful information. Given a particular level of noise in the data, the number of singular components that one can reasonable expect to retain can be used to quantify information content for problem (1.1) (see [8]). The rate of decay of the singular values plays an important role in this procedure. If the singular values decay to zero at a very rapid rate (e.g., they decay exponentially), the information content is typically quite low, and the problem is said to be *severely ill posed.* On the other hand, if the singular values decay slowly, the problem is said to be *mildly ill posed.* The SVD also plays an important role in the robust, efficient solution of constrained least squares problems and total least squares problems (see [10, Chap. 12]). Quadratically constrained least squares problems play a very important role in the computational solution of nonlinear ill-posed problems, since they arise in the implementation of so-called trust region, or hook step, methods [5], [25].

In each of the applications cited above, it suffices to compute a partial SVD consisting of only the largest singular values and the corresponding singular vectors. In this paper, two separate approaches to computing partial SVDs will be considered. The first approach is based on subspace iteration (see [23, Chap. 11]). This is also known as simultaneous iteration and as the block power method. Subspace iteration will henceforth be referred to as SI. In the symmetric case with block size p, SI yields approximations to the p largest eigenvalues together with their corresponding eigenvectors. This method can easily be adapted to compute the p largest singular values and corresponding singular vectors of a nonsymmetric (and often nonsquare) matrix. The second approach to be considered is based on the Lanczos method (see [10] and [23, Chap. 13]), and is essentially the approach outlined by O'Leary and Simmons in [22]. It is the property of the early iterates to converge quickly to the largest singular components, which makes both methods well suited for computing partial SVDs of matrices with rapidly decaying singular values. It will also be demonstrated in this paper that certain regularization techniques have the effect of accelerating the convergence of both SI and Lanczos.

In the following section, properties of the SVD of a compact linear operator are reviewed. Also discussed are regularization techniques and a posteriori error indicators and their representation and interpretation in terms of the SVD. Section 3 deals with SI for computing a partial SVD of a matrix. In §4 the Lanczos method and its implementation for computing a partial SVD are reviewed. Included in §§3 and 4 are asymptotic convergence results, a theoretical comparison, and a discussion of the role that regularization plays in accelerating the convergence of these two methods. Section 5 contains results of numerical experiments that compare SI with the Lanczos method. Also included in §5 are conclusions based on the analysis and numerical experiments.

2. The SVD and regularization. Let $\mathcal{H}_1, \mathcal{H}_2$ be separable Hilbert spaces and let $\mathcal{A} : \mathcal{H}_1 \to \mathcal{H}_2$ be a compact linear operator. An SVD, or singular system, for $\mathcal{A}$ is a sequence of triples $\{u_j, \sigma_j, v_j\}$ with the following properties:

(i) The sequence of right singular vectors $\{v_j\}$ forms an orthonormal basis for $N(\mathcal{A})^{\perp}$, the orthogonal complement of the null space of $\mathcal{A}$ in $\mathcal{H}_1$.

(ii) The sequence of left singular vectors $\{u_j\}$ forms an orthonormal basis for $\overline{R(\mathcal{A})}$, the closure of the range of $\mathcal{A}$ in $\mathcal{H}_2$.

(iii) The sequence of positive real numbers $\{\sigma_j\}$ consists of the nonzero singular values of $\mathcal{A}$, in decreasing order. If this sequence is infinite, then

$$\lim_{j\to\infty} \sigma_j = 0. \tag{2.1}$$

(iv) The left and right singular vectors are related via

$$\mathcal{A}v_j = \sigma_j u_j, \tag{2.2}$$

$$\mathcal{A}^* u_j = \sigma_j v_j. \tag{2.3}$$

A singular system can be constructed from the eigensystems of the selfadjoint, positive semidefinite compact linear operators $\mathcal{A}^*\mathcal{A}$ and $\mathcal{A}\mathcal{A}^*$. Let $\{\lambda_j\}$ denote the nonzero (i.e., positive) eigenvalues of $\mathcal{A}^*\mathcal{A}$, which are the same as the nonzero eigenvalues of $\mathcal{A}\mathcal{A}^*$. Assume these are in decreasing order and that the sequence includes multiplicities (if they exist), so that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots > 0$. Then

$$\sigma_j = \sqrt{\lambda_j}. \tag{2.4}$$

The corresponding eigenvectors of $\mathcal{A}^*\mathcal{A}$ are the right singular vectors v_j, while the corresponding eigenvectors of $\mathcal{A}\mathcal{A}^*$ are the left singular vectors u_j. Given v_j, one can obtain u_j using (2.2). Similarly, given u_j, one can compute v_j via (2.3).

In the discussion below, $\langle \cdot, \cdot \rangle_i$ denotes the inner product in $\mathcal{H}_i$, $i = 1, 2$. Given $f = f_0 + \sum_j f_j v_j \in N(\mathcal{A}) \bigoplus N(\mathcal{A})^\perp$, where $f_j = \langle v_j, f \rangle_1$, one obtains $\mathcal{A}f = \sum_j f_j \sigma_j u_j$. Similarly, given $g = \sum_j g_j u_j + g_\perp \in \overline{R(\mathcal{A})} \bigoplus R(\mathcal{A})^\perp$, where $g_j = \langle u_j, g \rangle_2$, one formally obtains the pseudoinverse, or least squares minimum norm, solution to the compact operator equation $\mathcal{A}f = g$,

$$f_{\text{lsmn}} = \mathcal{A}^\dagger g = \sum_j \frac{g_j}{\sigma_j} v_j. \tag{2.5}$$

The pseudoinverse solution exists as an element of $\mathcal{H}_1$ if and only if the Picard condition

$$\sum_j \frac{|g_j|^2}{\sigma_j^2} < \infty \tag{2.6}$$

is satisfied, in which case it uniquely solves the constrained minimization problem

$$\min \|f_{ls}\|_{\mathcal{H}_1} \tag{2.7}$$

subject to

$$\|\mathcal{A}f_{ls} - g\|_{\mathcal{H}_2} = \inf_{f \in \mathcal{H}_1} \|\mathcal{A}f - g\|_{\mathcal{H}_2}. \tag{2.8}$$

Except in the degenerate case when the $R(\mathcal{A})$ is finite dimensional (and hence there are only finitely many positive singular values) the pseudoinverse operator $\mathcal{A}^\dagger$ is unbounded. This is an immediate consequence of (2.1) and (2.5).

To achieve stable, accurate approximate solutions to (1.1), one must apply regularization. Many standard regularization methods yield approximations with a representation

$$f_\alpha = \sum_j w(\sigma_j^2; \alpha) \frac{g_j}{\sigma_j} v_j. \tag{2.9}$$

The function w is referred to as a *spectral filter* function [8], [12]. It has the qualitative behavior

$$w(\sigma^2; \alpha) \approx \begin{cases} 0 & \text{if } \sigma^2 \ll \alpha, \\ 1 & \text{if } \sigma^2 \gg \alpha. \end{cases} \tag{2.10}$$

The regularization parameter α can be interpreted as a cutoff parameter. Hence, these regularization methods filter out singular components corresponding to singular values whose squares are much smaller than α while retaining those components that correspond to singular values whose squares are much larger than α. One important example is a method commonly known as the truncated singular value decomposition (TSVD) [2], where

$$w(\sigma^2;\alpha) = \begin{cases} 0 & \text{if } \sigma^2 \le \alpha, \\ 1 & \text{if } \sigma^2 > \alpha. \end{cases} \tag{2.11}$$

In this case, (2.9) can be expressed as

$$f_p = \sum_{j=1}^{p} \frac{g_j}{\sigma_j} v_j. \tag{2.12}$$

Here the integer p is the number of singular values exceeding $\sqrt{\alpha}$ and is referred to as the truncation level. Various iterative methods, including Landweber iteration [19] and the conjugate gradient method, have representations (2.9) where the spectral filter function is a polynomial in σ^2 and the iteration count plays the role of the regularization parameter. See [12] and [13] for details.

Perhaps the most popular regularization method is Tikhonov Regularization [11], where f_α is chosen to minimize the functional

$$T_\alpha(f) = \frac{1}{2}\|\mathcal{A}f - g\|^2 + \frac{\alpha}{2}\|\mathcal{B}f\|^2. \tag{2.13}$$

The operator $\mathcal{B}$ is referred to as the regularization operator. The minimizer of (2.13) has a representation

$$f_\alpha = \left(\mathcal{A}^*\mathcal{A} + \alpha\mathcal{B}^*\mathcal{B}\right)^{-1}\mathcal{A}^* g. \tag{2.14}$$

When the regularization operator $\mathcal{B} = I$, the identity on $\mathcal{H}_1$, (2.13) is in standard form and its minimizer has a representation (2.9) with

$$w(\sigma^2;\alpha) = \frac{\sigma^2}{\sigma^2 + \alpha}. \tag{2.15}$$

More generally, when $\mathcal{B}$ is invertible, through the change of variables

$$\tilde{f} = \mathcal{B}f, \tag{2.16}$$

$$\tilde{\mathcal{A}} = \mathcal{A}\mathcal{B}^{-1}, \tag{2.17}$$

one can convert (2.13) to standard form

$$\tilde{T}_\alpha(\tilde{f}) = \|\tilde{\mathcal{A}}\tilde{f} - g\|^2 + \alpha\|\tilde{f}\|^2.$$

In this case, the minimizer of (2.13) is

$$f_\alpha = \mathcal{B}^{-1}\tilde{f}_\alpha, \tag{2.18}$$

where

$$\begin{aligned} \tilde{f}_\alpha &= \arg\min_{\tilde{f}} \tilde{T}_\alpha(\tilde{f}) \\ &= \sum_j w(\tilde{\sigma}_j^2;\alpha)\frac{\tilde{g}_j}{\tilde{\sigma}_j}\tilde{v}_j. \end{aligned} \tag{2.19}$$

Here $\{\tilde{u}_j, \tilde{\sigma}_j, \tilde{v}_j\}$ is an SVD of $\tilde{\mathcal{A}} = \mathcal{A}\mathcal{B}^{-1}$, $\tilde{g}_j \stackrel{\text{def}}{=} \langle \tilde{u}_j, g\rangle_2$, and w is defined as in (2.15). The change of variables (2.18) and the representation (2.19) remain valid if other regularization methods, such as the TSVD or conjugate gradient iteration, are applied (see [13]).

It is often the case that the regularization operator $\mathcal{B}$ is not invertible, e.g., $\mathcal{H}_1 = L^2(\Omega)$ with

$$\|\mathcal{B}f\|^2 = \int_\Omega |\nabla f|^2 \, dx. \tag{2.20}$$

In this case $\mathcal{B}$ has a finite dimensional null space and a bounded pseudoinverse, and one can replace $\mathcal{B}^{-1}$ in (2.18) by the pseudoinverse $\mathcal{B}^\dagger$. See [6] for further details. An alternative approach described in [13] is based on the generalized singular value decomposition (GSVD) [10, p. 562] of the operator pair $\mathcal{A}$, $\mathcal{B}$. When using either approach, the representation (2.19) must be modified to account for the nontrivial null space of $\mathcal{B}$. For simplicity, it is assumed for the remainder of this paper that $\mathcal{B}$ is invertible.

In practice, one is given error contaminated data

$$g = \mathcal{A}f_{\text{exact}} + \epsilon,$$

and the selection of the regularization parameter α becomes a very important issue. A variety of a posteriori methods are available for choosing α. Among these are generalized cross validation (GCV) [26] and the L-curve approach [15]. Both these methods require computation of functions that reflect the behavior of the regularized solution error

$$e(\alpha) = \|\mathcal{B}(f_\alpha - f_{\text{exact}})\| = \|\tilde{f}_\alpha - \tilde{f}_{\text{exact}}\|. \tag{2.21}$$

Both methods have very efficient implementations in terms of the SVD of $\tilde{\mathcal{A}} = \mathcal{A}\mathcal{B}^{-1}$. For instance, the GCV function takes the form

$$V(\alpha) = \frac{\sum_j \left(1 - w(\tilde{\sigma}_j^2; \alpha)\right)^2 \tilde{g}_j^2}{\left(\sum_j (1 - w(\tilde{\sigma}_j^2; \alpha))\right)^2}. \tag{2.22}$$

The minimizer of the GCV function is an estimate for the minimizer α^* of (2.21). The L-curve approach requires component functions

$$\begin{aligned} X(\alpha) &= \log \|\mathcal{A}f_\alpha - g\|^2 \\ &= \log \sum_j \left(1 - w(\tilde{\sigma}_j^2; \alpha)\right)^2 \tilde{g}_j^2 \end{aligned} \tag{2.23}$$

and

$$\begin{aligned} Y(\alpha) &= \log \|\mathcal{B}f_\alpha\|^2 = \log \|\tilde{f}_\alpha\|^2 \\ &= \log \textstyle\sum_j w(\tilde{\sigma}_j^2; \alpha)^2 \frac{\tilde{g}_j^2}{\tilde{\sigma}_j^2}. \end{aligned} \tag{2.24}$$

One plots $Y(\alpha)$ versus $X(\alpha)$ for a range of values of α. The resulting curve often has a characteristic L-shape, and the corner of the L is an estimate for α^*. When the regularization parameter α is continuous, as is the case with Tikhonov Regularization, one may identify the corner by extremizing the curvature function

$$\kappa(\alpha) = \frac{X''(\alpha)Y'(\alpha) - X'(\alpha)Y''(\alpha)}{\left(X'(\alpha)^2 + Y'(\alpha)^2\right)^{3/2}}. \tag{2.25}$$

If α is discrete, as in the case of TSVD or various iterative methods, κ may be approximated using finite differences, or interpolated and then differentiated as in [17].

Because of the fact that $w(\sigma^2; \alpha) \approx 0$ for small values of σ, each of the above expressions (2.19), (2.22), (2.23), and (2.24) can be closely approximated using a partial SVD consisting of only the largest singular values and their corresponding singular vectors. This motivates our interest in iterative methods for the efficient solution of a partial SVD for very large ill-conditioned matrices.

3. SI for computing a partial SVD. The truncated sequence of singular components $\{(u_j, \sigma_j, v_j)\}_{j=1}^{p}$ will be referred to as a partial SVD with truncation level p. Assume that $\sigma_1 \geq \cdots \geq \sigma_p > 0$. SI will be applied to compute a partial SVD of an $m \times n$ matrix A, or more generally, a partial SVD of $\tilde{A} \stackrel{\text{def}}{=} AB^{-1}$, where B is a discretization of the regularization operator $\mathcal{B}$, cf., (2.16)–(2.19). Motivated by the procedure for computing an SVD of A from an eigendecomposition of A^TA outlined in §2, let $\overline{A} \stackrel{\text{def}}{=} A^TA$. Note that $\overline{A}$ is $n \times n$, symmetric, and positive semidefinite. Following Parlett [23], let $V^{(0)}$ be an $n \times p$ matrix whose columns are orthonormal, let $\mathcal{S}_p$ denote the subspace spanned by the columns of $V^{(0)}$, and let $\overline{A}^k\mathcal{S}_p$ denote the subspace spanned by the columns of $V^{(k)} = \overline{A}^kV^{(0)}$. The integer k is the iteration count. SI consists of projecting the operator $\overline{A}$ onto the p-dimensional subspace $\overline{A}^k\mathcal{S}_p$, and then computing the eigensystem of the resulting $p \times p$ matrix. To be more precise, let $V^{(k)}$ be the $n \times p$ matrix, the columns of which are the current approximations of the eigenvectors corresponding to the p largest eigenvalues of $\overline{A}$. Then:

1. Let Q be an orthonormalization of $\overline{A}V^{(k)}$.
2. Define $H = Q^T\overline{A}Q$. This $p \times p$ matrix represents the Rayleigh–Ritz projection of $\overline{A}$ onto $\overline{A}^k\mathcal{S}_p$.
3. Compute the eigendecomposition $G\Delta G^T = H$.
4. Set $V^{(k+1)} = QG$. This amounts to representing the eigenvectors G in the larger space ($\mathbf{R}^n$) in which $\overline{A}$ is defined.

The diagonal entries of the matrix Δ in step 3 are the Ritz values. These approximate the p largest eigenvalues of $\overline{A}$. The vectors comprising the columns of $V^{(k)}$ are known as Ritz vectors and are estimates for the corresponding eigenvectors.

The algorithm just described, although conceptually transparent, has the disadvantage that an additional matrix-vector multiplication is required in step 2. A clever variant of this algorithm, which is mathematically equivalent but avoids the additional multiplication, is outlined in [23, p. 293]. This variant is adopted here, with modifications to account for the fact the singular values/vectors of a rectangular matrix, rather than eigenvalues/vectors of a symmetric matrix, are sought.

3.1. An SI algorithm for computing a partial SVD. Let A be an $m \times n$ real-valued matrix, and let $V^{(0)}$ be an $n \times p$ matrix having orthonormal columns. Compute the matrix product $U^{(0)} := AV^{(0)}$. Initialize the iteration count $k = 0$, and for $j = 1, 2, \ldots, p$, set $\sigma_j^{(0)} = 0$. Specify a stopping tolerance TOL.

1. $k := k + 1$.
2. $C^{(k)} := A^TU^{(k-1)}$.
3. Compute a QR factorization, $Q^{(k)}R^{(k)} = C^{(k)}$.
4. Compute an SVD

$$W^{(k)}\Sigma^{(k)}\left(P^{(k)}\right)^T = \left(R^{(k)}\right)^T.$$

The diagonal entries $\sigma_j^{(k)}$, $1 \leq j \leq p$, of $\Sigma^{(k)}$ are the square roots of the Ritz values of $\overline{A}$ and are estimates for the p largest singular values of A.

5. $V^{(k)} := Q^{(k)} P^{(k)}$. The columns of $V^{(k)}$ are the Ritz vectors of $\overline{A}$ and are estimates for the first p right singular vectors of A.

6. $U^{(k)} := AV^{(k)}$. The columns of $U^{(k)}$ are estimates for the first p left singular vectors.

7. Check for convergence. Stop if

$$\frac{|\sigma_j^{(k)} - \sigma_j^{(k-1)}|}{\sigma_j^{(k)}} \leq \text{TOL} \quad \text{for } j = 1, 2, \ldots, p. \tag{3.1}$$

Otherwise, goto step 1.

The stopping criterion (3.1) is applied rather than a more conventional convergence check like

$$\left| \frac{\left(u_j^{(k)}\right)^T A v_j^{(k)}}{\|u_j^{(k)}\| \, \|v_j^{(k)}\| \, \sigma_j^{(k)}} - 1 \right| \leq \text{TOL}$$

to minimize the number of evaluations of the operator A. Step 4 was recommended by P C. Hansen. It replaces steps (c) $H = R^{(k)}(R^{(k)})$ and (d) $P^{(k)}(\Sigma^{(k)})^2(P^{(k)})^T = H$ in Table 14-2-4 of [23]. This modification substantially reduces computational overhead and yields much more accurate singular value estimates.

By modifying results in [23] to account for the fact that one seeks singular values rather than eigenvalues, one obtains the following convergence result.

THEOREM 3.1. *Let the $p+1$ largest singular values of the matrix A satisfy*

$$\sigma_1 \geq \sigma_2 \geq \cdots \sigma_p > \sigma_{p+1} > 0. \tag{3.2}$$

Then the relative errors for the singular value approximations obtained by the above SI algorithm converge to zero at the asymptotic rate

$$\frac{\sigma_j - \sigma_j^{(k)}}{\sigma_j} = \mathcal{O}\left(\left(\frac{\sigma_{p+1}}{\sigma_j}\right)^{4k}\right), \qquad j = 1, 2, \ldots, p. \tag{3.3}$$

The angle between the spaces spanned by the true and the approximate right singular vectors converges to zero at the rate

$$\angle(\mathbf{v}_j, \text{span}\{\mathbf{v}_j^{(k)}\}) = \mathcal{O}\left(\left(\frac{\sigma_{p+1}}{\sigma_j}\right)^{2k}\right), \qquad j = 1, 2, \ldots, p. \tag{3.4}$$

An analogous result holds for the left singular vectors.

Proof. Equation (3.4) follows immediately from the theorems in [23, pp. 297–298] and the fact that the eigenvalues of $A^T A$ are the squares of the singular values of A. Note that from (3.2) the eigenvalues are decreasing (rather than increasing) and one desires the p largest (rather than smallest) eigenvalues. Since the left singular vectors are eigenvectors of AA^T and both $A^T A$ and AA^T have the same nonzero eigenvalues, the analogous result for left singular vectors follows by the same arguments.

To verify (3.3), let $\lambda_j, \mathbf{v}_j$ be an eigenpair for $\overline{A} \overset{\text{def}}{=} A^T A$ and let $\lambda_j^{(k)}, \mathbf{v}_j^{(k)}$ be a Ritz value and corresponding Ritz vector for $\overline{A}$. Assume that $\|\mathbf{v}_j\| = \|\mathbf{v}_j^{(k)}\| = 1$ and

$$\cos^{-1}(\mathbf{v}_j^T \mathbf{v}_j^{(k)}) = \angle(\mathbf{v}_j, \text{span}\{\mathbf{v}_j^{(k)}\}) \overset{\text{def}}{=} \theta_j. \tag{3.5}$$

Then

$$\|\mathbf{v}_j - \mathbf{v}_j^{(k)}\|^2 = 2 - 2\cos\theta_j. \tag{3.6}$$

Consequently,

$$\begin{aligned} |\lambda_j - \lambda_j^{(k)}| &= |\mathbf{v}_j^T \overline{A}\mathbf{v}_j - (\mathbf{v}_j^{(k)})^T \overline{A}\,\mathbf{v}_j^{(k)}| \\ &= |(\mathbf{v}_j - \mathbf{v}_j^{(k)})^T \overline{A}\,(\mathbf{v}_j - \mathbf{v}_j^{(k)}) - 2\mathbf{v}_j^T \overline{A}\,(\mathbf{v}_j - \mathbf{v}_j^{(k)})| \\ &\le \|\overline{A}\|\,\|\mathbf{v}_j - \mathbf{v}_j^{(k)}\|^2 + 2\lambda_j|\mathbf{v}_j^T(\mathbf{v}_j - \mathbf{v}_j^{(k)})| \\ &\le 4\,\|\overline{A}\|\,(1 - \cos\theta_j). \end{aligned} \tag{3.7}$$

But $\cos\theta = 1 - \theta^2/2 + \mathcal{O}(\theta^4)$, so $|\lambda_j - \lambda_j^{(k)}| = \mathcal{O}(\theta_j^2)$. From (3.4) and (3.5) above,

$$\theta_j = \mathcal{O}\left(\left(\frac{\sigma_{p+1}}{\sigma_j}\right)^{2k}\right). \tag{3.8}$$

Combining this with

$$\left|\frac{\sigma_j - \sigma_j^{(k)}}{\sigma_j}\right| \le \frac{|\sigma_j + \sigma_j^{(k)}|}{\sigma_j^2}\,|\sigma_j - \sigma_j^{(k)}| \le \frac{1}{\lambda_{p+1}}|\lambda_j - \lambda_j^{(k)}| \tag{3.9}$$

gives the desired result. □

By combining SI with the regularization techniques of the previous section, cf., (2.16)–(2.19), one obtains the following algorithm for computing approximate regularized solutions f_α to the operator equation (1.1).

3.2. An SI based regularization algorithm.

1. Apply SI to obtain a partial SVD $\{\tilde{\mathbf{u}}_j, \tilde{\sigma}_j, \tilde{\mathbf{v}}_j\}_{j=1}^p$ of $\tilde{A} \stackrel{\text{def}}{=} AB^{-1}$.
2. Select a regularization parameter α using the techniques described at the end of §2.
3. Compute transformed regularized solutions

$$\tilde{f}_\alpha = \sum_{j=1}^{p} w(\tilde{\sigma}_j^2; \alpha)\frac{\tilde{g}_j}{\tilde{\sigma}_j}\tilde{\mathbf{v}}_j, \quad \text{where } \tilde{g}_j = \langle \tilde{\mathbf{u}}_j, \mathbf{g}\rangle_2,$$

4. Back transform to obtain $f_\alpha = B^{-1}\tilde{f}_\alpha$.

Note that if one sets $w(\sigma^2; \alpha) = 1$ for all α, one obtains the TSVD regularization method (cf., (2.11)) with truncation level p. In certain cases, this may suffice. In other cases, further filtering of singular components may be required. Also note that the matrix AB^{-1} need not be explicitly computed. All that is required is the application of the operators A and B^{-1} and their adjoints to a sequence of vectors. Obviously, step 4 is usually implemented by solving the system $Bf_\alpha = \tilde{f}_\alpha$.

3.3. The effect of regularization on convergence. In typical applications, the regularization operator $\mathcal{B}$ is a differential operator whose inverse is compact. Hence, the singular values of $\mathcal{A}\mathcal{B}^{-1}$ are likely to decay more rapidly than those of $\mathcal{A}$. Provided an accurate approximation scheme is used, the singular values of the matrix AB^{-1} will then have a larger relative separation than those of A. As a consequence of Theorem 3.1, SI will converge much more rapidly when it is applied to AB^{-1} than when it is applied to A. In the sense of modifying the spectrum of the operator to accelerate convergence, regularization plays a role in SI that

is analogous to that of a preconditioner in the iterative solution of linear systems. When applying standard preconditioners, one effectively premultiples and postmultiplies A to reduce the condition number of the resulting matrix. Preconditioning can be used for ill-conditioned problems (see [14]), but, as pointed out in [13], this approach is inappropriate unless care is taken to *not* invert small singular values.

Regularization also differs from standard preconditioning in that the solution to the regularized problem is *not* the same as the solution to the unregularized problem. Regularization with a differential operator tends to produce smoother singular vectors, and hence, smoother approximate solutions than in the absence of regularization. While the regularized SI is more efficient, one may experience a loss of resolution due to an inability to approximate "rough" solutions with "smooth" basis functions.

4. Lanczos iteration for SVD computations. As pointed out by Parlett in [23, Chap. 13], each iteration k of the Lanczos method for computing eigenvalues of a symmetric matrix C may be viewed as a Rayleigh–Ritz projection of the operator C onto the kth Krylov subspace

$$\mathcal{K}^k(q_1) = \text{span}\{q_1, Cq_0, \ldots, C^{k-1}q_1\}, \tag{4.1}$$

where q_1 is an initial unit vector. The matrix representing this projection is a $k \times k$ symmetric tridiagonal matrix T_k. The eigenvalues of T_k, which are the Ritz values, approximate certain eigenvalues of C. The corresponding eigenvectors of T_k can be used to compute approximate eigenvectors of C.

To compute a sequence of approximate partial SVDs of an $m \times n$ matrix A, one can apply the Lanczos method to the $(m+n) \times (m+n)$ symmetric matrix

$$C = \begin{bmatrix} 0_{m\times m} & A \\ A^T & 0_{n\times n} \end{bmatrix}. \tag{4.2}$$

The spectrum of C is real and symmetric about the origin, with the nonzero singular values of A comprising the positive eigenvalues of C. It is the fact that the Lanczos Ritz values converge rapidly to the extremal eigenvalues of C that makes the Lanczos method appropriate for computing a partial SVD of A. However, in the presence of roundoff error, it is necessary either to guarantee the orthogonality of the Lanczos basis vectors q_i, or to detect the presence of "ghost" eigenvalues, which are spurious copies of convergent eigenvalues. With reorthogonalization, one obtains the following algorithm (see [10, p. 499] or [22]).

4.1. A Lanczos algorithm for computing a partial SVD. Let A be an $m \times n$ real-valued matrix, and let q_1 be an $n \times 1$ unit vector. Compute

$$y := Aq_1, \quad \alpha_1 := \|y\|, \quad u_1 := y/\alpha_1.$$

Define $Q^{(k)} = [q_1, q_2, \ldots, q_k]$ and $U^{(k)} = [u_1, u_2, \ldots, u_k]$.
For iterations $k = 1, 2, \ldots,$

1. $w := A^T u_k - \alpha_k q_k$
2. Reorthogonalize. $w := w - Q^{(k)}\left((Q^{(k)})^T w\right)$
3. $\beta_k := \|w\|$
4. $q_{k+1} := w/\beta_k,$
5. $y := Aq_{k+1} - \beta_k u_k,$
6. Reorthogonalize. $y := y - U^{(k)}\left((U^{(k)})^T y\right)$
7. $\alpha_{k+1} := \|y\|$
8. $u_{k+1} := y/\alpha_{k+1}$

end

For each k, estimates for the singular values and corresponding singular vectors of A can be obtained as follows: Let $\tilde{T}_k$ denote the $k \times k$ upper bidiagonal matrix comprised of the α_j's on the main diagonal and the β_j's on the upper subdiagonal.

8a. Compute the SVD $\tilde{U}^{(k)}\Sigma^{(k)}(\tilde{Q}^{(k)})^T = \tilde{T}_k$

8b. $U := U^{(k)}\tilde{U}^{(k)}$

8c. $V := Q^{(k)}\tilde{Q}^{(k)}$

The diagonal entries $\sigma_1^{(k)}, \ldots, \sigma_k^{(k)}$ of $\Sigma^{(k)}$ are approximations to the k largest singular values of A. The columns of U and V approximate the corresponding left and right singular vectors.

4.2. Stopping conditions, regularization. The following strategy is applied when using the above Lanczos algorithm to compute a partial SVD with truncation level p: For iterations $k = 1, \ldots, p$, perform only steps 1–8 of the above algorithm. For iterations $k > p$, perform, in addition, step 8a and the convergence check (3.1). If condition (3.1) holds, then perform steps 8b–8c, and stop.

The previous SI based regularization algorithm can easily be modified for Lanczos iteration. In step 1, simply replace "SI" with "Lanczos iteration."

Using the results of Saad [24], one obtains the following convergence result.

THEOREM 4.1. *Let the $p+1$ largest singular values of the matrix A satisfy* (3.2). *Then the singular value approximations obtained by the above Lanczos algorithm satisfy*

$$0 \le \frac{\sigma_j - \sigma_j^{(k)}}{\sigma_j} \le M_j\left(\overline{\gamma}_j + \sqrt{\overline{\gamma}_j^2 - 1}\right)^{-2(k-j)}, \quad 1 \le j \le p,\ k > j, \tag{4.3}$$

where M_j is a constant (depending on the angle between q_1 and v_j and on the separation of $\sigma_1, \sigma_2, \ldots, \sigma_j$), and

$$\overline{\gamma}_j \stackrel{\text{def}}{=} 2\left(\frac{\sigma_j^2}{\sigma_{j+1}^2}\right) - 1. \tag{4.4}$$

Proof. As noted in [10, p. 499] (see also [9]), it is not difficult to show that

$$\begin{aligned} AQ^{(k)} &= U^{(k)}\tilde{T}_k, \\ A^T U^{(k)} &= Q^{(k)}\tilde{T}_k^T. \end{aligned}$$

Applying A^T to the first of these and substituting into the second leads to

$$\begin{aligned} A^T A Q^{(k)} &= A^T U^{(k)}\tilde{T} \\ &= Q^{(k)}\tilde{T}_k^T\tilde{T}_k. \end{aligned} \tag{4.5}$$

Since $\tilde{T}_k^T\tilde{T}_k$ is symmetric tridiagonal, it follows that (4.5) is a matrix representation of the Lanczos method applied to A^TA with initial vector q_1. As before, let $\lambda_j = \sigma_j^2$ be the jth eigenvalue of A^TA, and now let $\lambda_j^{(k)} = (\sigma_j^{(k)})^2$ be the jth eigenvalue of $\tilde{T}_k^T\tilde{T}_k$. Since $\lambda_{\min} = \sigma_n^2 > 0, \gamma_j \stackrel{\text{def}}{=} 1+2(\lambda_j-\lambda_{j+1})/(\lambda_{j+1}-\lambda_{\min}) \ge \overline{\gamma}_j$. Then by Theorem 2 of [24] and the remarks following it, together with the Chebyshev polynomial inequality $T_i(\gamma) \ge (\gamma + \sqrt{\gamma^2-1})^i/2$,

$$0 \le \lambda_j - \lambda_j^{(k)} \le M_j\lambda_j\left(\overline{\gamma}_j + \sqrt{\overline{\gamma}_j^2 - 1}\right)^{-2(k-j)}. \tag{4.6}$$

Combining (3.9) with (4.6) yields (4.3). □

4.3. A theoretical comparison of Lanczos and SI. For most practical ill-posed problems (e.g., coefficient identification problems or Fredholm first-kind integral equations that occur in remote sensing), the matrix A is not sparse and the cost of applying A and A^T far exceeds other computational costs in the implementation of SI with block size $p \ll n$. See [25] for a detailed discussion. The same should hold for Lanczos when $k \ll n$ iterations are applied. Additional computational costs will be referred to as "overhead."

Each SI with block size p requires the same number of evaluations of the operators A and A^T as p iterations of the Lanczos algorithm. In contrast to SI, in the absence of computer roundoff error the three-term Lanczos recursion automatically generates an orthonormal basis $\{q_1, q_2, \ldots, q_k\}$ for the kth Krylov subspace. The Lanczos projection matrix $\tilde{T}_k$ is upper bidiagonal, while the SI projection matrix $R^{(k)}$ is upper triangular. Hence, in the absence of roundoff, p Lanczos iterations require much less storage and overhead than does a single SI. However, due to the combined effects of roundoff and very rapid convergence (see [23, p. 270]), we have found reorthogonalization to be necessary to increase the robustness of the Lanczos algorithm and to substantially decrease the number of operator evaluations. Reorthogonalization negates much of the advantage of Lanczos in terms of storage and overhead.

To interpret the results of Theorems 3.1 and 4.1, assume that

$$\sigma_j^2/\sigma_{j+1}^2 \geq 1 + c, \quad 1 \leq j \leq p, \tag{4.7}$$

where $c > 1$. This assumption holds for many ill-posed problems. In particular, if the singular values decay exponentially, (4.7) holds with $p = n$. From (3.3) the relative error in σ_j at iteration k for SI is of the form

$$\frac{\sigma_j - \sigma_j^{(k)}}{\sigma_j} = \mathcal{O}(r_j^k), \quad 1 \leq j \leq p, \quad k = 1, 2, \ldots, \tag{4.8}$$

where

$$r_j = (\sigma_{p+1}/\sigma j)^4 \leq (1 + c)^{-2(p+1-j)}. \tag{4.9}$$

Thus one should expect extremely rapid convergence for the largest singular values, with a marked decrease in the convergence rates of the smaller singular values.

Using (4.7), one can show from (4.3) and (4.4) that when $p \ll n$, the Lanczos approximations satisfy

$$\frac{\sigma_j - \sigma_j^{(k)}}{\sigma_j} = \mathcal{O}(r^{k-j}), \qquad 1 \leq j \leq p, \ k > j, \tag{4.10}$$

where

$$r = (1 + 4c)^{-2}. \tag{4.11}$$

In this case one should expect a high degree of accuracy when the iteration count k is much larger than the index j. Moreover, since one SI requires as many operator evaluations as p Lanczos iterations, Lanczos should require far fewer operator evaluations than SI to obtain σ_j for $j \ll p$ when relatively large block sizes p are used.

Finally, since applying a regularization operator (cf., (2.15)) and TSVD (cf., (2.11)) may increase the decay rate of the singular values, it may decrease both the r_j in (4.8) and r in (4.10). Hence one would expect more rapid convergence for both methods when a regularization operator is applied.

5. A numerical comparison. Consider the linear ill-posed problem arising in the numerical second differentiation of discrete, one-dimensional data. While this is not a large scale problem, the spectrum of the operator is known, and hence numerical approximations can be compared with exact values. Moreover, the performance of the iterative SVD methods for this problem is consistent with that observed in numerical experiments with much larger systems arising in multidimensional applications. Given observations of a function $-g(x)$ at equispaced points x_i in the interval [0, 1], one wishes to estimate the second derivative $f = -g''$. Assume in addition that $g(0) = g(1) = 0$. Then f and g satisfy the Fredholm first-kind integral equation

$$g(x) = \mathcal{A}f(x) = \int_0^1 a(x, y)\, f(y)\, dy, \qquad 0 \le x \le 1, \tag{5.1}$$

where the kernel, $a(x, y)$, is the Green's function for the differential operator

$$\mathcal{L}u = -u'', \quad 0 < x < 1, \quad u(0) = u(1) = 0. \tag{5.2}$$

In the space $\mathcal{H} = L^2(0, 1)$, $\mathcal{L}$ has eigenvalues $\lambda_j = (\pi j)^2$ and corresponding orthonormal eigenfunctions $\phi_j(x) = \frac{1}{2}\sin(\pi j x)$, $j = 1, 2, \ldots$. Let A be the inverse of the $n \times n$ tridiagonal matrix L obtained from the standard finite difference discretization of (5.2). L has diagonal entries $2/h^2$ and subdiagonal and superdiagonal entries $-\frac{1}{h}$, where $h = 1/(n + 1)$. The singular values of A, which are the reciprocals of the eigenvalues of L, are known to be [1, p. 57]

$$\sigma_j = \left(\frac{4}{h^2}\sin^2(j\pi h/2)\right)^{-1} \approx 1/(\pi j)^2 \quad \text{for } j \ll n. \tag{5.3}$$

Since the singular value decay rate is relatively slow, this problem is classified as mildly ill posed [8].

Consider the regularization operator $\mathcal{B} = \mathcal{L}^{\beta/2}$, where $\beta \ge 0$. This choice is convenient for the present numerical experiments since the spectra of the operators $\mathcal{A}$ and $\mathcal{B}$ are known and easily related. When $\beta = 1$, this corresponds to first derivative regularization, cf. (2.20). Approximate $\mathcal{B}$ by the matrix $B = L^{\beta/2}$. Given the eigendecomposition $L = V\ \mathrm{diag}(\lambda_j)\, V^T$, where $\lambda_j = 1/\sigma_j$, one can easily compute

$$B = L^{\beta/2} = V\ \mathrm{diag}(\lambda_j^{\beta/2})\, V^T. \tag{5.4}$$

Note that B is invertible and that the matrix $\tilde{A} = AB^{-1}$ has singular values

$$\tilde{\sigma}_j = \sigma_j^{1+\beta/2}. \tag{5.5}$$

By adjusting β, one can control the rate of decay of the singular values and study the effect on the performance of iterative SVD methods. In the experiments described below, we applied both SI and Lanczos iteration to compute partial SVDs of $\tilde{A}$. The convergence rates of the two methods are first examined. Then the SVD approximations are used in the solution of the ill-posed problem described above. These experiments were performed using the PRO-MATLAB [20] software package on a DECSTATION 5000/200 workstation. Double precision machine epsilon is $2^{-52} \approx 2.2204 \times 10^{-16}$. The matrix size is $n = 100$, and the initial vectors w and the columns of $V^{(0)}$ are taken to be random vectors.

Table 1 provides a comparison of the two methods when the regularization index $\beta = 2$. Column 2 contains the first 20 singular values $\tilde{\sigma}_j$ of $\tilde{A}$, cf., (5.3) and (5.5). Column 3 contains

the singular value approximations obtained using $k = 2$ iterations of the SI algorithm of §3 with block size $p = 20$. The approximations in column 4 were obtained using 20 iterations of the Lanczos algorithm of §5. Note that the first 10 SI approximates are (to within five decimal digits relative accuracy) indistinguishable from the true singular values. There is a gradual loss of accuracy as the singular values decrease. This is consistent with (4.8). In contrast, the first 13 Lanczos approximates are indistinguishable from the true values, but there is a very rapid decrease in the accuracy of singular values σ_j for which $20 - j$ is small. This is consistent with (4.10).

TABLE 1
Comparison between methods.

index j	Singular Value $\tilde{\sigma}_j$	SI approx.	Lanczos approx. with reorthog.	Lanczos approx. without reorthog.
1	1.0268e–02	1.0268e–02	1.0268e–02	1.0268e–02
2	6.4204e–04	6.4204e–04	6.4204e–04	1.0268e–02
3	1.2692e–04	1.2692e–04	1.2692e–04	1.0268e–02
4	4.0205e–05	4.0205e–05	4.0205e–05	1.0268e–02
5	1.6492e–05	1.6492e–05	1.6492e–05	1.0268e–02
6	7.9674e–06	7.9674e–06	7.9674e–06	5.2964e–03
7	4.3096e–06	4.3096e–06	4.3096e–06	6.4204e–04
8	2.5324e–06	2.5324e–06	2.5324e–06	6.4204e–04
9	1.5853e–06	1.5853e–06	1.5853e–06	6.4204e–04
10	1.0433e–06	1.0433e–06	1.0433e–06	1.2692e–04
11	7.1501e–07	7.1499e–07	7.1501e–07	1.2692e–04
12	5.0672e–07	5.0667e–07	5.0672e–07	4.0205e–05
13	3.6938e–07	3.6937e–07	3.6938e–07	3.9435e–05
14	2.7583e–07	2.7576e–07	2.7582e–07	1.6492e–05
15	2.1029e–07	2.1027e–07	2.1029e–07	7.9674e–06
16	1.6326e–07	1.6239e–07	1.6141e–07	4.3096e–06
17	1.2879e–07	1.2729e–07	1.1971e–07	2.5323e–06
18	1.0305e–07	1.0261e–07	9.1849e–08	1.5834e–06
19	8.3511e–08	6.9451e–08	4.8531e–08	6.5545e–07
20	6.8452e–08	6.2450e–08	6.6155e–09	7.3456e–08

The approximations in column 5 were also obtained using the Lanczos algorithm, except that the reorthogonalization steps 2 and 6 were omitted. In this case, seven of the first 20 approximates can be positively identified as spurious copies of true singular values. An additional four do not seem to correspond to anything. Only the first seven singular values are approximated to within five digits (relative) accuracy.

In terms of the number of evaluations of the operators A and A^T, twice as much work was required to generate the SI approximations in this table as was required to generate the Lanczos approximations.

Figure 1 shows the effect on SI of varying the regularization index β. The SI count is fixed at $k = 2$. Note that there is a substantial increase in the convergence rate with increased β. In addition, the larger singular values are approximated much more accurately than the smaller ones. Both these observations are consistent with (4.8) when $r_j = (\tilde{\sigma}_{p+1}/\tilde{\sigma}_j)^4 \approx (j/(p+1))^{(2+\beta)4k}$.

Figure 2 shows that increasing the regularization index β also increases the convergence rate for Lanczos, but the effect is not as pronounced as with SI. Also, a comparison of Figs. 1 and 2 shows that the Lanczos estimates of the larger singular values exhibit extremely high relative accuracy, while the corresponding SI estimates are only moderately accurate. As was the case with Table 1, twice as many operator evaluations were used to generate Fig. 1 (SI

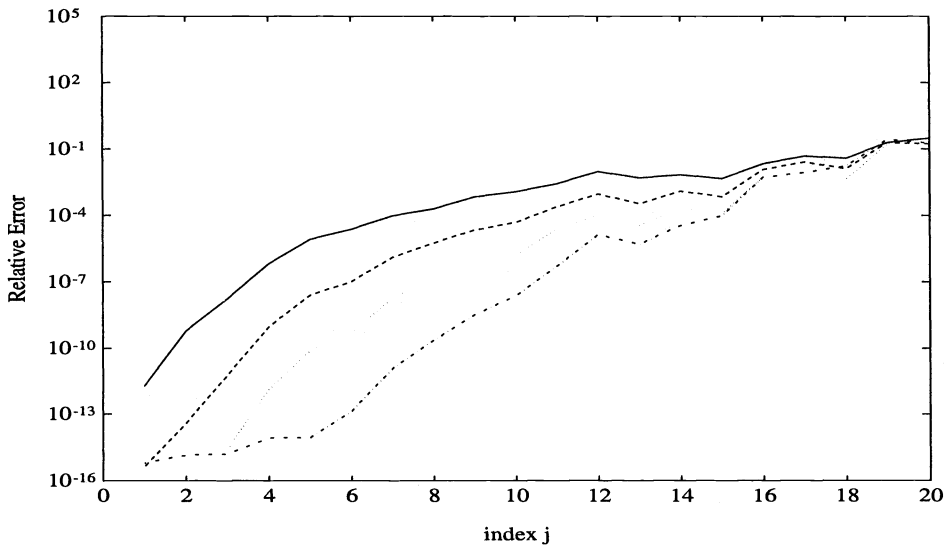


FIG. 1. *Plots of relative error in SI singular value approximates for varying regularization index β. Solid line corresponds to $\beta = 0$; dashed line corresponds to $\beta = 1$; dotted line corresponds to $\beta = 2$; dotted-dashed line corresponds to $\beta = 4$.*

with $k = 2$) as were used to obtain Fig. 2 (Lanczos with $k = 20$).

Next, this partial SVD was applied to compute regularized solutions and error indicators

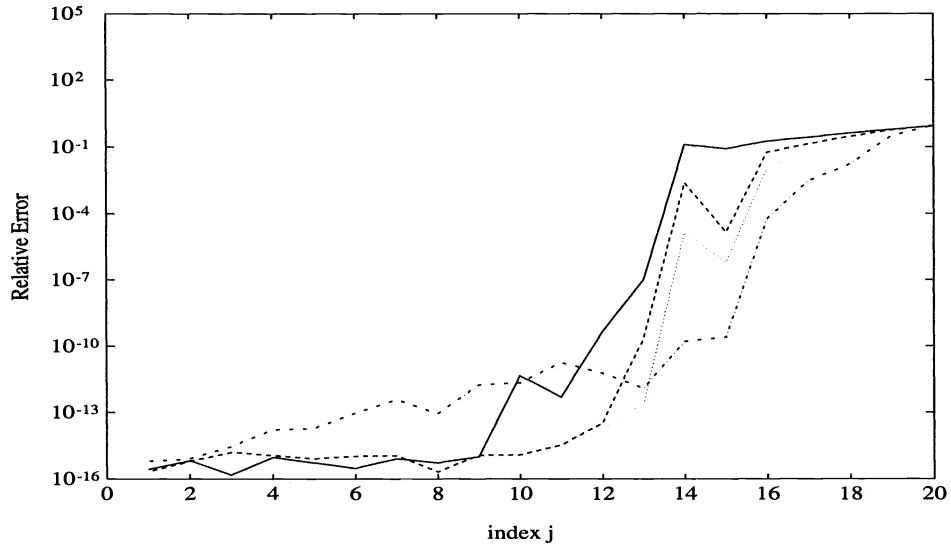


FIG. 2. *Plots of relative error in Lanczos singular value approximates for varying regularization index β. Solid line corresponds to $\beta = 0$; dashed line corresponds to $\beta = 1$; dotted line corresponds to $\beta = 2$; dotted-dashed line corresponds to $\beta = 4$.*

for the ill-posed problem (5.1). Synthetic data was generated by taking

$$\mathbf{g} = A\mathbf{f}_{\text{exact}} + \epsilon, \tag{5.6}$$

where ϵ denotes discrete white noise obtained from a Gaussian distribution with mean zero and variance η^2 selected so that the noise-to-signal ratio $\eta / \|A\mathbf{f}_{\text{exact}}\| = .005$. $\mathbf{f}_{\text{exact}}$ was obtained by evaluating the function

$$f(x) = a_1 \exp\left(-c_1(x - b_1)^2\right) + a_2 \exp\left(-c_2(x - b_2)^2\right),$$
$$a_1 = .5, \quad a_2 = .3, \quad b_1 = \frac{1}{3}, \quad b_2 = \frac{5}{8}, \quad c_1 = 60, \quad c_2 = 200,$$

at the mesh points x_i. Since f is smooth, the regularization operator B in (5.4) is appropriate for any $\beta \geq 0$.

By inverting the data (5.6) without any filtering, one obtains the singular component expansion

$$\hat{f} \stackrel{\text{def}}{=} A^{-1}\mathbf{g} = B^{-1}\tilde{A}^{-1}\mathbf{g}$$
$$= \sum_{j=1}^{n} \left(\tilde{v}_j^T \mathbf{f}_{\text{exact}} + \frac{\tilde{u}_j^T \epsilon}{\tilde{\sigma}_j} \right) B^{-1}\tilde{v}_j$$
$$\stackrel{\text{def}}{=} \sum_{j=1}^{n} \tilde{f}_j B^{-1}\tilde{v}_j.$$

The terms in this expansion are presented in Fig. 3. The magnitude of the coefficients $\tilde{f}_j$ appear as asterisks in Subplots B and C. As seen from Subplot B, these coincide with the magnitude of the coefficients $\tilde{u}_j^T \mathbf{f}_{\text{exact}}$ of the exact solution (circles) for small indices j that correspond to the larger singular values. On the other hand, Subplot C shows that for the smaller singular values, these coincide with the expected error magnitude

$$\sqrt{E|\tilde{u}_j^T/\tilde{\sigma}_j|^2} = \frac{\eta}{\tilde{\sigma}_j}.$$

As pointed out in §2, regularization has the effect of filtering out components associated with small singular values. This takes care of the undesirable large terms $\tilde{u}_j^T/\tilde{\sigma}_j$. Unfortunately, some desirable components $\tilde{v}_j^T \mathbf{f}_{\text{exact}}$ are lost in the process, but this effect is not too severe, since these components also decay as the singular values decrease. Plots such as these can be used to determine information content and to select a reasonable block size p.

Using only the first 20 singular components, two spectral filters were applied-Tikhonov (cf., (2.15)) and TSVD (cf., (2.11)). Figures 4 and 5 illustrate the behavior of various error indicators for these two methods. Subplot A in each of the respective figures shows behavior of the regularized solution error (2.21) as the regularization parameter varies. Subplot B shows the L-curve, while Subplot D shows an approximation to its curvature (2.25), as computed using the finite difference approximations

$$X'(\alpha) \approx \frac{X(\alpha + \tau) - X(\alpha - \tau)}{2\tau}, \qquad X''(\alpha) \approx \frac{X(\alpha + \tau) - X(\alpha) + X(\alpha - \tau)}{\tau^2},$$

and similarly for Y. With Tikhonov we selected the finite difference increment $\tau = \alpha\epsilon_M^{1/3}$, while for TSVD the number of retained singular components k plays the role of the regularization parameter, and $\tau = 1$. Figure 6 shows the best approximate solutions obtained with

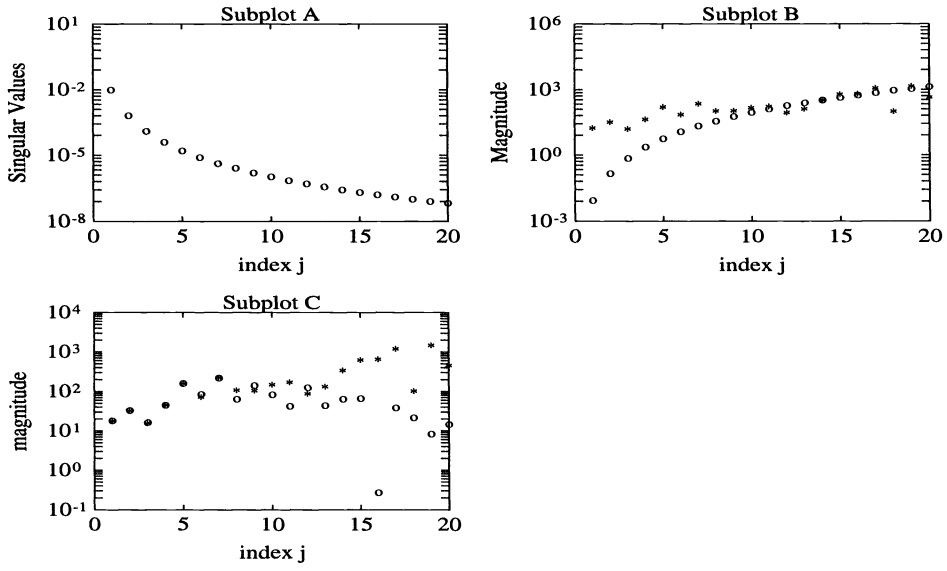


FIG. 3. *Singular components of data. Subplot* A: *first* 20 *singular values of* AB^{-1} *with regularization index* $\beta = 2$; *Subplot* B: *asterisks indicate magnitudes of the coefficients of the pseudoinverse solution,* $\tilde{g}_j/\tilde{\sigma}_j = \langle \tilde{u}_j, g\rangle_2/\tilde{\sigma}_j$ *and circles indicate magnitude of the coefficients of the expected error,* $\langle \tilde{u}_j, \epsilon\rangle_2/\tilde{\sigma}_j$; *Subplot* C: *the circles indicate the coefficients of the exact solution* $\langle \tilde{v}_j, f_{\text{exact}}\rangle_1$.

these regularization methods. Figs. 4–6 illustrate the observation made in [16], that different regularization methods yield error indicators and approximate solutions that are qualitatively quite similar. Although a particular method may fail for a particular problem (e.g., the L-curve estimate in Subplot D of Fig. 4 underestimates α by four orders of magnitude), information from Figs. 3–5 can be combined to make such failures less likely.

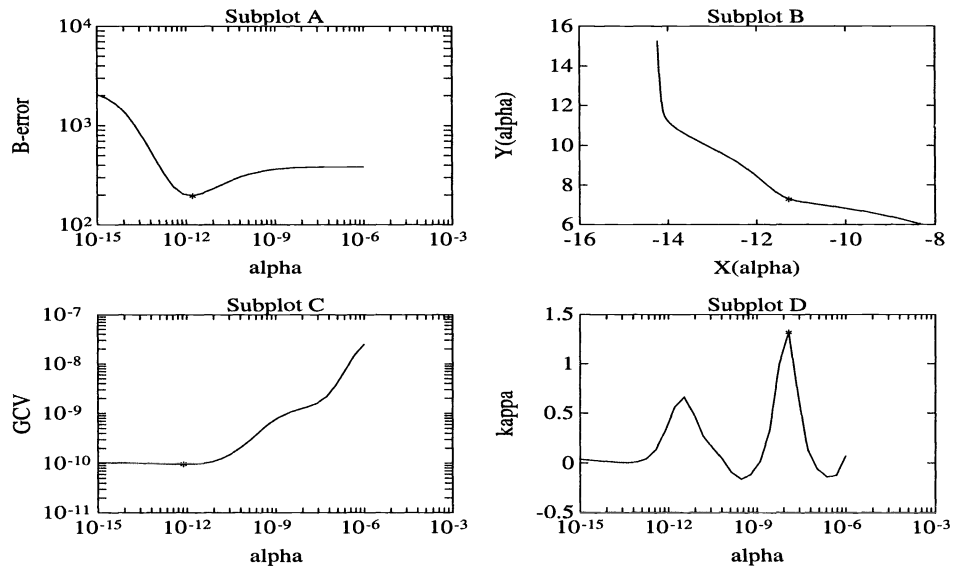


FIG. 4. *Error indicators for Tikhonov Regularization.* α *is the regularization parameter. Subplot* A: *regularized solution error* $\|B(f_\alpha - f_{\text{exact}})\|$; *Subplot* B: *the L-curve*; *Subplot* C: *the GCV function*; *Subplot* D: *curvature* $\kappa(\alpha)$ *of the L-curve. The asterisks in each subplot indicate an estimator of the optimal value of* α.

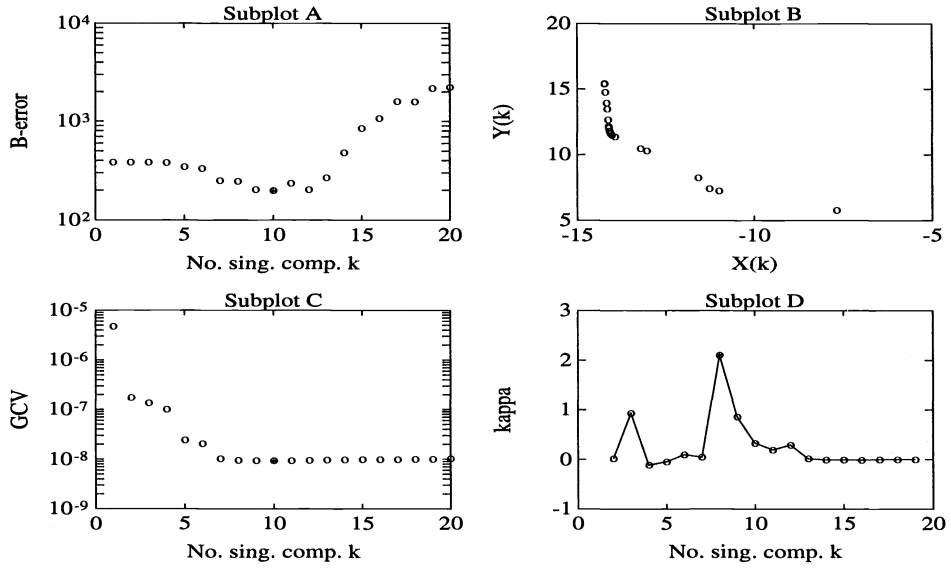


FIG. 5. *Error indicators for TVD. The regularization parameter k is the number of singular components. Subplot* A: *regularized solution error* $\|B(f_k - f_{\text{exact}})\|$; *Subplot* B: *the L-curve*; *Subplot* C: *the GCV function*; *Subplot* D: *curvature* $\kappa(k)$ *of the L-curve. The asterisks in each subplot indicate an estimator of the optimal value of k.*

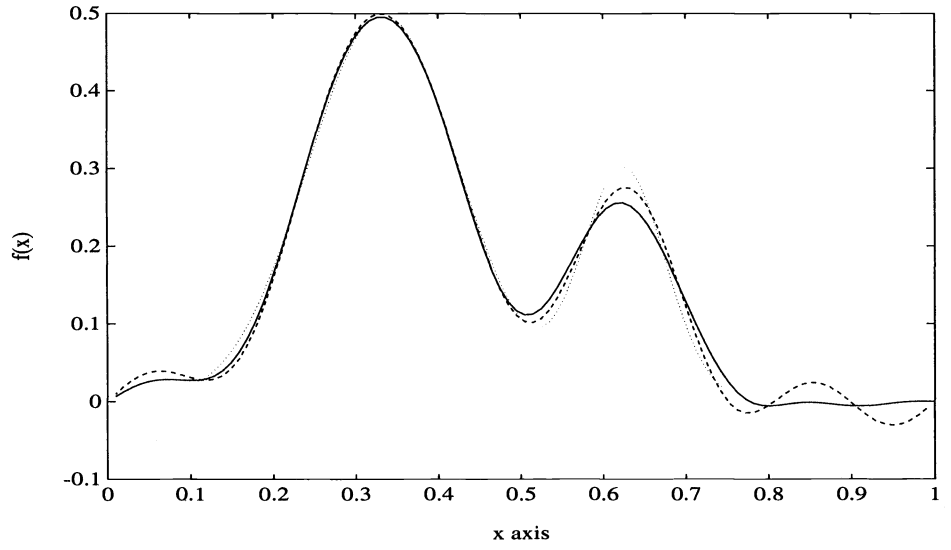


FIG. 6. *Exact and approximate solutions. The dotted line is exact solution; the solid line is best Tikhonov regularized solution; the dashed line is best TSVD solution.*

Finally, the computations used to generate Figs. 4–6 were repeated using the exact singular components, SI approximations, and Lanczos approximations. The results obtained using each of these three were indistinguishable.

5.1. Conclusions. From our analysis and these particular experiments, we make the following conclusions.

1. The analyses for SI (Theorem 3.1 and (4.8)) and for Lanczos (Theorem 4.1 and (4.10)) explain our numerical results. For the jth singular value, the SI relative error is $\mathcal{O}(r_j^k)$, where r_j depends on the ratio σ_{p+1}/σ_j. This implies that the larger singular values converge much more quickly than the smaller ones. For Lanczos, the corresponding error is $\mathcal{O}(r^{k-j})$, where r is essentially independent of j, but does depend on the rate of decay of the singular values.

2. When using a regularization operator, transforming the operator A (cf., (2.16)–(2.17)) can significantly increase the convergence rates for both SI and Lanczos. This effect seems to be more pronounced for SI.

3. In terms of operator evaluations, Lanczos with reorthogonalization is substantially less expensive than SI. Precisely how much less expensive depends on desired accuracy. Since smaller singular components are filtered out by regularization methods, uniformly high accuracy is not necessary. For our particular numerical study, we judge Lanczos to be about half as expensive as SI.

4. When the singular values decay rapidly, Lanczos should be used with reorthogonalization. Even if spurious "ghost" singular values could be distinguished from the true ones, advantage gained by applying Lanczos without reorthogonalization is offset by the fact that significantly more operator evaluations are required.

In the numerical example presented here, the overall performance of the Lanczos algorithm is superior to that of SI. This same general conclusion was reached in [21] for applications in structural engineering. In that paper, Lanczos was observed to be an order of magnitude less expensive than SI in terms of operator evaluations. In our applications to ill-posed problems, we have found Lanczos to be roughly half as expensive as SI. The difference may be due to our less stringent accuracy requirements and to the rapid decay rate of the singular values. There are other applications where this comparison may be even better for SI. For example, Gauss–Newton and Levenberg–Marquardt methods for nonlinear least-squares inverse problems give rise to *sequences* of linear subproblems of the form (2.14). In this context $\mathcal{A}$ is the Fréchet derivative of a nonlinear compact operator. It is demonstrated in [25] that for problems with many degrees of freedom (distributed parameter estimation problems), very low rank approximations (i.e., partial SVDs) of $\mathcal{A}$ can be used quite efficiently in such algorithms. In this situation, each of the linear subproblems is a perturbation of the preceding one, so that the partial SVD of the preceding $\mathcal{A}$ provides an excellent initial guess for the current one. The block structure of SI is better suited to take advantage of this than is the basic Lanczos algorithm, where each new partial SVD is obtained from a single initial vector.

The block structure of the SI algorithm also lends itself very naturally to parallel implementation. This is a distinct advantage over basic Lanczos iteration, which is inherently sequential. There are block Lanczos algorithms that combine the advantages of block structure with the rapid convergence rates of Lanczos (see [4], [9], [24]). Applying arguments in the proof of Theorem 4.1 to Theorem 5 of [24], with block size b one obtains

$$\frac{\sigma_j - \sigma_j^{(k)}}{\sigma_j} = \mathcal{O}(\hat{r}_j^{\,k-j}), \qquad 1 \le j \le p,\ k > j, \tag{5.7}$$

where

$$\hat{r}_j = \left(\overline{\gamma}_j + \sqrt{\overline{\gamma}_j^2 - 1}\right)^{-2}, \qquad \overline{\gamma}_j = 2\left(\frac{\sigma_j^2}{\sigma_{j+p}^2}\right) - 1. \tag{5.8}$$

With rapidly decaying singular values and a large block size b, $\sigma_j \gg \sigma_{j+b}$, which implies $\hat{r}_j \approx (\sigma_{j+b}/\sigma_j)^4/16$. One would expect the block Lanczos algorithm to converge much more

quickly than basic Lanczos. However, each block Lanczos iteration requires the same number of operator evaluations as b basic Lanczos iterations, which makes a precise cost analysis difficult. When the block sizes are the same, block Lanczos will converge more rapidly than SI, and both will require the same number of operator evaluations at each iteration.

REFERENCES

[1] W. F. AMES, *Numerical Methods for Partial Differential Equations*, 2nd ed., Academic Press, New York, 1977.

[2] C. T. H. BAKER, L. FOX, D. F. MEYER, AND K. WRIGHT, *Numerical solution of Fredholm integral equations of the first kind*, Comput. J., 7 (1964), pp. 141–148.

[3] H. T. BANKS AND K. KUNISCH, *Estimation Techniques for Distributed Parameter Systems*, Birkhäuser, Basel, 1989.

[4] J. CULLUM AND R. A. WILLOUGHBY, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1 Theory*, Birkhäuser, Basel, 1985.

[5] J. E. DENNIS AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[6] L. ELDEN, *Algorithms for the regularization of ill-conditioned least squares problems*, BIT, 17 (1977), pp. 134–145.

[7] H. W. ENGL AND C. W. GROETSCH, EDS., *Inverse and Ill-Posed Problems*, Academic Press, New York. 1987.

[8] D. S. GILLIAM, J. R. LUND, AND C. R. VOGEL, *Quantifying information content for ill-posed problems*, Inverse Problems, 6 (1990), pp. 725–736.

[9] G. H. GOLUB, F. T. LUK, AND M. OVERTON, *A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix*, ACM Trans. Math. Software, 7 (1981), pp. 149–169.

[10] G. H. GOLUB AND C. VAN LOAN, *Matrix Computations*, 2nd ed., Johns Hopkins Press, Baltimore, 1989.

[11] C. W. GROETSCH, *The Theory of Tikhonov Regularization for Fredholm Equations of the First Kind*, Pitman, Boston, 1984.

[12] C. W. GROETSCH AND C. R. VOGEL, *Asymptotic theory of filtering for linear operator equations with discrete noisy data*, Math. Comp., 49 (1987), pp. 499–506.

[13] M. HANKE AND P. C. HANSEN, *Regularization methods for large scale problems*, Report UNIC-92-04, Technical University of Denmark, DK-2800 Lyngby, Denmark, August 1992; Surveys on Mathematics for Industry, to appear.

[14] M. HANKE, J. NAGY, AND R. PLEMMONS, *Preconditioned iterative regularization for ill-posed problems*, in Numerical Linear Algebra and Scientific Computing, L. Reichel, A. Ruttan, and R. Varga, eds., de Guyter Press, Berlin, pp. 141–163.

[15] P. C. HANSEN, *Analysis of discrete ill-posed problems by means of the L-curve*, SIAM Rev., 34 (1992), pp. 561–580.

[16] ———, *Numerical tools for analysis and solution of Fredholm integral equations of the first kind*, Inverse Problems, 8 (1992), pp. 849–872.

[17] P. C. HANSEN AND D. P. O'LEARY, *The Use of the L-curve in the Regularization of Discrete Ill-posed Problems*, Report UMAICS-TR-91-142, Dept. of Computer Science, Univ. of Maryland, College Park, Oct. 1991; SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[18] R. KRESS, *Linear Integral Equations*, Springer-Verlag, New York, Berlin, 1989.

[19] L. LANDWEBER, *An iteration formula for Fredholm integral equations of the first kind*, Amer. J. Math., 73 (1951), pp. 615–624.

[20] MATLAB, The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, South Natick, MA 01760, 1992.

[21] B. NOUR-OMID, B. PARLETT, AND R. L. TAYLOR, *Lanczos vs. subspace iteration for solution of eigenvalue problems*, Internat. J. Numer. Meth. Engrg., 19 (1983), pp. 859–871.

[22] D. P. O'LEARY AND J. A. SIMMONS, *A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 474–489.

[23] B. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, 1980.

[24] Y. SAAD, *On the rates of convergence of the Lanczos and block-Lanczos methods*, SIAM J. Numer. Anal., 17 (1980), pp. 687–706.

[25] C. R. VOGEL AND J. G. WADE, *A modified Levenberg-Marquardt algorithm for large-scale inverse problems*, in Computation and Control III, Proc. 3rd Bozeman Conference, K. L. Bowers and J. R. Lund, eds., Birkhäuser, Basel, 1993, pp. 367–378.

[26] G. WAHBA, *Practical approximate solutions to linear operator equations when the data are noisy*, SIAM J. Numer. Anal., 14 (1977), pp. 651–667.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 755–775, July 1994

FAST WAVELET BASED ALGORITHMS FOR LINEAR EVOLUTION EQUATIONS*

BJORN ENGQUIST†, STANLEY OSHER†, AND SIFEN ZHONG‡

Abstract. The authors devise a class of fast wavelet based algorithms for linear evolution equations whose coefficients are time independent. The method draws on the work of Beylkin, Coifman, and Rokhlin [*Comm. Pure Appl. Math.*, 44 (1991), pp. 141–184], which they applied to general Calderon–Zygmund type integral operators. The authors apply a modification of their idea to linear hyperbolic and parabolic equations, with spatially varying coefficients. The complexity for hyperbolic equations in one dimension is reduced from $O(N^2)$ to $O(N \log^3 N)$. There are somewhat better gains for parabolic equations in multidimensions.

Key words. wavelets, hyperbolic, parabolic, numerical methods

AMS subject classifications. primary 65M06; secondary 65M12

1. Introduction.

During the last few years a number of fast computational algorithms have been developed for elliptic problems. These are techniques for which the number of arithmetic operations needed are close to linear as a function of the number of unknowns. Examples of algorithms of such complexity are multigrid methods and the so-called fast Poisson solvers. The fast multipole method and wavelet based methods for elliptic problems formulated as integral equations also belong to this category [8], [1].

There has not been the same progress for hyperbolic and parabolic methods. In general, classical numerical techniques for these problems are already optimal.

Consider a system of evolution equations

$$\begin{aligned} \partial_t u + L(x, \partial_x)u &= f(x), \quad x \in \Omega \subset \mathbf{R}^d, \quad t > 0, \\ u(x, 0) &= u_0(x), \end{aligned} \tag{1.1}$$

with boundary conditions, where L is a differential operator.

An explicit discretization of this problem typically takes the form

$$\begin{aligned} u_j^n &\approx u(x_j, t_n), \qquad t_n = n\Delta t, \\ & \qquad\qquad x_j = (j_1 \Delta x_1, \ldots, j_d \Delta x_d), \\ u^{n+1} &= Au^n + F, \\ u^0 &= u_0, \\ u, F &\in \mathbf{R}^{N^d}, \qquad \Delta t = \text{const}\,|\Delta x|^r. \end{aligned} \tag{1.2}$$

The vector u^n contains all the unknowns u_j^n at time level t_n. For simplicity we shall assume $j_\nu = 1, 2, \ldots, N$ in all dimensions $\nu = 1, \ldots, d$.

The matrix A is $(N^d \times N^d)$ with the number of elements $\neq 0$ in each row and each column bounded by a constant. Every timestep requires $\mathcal{O}(N^d)$ arithmetic operations, and the overall complexity for a time interval of $\mathcal{O}(1)$ is of the same order as the number of unknowns, $\mathcal{O}(N^{d+r})$.

There are, however, some fast methods based on the analytic form of the solution operator. In [3] the multidimensional heat operator was treated with u_0 and f both zero, but with

*Received by the editors February 19, 1992; accepted for publication (in revised form) June 9, 1993.

†Department of Mathematics, University of California, Los Angeles, California 90024 (engquist@math.ucla.edu, sjo@math.ucla.edu).

‡Department of Mathematics, University of California, Los Angeles, California 90024 (szhong@math.ucla.edu).

Research supported by Office of Naval Research grant N00014-91-J-1034.

inhomogeneous boundary data given at M points for N time levels. There the closed form of the solution evaluated at M points at each time level N was obtained in $O(NM)$ rather than $O(N^2M^2)$ operations. Also, in [4], the same authors obtained an algorithm for evaluating the sum of N Gaussians at M arbitrarily distributed points in $O(N+M)$ operations. So far, their interesting method appears to need an explicit analytic representation of the heat kernel, effectively ruling out variable coefficient problems.

The formula (1.2) has a simple closed form solution

$$u^n = A^n u_0 + \sum_{\nu=0}^{n-1} A^\nu F. \tag{1.3}$$

This form can be used to compute the solution $A^n u_0$ for $F = 0$ in $\log n$ steps ($n = 2^m$, m integer; here and throughout, $\log n = \log_2 n$) by repeated squaring of $A : A, A^2, A^4, A^8, \ldots, A^{2^m}$.

Unfortunately the later squarings involve almost dense matrices and the overall complexity is $O(N^{3d} \log N)$, which is larger than that using (1.2) directly.

For an appropriate representation of A in a wavelet basis, all of the powers A^ν may be approximated by sparse matrices and the algorithm using repeated squaring should then be advantageous.

We shall consider the following algorithms for the computation of the closed form solution (1.3) of the inhomogeneous problem in $m = \log n$ steps

$$\left.\begin{aligned} &B := SAS^{-1}, \\ &C := I, \\ &C := TRUNC(C + BC, \varepsilon) \\ &B := TRUNC(BB, \varepsilon) \end{aligned}\right\} \quad \text{(iterate } m \text{ steps)}, \tag{1.4}$$
$$u^n := S^{-1}(BSu^0 + CSF).$$

The matrix S corresponds to a fast transform of wavelet type and the truncation operator sets elements in a matrix to zero if their absolute value is below a given threshold.

$$\tilde{A} = TRUNC(A, \varepsilon) : \begin{cases} \tilde{a}_{ij} = a_{ij} & |a_{ij}| \geq \varepsilon, \\ \tilde{a}_{ij} = 0 & |a_{ij}| < \varepsilon. \end{cases} \tag{1.5}$$

It is easy to see that algorithm (1.4) is equivalent to (1.3) for $\varepsilon = 0$. This is not so for $\varepsilon > 0$. We shall, however, show that it is possible to choose ε small enough for the result of (1.4) to be arbitrarily close to (1.3) but still with very few arithmetic operations.

For a fixed predetermined accuracy level, the computational complexity to calculate a one-dimensional hyperbolic equation can be reduced from the standard $O(N^2)$ to $O(N(\log N)^3)$. The extra cost per timestep is minimal. This also makes it possible, as a curiosity, to use algorithms that are unstable in the traditional sense.

Our technique is more favorable for parabolic problems. A d-dimensional explicit calculation with standard complexity $O(N^{d+2})$ may be reduced to $O(N^d(\log N)^3)$.

The algorithm (1.4) can be extended to some problems with time dependent data. In this case, we clearly need to compress the information in the data such that not all the $O(N^{d+r})$ values in, e.g., the inhomogeneous term $f(x_j, t_n)$ are needed.

One simple, but important, application of this type is from optics or electro-magnetic scattering with a time periodic source. If k points are needed to resolve one time period, we can group k timesteps together

$$u^{n+k} = A^k u^n + \sum_{j=0}^{k-1} A^j F_{n+k-j-1}, \tag{1.6a}$$

where

$$F_n = \Delta t f(t_n). \tag{1.6b}$$

This equation is now of the type (1.2) with timestep $k\Delta t$ and with inhomogeneous term

$$F = \sum_{j=0}^{k-1} A^j F_{n+k-j-1}. \tag{1.6c}$$

In §§2 and 3 we shall discuss the analytical properties of the algorithm. Numerical examples are presented in §4.

2. Hyperbolic problems. Consider first the simple one-dimensional scalar advection equation

$$\begin{aligned} &\partial_t u + a\partial_x u = 0, \qquad a > 0, \\ &u(x,0) = u_0(x), \qquad 0 \le x \le 1. \end{aligned} \tag{2.1}$$

The functions u_0 and thus u are assumed to be 1-periodic in x. The solution of (2.1) is given by:

$$u(x,t) = u_0(x - at). \tag{2.2}$$

The different rows of A^ν in a numerical solution of (2.1) will represent approximations of the Green function, G, below.

$$\begin{aligned} u(x,t) &= \int_{-\infty}^{\infty} G(x,y,t)u_0(y)dy, \\ u(x,t) &= \int_{-\infty}^{\infty} \delta(x - y - at)u_0(y)dy. \end{aligned} \tag{2.3}$$

Let φ_J be a truncated wavelet expansion of a δ-function with an orthonormal set of compactly supported wavelets

$$\delta(x) \sim \varphi_J(x) = \sum d_k^j 2\left(\frac{\nu - j}{2}\right)\psi(2^{\nu-j}x - k + 1) + s_1^\nu \varphi(x).$$

The choices of $\psi(x)$ and the resulting $\varphi(x)$ will be discussed below. Assume that the rows of A^ν are discrete δ-functions, i.e., just one element is nonzero and large. For each level $j = 1, 2, \ldots, J$ there are only a finite number of $d_k^j \neq 0$ since the wavelets are compactly supported. With $J = m = \log N$ there is only $\log N$ of all $d_k^j \neq 0$. Thus each row in B, (1.4), has $\log N$ elements, $b_{jk} \neq 0$. The matrix B^2 is also a transform of an idealized matrix A^ν and will have $N \log N$ elements different from zero. This means that each iteration step in the algorithm (1.4) produces $O(N(\log N)^2)$ flops when $F = 0$. We have assumed that calculations are only carried out for those B^2 elements that are different from zero. In practice, a slightly larger number of elements needs to be computed and then truncated. This corresponds to the case when the location of the δ-functions is only approximately known. Compare the wavelet technique for the Burgers equation by Maday, Perrier, and Ravel [6].

Each row of C, (1.4), is a transform of a step function

$$\tilde{c}(x) = \begin{cases} \text{const} & 0 \le x \le at, \\ 0, & \text{else.} \end{cases}$$

Since $\tilde{c}(x) \equiv$ constant only at $x = at$, this function can also be represented by $\log N$ wavelets and thus *the overall cost is* $O(N(log\ N)^3)$.

In numerical computations the rows of A^ν are only approximations of δ-functions. If an upwind scheme

$$\begin{aligned} u_j^{n+1} &= u_j^n - \lambda(u_j^n - u_{j-1}^n), \\ u_j^0 &= u_0(x_j), \qquad j = 1, 2, \ldots, N, \\ \lambda &= a\Delta t/\Delta x < 1, \end{aligned} \tag{2.4}$$

is used, A will have the form

$$A = \begin{bmatrix} 1-\lambda & 0 & \cdots & & \lambda \\ \lambda & 1-\lambda & 0 & \cdots & 0 \\ 0 & \lambda & 1-\lambda & 0 \quad \cdots & 0 \\ & & & & \\ 0 & \cdots & 0 & \lambda & 1-\lambda \end{bmatrix}.$$

The matrix A^ν will have Toeplitz structure. Each row is still an approximation of a δ-function. The first order smoothing effect of (2.4) is given by the modified equation (see [5])

$$\partial_t u + a\partial_x u = (a\Delta x/2)\partial_x^2 u. \tag{2.5}$$

Equation (2.5) is parabolic with a fundamental solution of the form

$$G(x - y, t) = (2\pi a\Delta x t)^{-\frac{1}{2}} \exp(-(x - y - at)^2/(2a\Delta x t)). \tag{2.6}$$

Compare the solution formula for parabolic problems (3.2).

Each row of A^ν is thus a close approximation to the function $G(x - y, t)$ above. The computational complexity of the algorithm (1.4) depends on how many wavelets are needed to represent $G(x - y, t)$ as a function of x, $(0 \le t \le T)$ with a given accuracy.

Higher order accurate (say, order $2p - 1$) dissipative finite difference approximations to (2.1) are usually modelled by the equation

$$u_t + au_x = (-1)^{p+1} k_p (\Delta x)^{2p-1} \left(\frac{\partial}{\partial x}\right)^{2p} u \tag{2.7}$$

with $k_p \ge \delta > 0$, δ independent of Δx.

The fundamental solution for this parabolic equation is:

$$G_p(x, t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\xi \exp(i\xi(x - at) - k_p(\Delta x)^{2p-1}\xi^{2p}t).$$

The key, simple estimate we shall obtain here (and which we certainly do not claim is new) is:

$$\left| x^{m+1} \left(\frac{\partial}{\partial x}\right)^m G_p(x + at, t) \right| \le C_{m,p} \tag{2.8}$$

uniformly in $0 < t$ and Δx and for all nonnegative integers m.

Proof of 2.8. We wish to bound

$$\frac{1}{2\pi}\int_{-\infty}^{\infty}(i\xi)^m x^{m+1}e^{i\xi x-k_p(\Delta x)^{2p-1}t\xi^{2p}}d\xi$$
$$=\frac{i}{2\pi}\int_{-\infty}^{\infty}e^{i\xi x}\left(\frac{\partial}{\partial\xi}\right)^{m+1}\left[\xi^m e^{-k_p(\Delta x)^{2p-1}\xi^{2p}t}\right]d\xi$$
$$=\frac{i}{2\pi}\int_{-\infty}^{\infty}\left[\exp\left(\frac{i\xi x}{(t(\Delta x)^{2p-1}k_p)^{1/2p}}\right)\right]\left[\left(\frac{\partial}{\partial\xi}\right)^{m+1}\left[\xi^m e^{-\xi^{2p}}\right]\right]d\xi.$$

The result is now clear. Also, an inspection of the right-hand side of the above equation shows that $C_{m,p}$ can be chosen to be arbitrarily small if $t(\Delta x)^{2p-1}$ is large enough. Discrete estimates analogous to (2.8) uniform in powers of A are needed so that the compression method described below is valid.

Remark R1. Let the general space dependent coefficient, one-dimensional system of hyperbolic equations

$$u_t + A(x)u_x = C(x)u$$

be approximated by a dissipative finite difference scheme of order $2p-1$, where u is an ℓ vector, A is a uniformly diagonalizable smooth $\ell\times\ell$ matrix, with all real eigenvalues $\lambda_i(x)$, and $C(x)$ is smooth. Typically, its model equation is a systems version of (2.1)

$$u_t + A(x)u_x = C(x)u + (-1)^{p+1}(\Delta x)^{2p-1}P\left(x,\frac{\partial}{\partial x}\right)u,$$

where $(-1)^{p+1}P(x,\frac{\partial}{\partial x})$ is a $2p$ order elliptic operator. A more involved argument shows that the fundamental solution satisfies an estimate of the type (2.8) with the expression $x+at$ replaced appropriately by solutions of $\frac{d\tilde{x}}{dt}=\lambda_i(\tilde{x})$, $\tilde{x}(0)=x$, $i=1,\dots,\ell$, and with $C_{m,p}$ possibly growing in time like $C_{m,p}e^{kt}$ for k fixed.

Our numerical procedure involves the compression of the matrix A^ν, which, for the purpose of analysis only, we shall view as the discretization of the fundamental solution for either (2.5) or (2.7),

$$(A^n)_{jk}=G(x_j,y_k,t^n),$$

where the interval $[0,1]$ is discretized via

$$x_j=\frac{j}{N},\quad j=1,\dots,N,\quad N=2^\nu,$$

$[0,1]\times[0,1]$ is discretized via (x_j,y_k), and $t^n=n\Delta t=n\lambda\Delta x$, $n=0,1,\dots$.

We now adapt the terminology, notation, and results of [1] to this unsteady problem (1.1).

Finite difference schemes approximating (1.1), e.g., (2.4), are regarded as acting on a vector $\{s_k^0\}_{k=1}^N$, which is to be viewed as approximating $u(x,0)$ on the finest scale:

$$s_k^0=2^2\int\varphi(2^\nu x-k+1)u(x,0)dx.$$

All functions, both continuous and discrete, are extended periodically:

$$u(x,t)\equiv u(x+1,t),$$
$$s_{k+N}^0\equiv s_k^0,$$

etc.

The function φ satisfies

$$\varphi(x) = \sum_{p=0}^{2m-1} h_{p+1}\varphi(2x - p).$$

The function $\psi(x)$, which will generate an orthonormal basis, is obtained via

$$\psi(x) = \sum_{p=0}^{2m-1} g_{p+1}\varphi(2x - p)$$

with $g_p = (-1)^{p-1}h_{2m-p+1}$, $p = 1, \ldots, 2m$, and $\int \varphi(x)dx = 1$.

The coefficients $\{h_p\}_{p=1}^{2m}$ are generally chosen so that

$$\psi_{j,k}(x) = 2^{-\frac{j}{2}}\psi(2^{-j}x - k + 1)$$

for j, k integers, form an orthonormal basis, and, in addition, the function $\psi(x)$ has m vanishing moments

$$\int \psi(x)x^{\ell}dx = 0, \qquad \ell = 0, 1, \ldots, m - 1.$$

Also we define

$$\varphi_{jk} = 2^{-\frac{j}{2}}\varphi(2^{-j}x - k + 1).$$

Finally, we assume as in [1] that there exists a real constant $\tau_m (\tau_1 = \frac{1}{2})$ such that the following conditions are satisfied:

$$\int \varphi(x + \tau_m)x^{\ell}dx = 0 \quad \text{for } \ell = 1, \ldots, m - 1,$$

and $\int \varphi(x)dx = 1$.

In this case the quadrature formula becomes:

$$s_k^0 = \frac{1}{\sqrt{N}}\left(f\left(\frac{k - 1 + \tau_m}{N}\right) + O(N^{-(m+1)})\right),$$

and the initial discretization error is $O(N^{-(m+1)})$ up to uniform translation.

The decomposition of the vector $\{s_1^0, \ldots, s_{2^v}^0\}$ into the basis we use to compute with comes via

$$\begin{array}{ccccccc} \{s_k^0\} & \longrightarrow & \{s_k^1\} & \longrightarrow & \{s_k^2\} & \cdots \longrightarrow & \{s_k^v\} \\ & \searrow & \{d_k^1\} & \searrow & \{d_k^2\} & \cdots \searrow & \{d_k^v\}. \end{array}$$

This is implemented in $O(N)$ operations using:

$$s_k^j = \sum_{p=1}^{p=2m} h_p s_{p+2k-1}^{j-1},$$

$$d_k^j = \sum_{p=1}^{p=2m} g_p s_{p+2k-1}^{j-1},$$

and the s_k^j, d_k^j are viewed as periodic sequences with period $2^{\nu-j}$.

The coordinates in the orthonormal basis consist of

$$[d_1^1, \ldots, d_{\frac{N}{2}}^1, d_1^2, \ldots, d_{\frac{N}{4}}^2, \ldots, d_1^n, s_1^n].$$

The inverse mapping can also be done in $O(N)$ operations.

Each of the s_k^j is thought of as approximating

$$\begin{aligned} s_k^j &= 2\left(\frac{\nu-1}{2}\right) \int f(x)\varphi(2^{\nu-j}x - k + 1)dx \\ &= 2^{-(\frac{\nu-j}{2})}\Big[f(2^{-\nu+j}(k-1+\tau_m)) \\ &\qquad + O(N^{(-\nu+j)(m+1)})\Big], \end{aligned}$$

while each d_k^j is thought of as approximating

$$d_k^j = 2\left(\frac{\nu-1}{2}\right) \int f(x)\psi_{jk}(x)dx.$$

The numerical procedure effectively transforms the approximate discretization of the matrix $G(x_j, y_k, t^n)$ which is $(A^n)_{jk}$. Estimate (2.8) (corresponding to (4.5) and (4.6) of [1]), uniform in all parameters, indicates (via an argument of [1]) that truncating A^n by removing elements of a band of width $b \geq 2m$ around a shifted diagonal (and its periodic extension), i.e., those for which

$$|j - k - a\lambda n| \geq b > 2m,$$

which replaces A^n by $A^{n,b}$, leads to an estimate

$$\|A^n - A^{n,b}\| \leq \frac{C}{b^m} \log(N)$$

for C depending only on G.

Figure 1 shows what the nonzero elements in $A^{n,b}$ look like in the transformed basis for a variable coefficient case when $a = a(x)$ in (2.1).

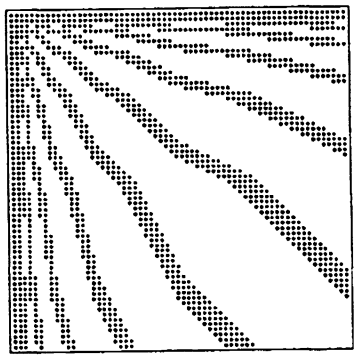


FIG. 1. *Hyperbolic equation: the significant elements in* $(SAS^{-1})^n$ *for* $n = 1024$.

Our examples are, of course, academic. The gain will come for highly oscillatory problems for which a large number of grid points are needed.

It also follows easily that for large N and fixed precision ε, only $O(N \log N)$ elements will be greater than ε. Alternatively, by discarding all elements that are smaller than a fixed threshhold we compress it to $O(N \log N)$ elements. Again, following the discussion in [1], we note that this naive approach is to construct the full matrix in the wavelet basis and then to threshold. Clearly this is an $O(N^2)$ operation.

Since we have, a priori, the structure of the singularities of the matrix A^ν, the relevant coefficients can be evaluated by using the quadrature formulas. Estimate (2.8) guarantees that this procedure requires $O(N \log N)$ operations.

Remark R2. It is interesting to note that certain so-called unstable difference schemes can be used without any drastic loss of efficiency. If (2.1) is approximated by

$$\begin{aligned} u_j^{n+1} &= u_j^n - \lambda(u_{j+1}^n - u_{j-1}^n)/2, \\ u_j^0 &= u_0(x_j), \quad j = 1, 2, \ldots, N, \end{aligned} \tag{2.9}$$

the algorithm is not stable for any fixed $\lambda > 0$, see, e.g., [7].

The approximation does converge if $\Delta t \leq C\Delta x^2$, ($\lambda \leq C\Delta x$) with an amplification factor $1 + \mathcal{O}(\Delta t)$. The number of timesteps for $t = \mathcal{O}(1)$ calculation will be large, $n = \mathcal{O}(\Delta x^{-2}) = \mathcal{O}(N^2)$. This is devastating for the standard explicit algorithm (1.2) but will only affect the complexity of (1.4) by a constant factor. The number of iterations (m in (1.4)) will increase from $\log(N)$ to $\log(N^2)$.

Our approach is, in general, not as favorable for multidimensional hyperbolic systems

$$\begin{aligned} \partial_t u + \sum_{j=1}^{d} A_j(x)\partial_{x_j} u &= f(x), \qquad x \in \mathbf{R}^d, \\ u(x, 0) &= u_0(x). \end{aligned} \tag{2.10}$$

When u is a scalar, or if the system can be diagonalized, the algorithm (1.4) works well. The solution is given by integration along characteristics and the support of the Green function is a small number of points (see Remark R1 above). In the idealized case, each row of A^ν consists of a fixed number of δ-functions. Its wavelet representation will have $\log(N^d)$ nonzero terms. The overall complexity for (1.4) is then $\mathcal{O}((\log N)^3 N^d)$ when the knowledge of the location of the δ-functions is used. This is better than the standard $\mathcal{O}(N^{d+1})$ estimate.

In general, however, the Green function for (2.6) has a support with positive volume in $\mathbf{R}^d$ and with a singular support of positive measure in Hausdorff dimension $d - 1$. The representation of the singular support consists of $\mathcal{O}(N^{d-1})\delta$-functions in each row of A^ν. This corresponds to $\mathcal{O}(\log(N)N^{d-1})$ wavelets and the overall algorithm contains at least $(\mathcal{O}(\log N)^2 N^{2d-1})$ wavelets.

For general multidimensional problems and for very large time, the new algorithm is still of interest in special cases, e.g., if it is needed for a large number of different data u_0, f.

3. Parabolic problems. The Green function for parabolic problems is smooth in contrast to the hyperbolic case. The pure initial value problem for the heat equation

$$\begin{aligned} \partial_t u &= \Delta u, \quad t > 0, \quad x \in R^d, \\ u(x, 0) &= u_0(x), \end{aligned} \tag{3.1}$$

has a solution of the form

$$u(x, t) = (4\pi t)^{-d/2} \int_{R^d} \exp(-|x - y|^2/4t) u_0(y) dy. \tag{3.2}$$

In bounded domains the kernel has to be changed slightly depending on the boundary conditions. For positive $t(= n\Delta t)$ each row in A^n is always an approximation of segments of regular functions.

Our new technique is, in general, more favorable for parabolic problems than hyperbolic ones. The structure of the matrix B in (1.4) is simpler. When t increases, the kernel becomes smoother and α_{jk} can be truncated to zero for all k when j is large enough.

Explicit methods for (3.1) also require more operations than for hyperbolic problems when the standard method is used. This follows from the parabolic stability requirement

$$\Delta t \leq \text{const } |\Delta x|^2. \tag{3.3}$$

The new technique is only marginally affected by the constraint (3.3). Compare here the discussion above for unstable hyperbolic methods.

In more general higher order multidimensional parabolic cases the fundamental solution of, e.g.,

$$u_t + (-\Delta)^d u = 0$$

is

$$G_d(x,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} d\xi \exp(i\xi \cdot x - |\xi|^{2d}t).$$

This is merely a multidimensional and rescaled version of the fundamental solution used in (2.8), and a simpler, but multidimensional, version of (2.8) is just:

$$\left||x|^{m+1} D_x^m G_d(x,t)\right| \leq C_{md}.$$

Moreover, C_{md} is arbitrarily small if t is large enough (this of course requires the nonexistence or other special behavior of lower order terms).

The matrix compression technique is easy here (for periodic problems without boundary conditions) because the significant terms of $[A^\nu]$ lie near the main diagonal and its periodic extension in one dimension. In two space dimensions (as is usual for elliptic operators), we also need to consider diagonals $i = j \pm kN$ for $0 < k \leq d$. Recall A is an $N^2 \times N^2$ matrix in 2 dimensions.

It is clear that a priori thresholding (to obtain $O(\epsilon)$ precision) near the image of these diagonals will give us an $O(N^d(\log N)^3)$ operation for each evaluation of the solution, where d is the number of space dimensions for the problem.

4. Numerical experiments. The algorithm (1.4) was applied to hyperbolic problems in one space dimensions and to one- and two-dimensional parabolic problems. Various difference approximations and wavelet spaces were used. We present results concerning the accuracy of the calculations and the sparsity of $(SAS^{-1})^n$.

4.1. Hyperbolic problems. Consider the following scalar hyperbolic problem:

$$\begin{aligned} \partial_t u + a(x)\partial_x u &= f(x), \\ u(x,0) &= u_0(x), \end{aligned} \tag{4.1a}$$

with periodic boundary conditions ($0 \leq x \leq 1$). We made the following choices:

$$a(x) = 0.5 + 0.115\sin(4\pi x), \tag{4.1b}$$

$$f(x) = \cos(4\pi x), \tag{4.1c}$$

$$u_0(x) = \sin(4\pi x). \tag{4.1d}$$

In the discretization, $\Delta x = 1/1024$ and $\Delta t/\Delta x = 1$. The wavelet transform operator S uses the Daubechies-8 wavelets, which have 8 coefficients and have 4 vanishing moments. Finite difference schemes of order of accuracy 1, 2, 3, 4, and 5 are tested.

These finite difference schemes are obtained as follows. In each interval

$$I_{\nu-\frac{1}{2}} = \{x/(\nu-1)\Delta x \le x \le \nu\Delta x\} \tag{4.2}$$

a polynomial of degree k is constructed. This polynomial interpolates the two points $(x_{\nu-1}, u^n_{\nu-1})$ and (x_ν, u^n_ν) and $k-1$ of its neighbors. If k is even, these interpolation points go from $x_{\nu-k/2}$ to $x_{\nu+k/2}$. If k is odd, they go from $x_{\nu-((k-1)/2)-1}$ to $x_{\nu+((k-1)/2)}$. This gives us a reconstruction function that is a polynomial of degree k in each $I_{\nu-1/2}$ and is continuous, but generally not differentiable, at the boundary points $x_{\nu-1}$ and x_ν. We call this function $R^{n,k}(x)$.

To approximate (4.1) at the grid points (x_ν, t^{n+1}), we solve (4.1) "exactly" with initial data

$$u_{\Delta x}(x, t^n) = R^{n,k}(x) \tag{4.3}$$

for $t^n \le t \le t^{n+1}$, evaluate the solution at (x_ν, t^{n+1}), and set $u^{n+1}_\nu = u_{\Delta x}(x_\nu, t^{n+1})$. We require $\frac{\Delta t}{\Delta x}\|a\|_\infty < 1$, so the solution depends only on data in $I_{\nu-1/2}$ if $a(x) > 0$ and $I_{\nu+1/2}$ if $a(x) < 0$.

In the special case when $a(x) = a$, constant, then

$$\begin{aligned} u^{n+1}_\nu &= R^{n,k}(x_\nu - a\Delta t) \\ &\quad + \int_{t^n}^{t^{n+1}} f(x_\nu - a(t^{n+1} - s))ds. \end{aligned} \tag{4.4}$$

In the case when $f \equiv 0$, we get some familiar schemes: For $k = 1$ this is just the first order accurate upwind difference scheme (2.4). For $k = 2$ this is just the classical Lax–Wendroff second order accurate three point scheme, see, e.g., [7]. For $k = 3, 4, 5$ the schemes are less studied, but are known to be L^2 stable, see, e.g., [9] and the references therein.

For variable coefficients the result is

$$\begin{aligned} u_{\Delta x}(x_\nu, t^{n+1}) &= R^{n,k}(x_\nu(t^n)) \\ &\quad + \int_{t^n}^{t^{n+1}} f(x_\nu(t^{n+1} - s))ds, \end{aligned} \tag{4.5a}$$

where $x_\nu(t)$ solves

$$\frac{dx_\nu}{dt} = a(x_\nu), \qquad t^n \le t \le t^{n+1}, \tag{4.5b}$$

$$x_\nu(t^{n+1}) = x_\nu. \tag{4.5c}$$

A fourth order Runge–Kutta method is used to integrate the ODE (4.5b), (4.5c) and the Simpson rule is used to evaluate the integral in (4.5a). The result of this approximation to the right side of (4.5a) is defined to be u^{n+1}_ν.

Returning to the present case, the computations ran 13 steps until $t = 8$, that is, $(SAS^{-1})^{2^{13}}$ was computed.

At each step n the number of elements of A^n and $(SAS^{-1})^n$ whose absolute values are greater than 10^{-4} is shown in Table 1. This is for methods whose order of accuracies go from 1 through 5. The compression ratio decreases with the order of accuracy of the scheme. The results are also plotted on Fig. 2.

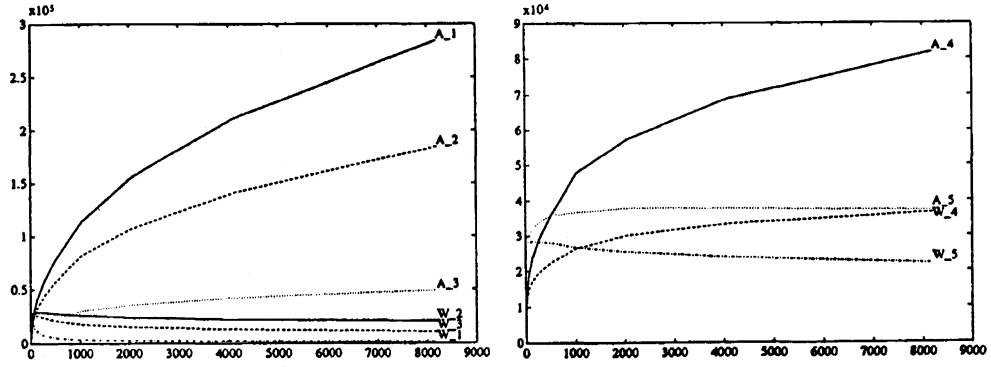


FIG. 2. *Hyperbolic equation: the number of elements in A^n and $(SAS^{-1})^n = w^n$ whose absolute values are greater than 10^{-4} for the first through fifth order accurate methods.*

TABLE 1
Hyperbolic equation: the number of elements in A^n and $(SAS^{-1})^n$ whose absolute values are greater than 10^{-4}.

Timesteps	Order 1 A^n	Order 1 $(SAS^{-1})^n$	Order 2 A^n	Order 2 $(SAS^{-1})^n$	Order 3 A^n	Order 3 $(SAS^{-1})^n$	Order 4 A^n	Order 4 $(SAS^{-1})^n$	Order 5 A^n	Order 5 $(SAS^{-1})^n$
1	2048	48564	3072	48174	4096	49538	5120	49280	6144	49988
2	3072	51172	5120	52916	7168	53394	9216	54260	11264	54442
4	5120	50258	9216	54404	11264	53618	13588	53628	14736	52840
8	9216	48830	13996	52734	15502	53014	18596	54018	18732	54038
16	14886	45326	20822	53192	18480	52766	23462	55992	21372	54778
32	21376	40110	27860	54700	21788	51988	28370	57582	24796	55582
64	29728	31538	37032	56650	25656	51808	34702	60710	28188	56050
128	41190	22160	49420	58668	30462	51294	42602	63630	32192	56458
256	56652	15268	66054	58828	36074	47750	52600	67522	36344	56298
512	78586	10320	89838	55850	43358	41950	66560	71408	41820	55614
1024	113954	6836	130366	52256	54574	35810	89112	73374	50456	53532
2048	155624	4640	173238	47966	63564	30254	107704	75466	56242	50956
4096	211340	3126	229458	44002	74226	25544	130084	75416	62384	47858
8192	284318	2104	304854	40970	84896	21204	155688	74288	68536	43734

These significant elements are located near the subdiagonal corresponding to the characteristic curve which is known a priori. As an example, for the equation $u_t + u_x = 0$ the Green function $\delta(x - y - t)$ has singular support $x = y + t$. Thus, we consider our discrete approximation to have singular support concentrated near the diagonal closest to $i = j + n\frac{\Delta t}{\Delta x}$ (mod N because of periodicity). The image of these locations in $(SAS^{-1})^n$, shown in Fig. 1, has total length of $O(N \log N)$ elements as $N \to \infty$. In this figure $n = 1024$.

In the computation of $(SAS^{-1})^n$, first, from the knowledge of the PDE, we figure out the structure of the singularities of A and its image in $(SAS^{-1})^n$. Then we compute $(SAS^{-1})^{2n} = (SAS^{-1})^n * (SAS^{-1})^n$ considering only the elements in a neighborhood of the singularities of A. In particular, we define the neighborhood of a singularity to be locations whose distance from the singularity are less than or equal to 5, 6, 7, and 8. If the singularities lie on a subdiagonal and its periodic extension, its neighborhood forms a subband of bandwidth 11 (the wavelet filters have 8 elements). This bandwidth is independent of the time t (the step n) and the size of the problem. The errors due to the subband truncation, measured by $\|u^n - \tilde{u}^n\|/\|u^n\|$, are shown in Table 2(b). Table 2(a) shows the relative error between the subband truncation and the exact solution. Here and throughout, "$\|\cdot\|$" denotes the ℓ^2 norm. Table 2(c) shows the relative error between the truncated subband and untruncated subband under grid refinement

for the various orders. Unsurprisingly, since the relative length of the subband that is preserved decreases linearly with grid size, the error increases, but only slightly, under this process.

TABLE 2

Hyperbolic equation: (a) *the error measured by* $\|u^n - \tilde{u}^n\|/\|u^n\|$ *compared with the exact solution*; (b) *due to the truncation only*; (c) *due to the truncation only under grid refinement.*

$B = 11$	Order 1	Order 2	Order 3	Order 4	Order 5	(a)
Error	.1621	.0125	.0080	.0129	.0121	
	Order 1	Order 2	Order 3	Order 4	Order 5	(b)
Error	.0044	.0122	.0077	.0127	.0119	
m	Order 1	Order 2	Order 3	Order 4	Order 5	(c)
1024	.0044	.0122	.0077	.0127	.0119	
512	.0030	.0081	.0062	.0079	.0073	
256	.0017	.0050	.0046	.0065	.0054	
128	.0006	.0040	.0021	.0032	.0031	
$B = 13$	Order 1	Order 2	Order 3	Order 4	Order 5	(a)
Error	.1602	.0079	.0056	.0125	.0082	
	Order 1	Order 2	Order 3	Order 4	Order 5	(b)
Error	.0025	.0075	.0055	.0124	.0081	
m	Order 1	Order 2	Order 3	Order 4	Order 5	(c)
1024	.0025	.0075	.0055	.0124	.0081	
512	.0019	.0052	.0040	.0057	.0058	
256	.0013	.0032	.0031	.0047	.0039	
128	.0005	.0024	.0013	.0018	.0019	
$B = 15$	Order 1	Order 2	Order 3	Order 4	Order 5	(a)
Error	.1604	.0068	.0045	.0090	.0090	
	Order 1	Order 2	Order 3	Order 4	Order 5	(b)
Error	.0019	.0064	.0042	.0089	.0089	
m	Order 1	Order 2	Order 3	Order 4	Order 5	(c)
1024	.0019	.0064	.0042	.0089	.0089	
512	.0011	.0041	.0030	.0041	.0042	
256	.0006	.0025	.0018	.0032	.0026	
128	.0003	.0020	.0007	.0013	.0012	
$B = 17$	Order 1	Order 2	Order 3	Order 4	Order 5	(a)
Error	.1600	.0034	.0024	.0072	.0063	
	Order 1	Order 2	Order 3	Order 4	Order 5	(b)
Error	.0009	.0028	.0022	.0071	.0062	
m	Order 1	Order 2	Order 3	Order 4	Order 5	(c)
1024	.0009	.0028	.0022	.0071	.0062	
512	.0004	.0017	.0012	.0020	.0020	
256	.0002	.0009	.0005	.0010	.0008	
128	.0000	.0005	.0000	.0002	.0000	

We note that the compression (as seen in Fig. 2 and Table 1) is better for odd order than for even order schemes. This is perhaps not surprising since (2.7) models schemes of odd order accuracy. Singularities behave a bit differently for even order (say, order $= 2p$) schemes.

These are modeled by

$$u_t + au_x = \ell_p(\Delta x)^{2p}\left(\frac{\partial}{\partial x}\right)^{2p+1} u + (-1)^p k_p(\Delta x)^{2p+1}\left(\frac{\partial}{\partial x}\right)^{2p+2} u, \tag{4.6}$$

where $k_p > 0$ and ℓ_p are nonzero constants. The odd order dispersive term above may tend to spread singularities of the fundamental solution spuriously.

Finally, Table 3 shows the relative error due to truncation when the bandwidth of the subband is 9, 11, and 13 for the methods of first and second order. Figures 3(a) and 3(b) compare the truncated versus the nontruncated approximate solutions due to truncation of bandwidth 9 for the first and second order methods (the truncated graphs are dotted). As described in the previous paragraph, the first order method has a smoother truncation error and is hence more compressible by the wavelet representation.

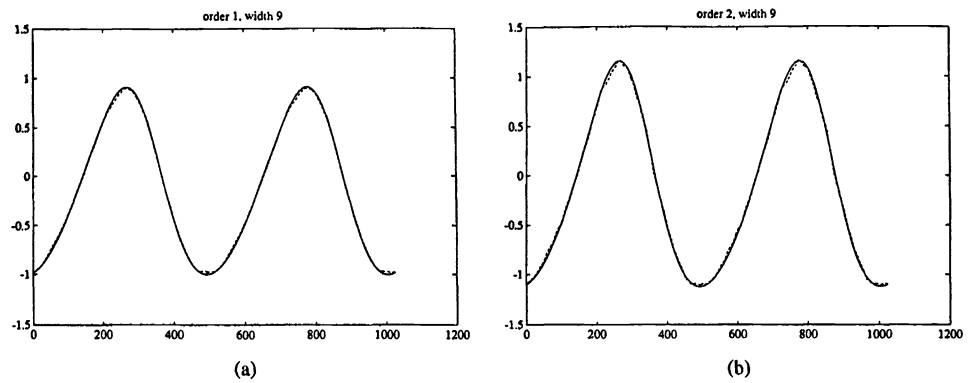


FIG. 3. (a) *Truncated versus nontruncated approximate solution, first order method truncated at bandwidth 9. (Truncated is dotted.)* (b) *Truncated versus nontruncated approximate solution, second order method, truncated at bandwidth 9. (Truncated is dotted.)*

4.2. Unstable schemes. For theoretical interest, we apply the method to a finite difference scheme that is unstable for $\frac{\Delta t}{\Delta x} = \lambda > 0$

$$u_j^{n+1} = u_j^n - \lambda(u_{j+1}^n - u_{j-1}^n)/2, \tag{4.7a}$$

$$u_j^0 = u_0(x_j). \tag{4.7b}$$

The amplification factor of this scheme is

$$1 - \lambda i \sin\theta = r(e^{i\theta}), \qquad -\pi < \theta \le \pi, \tag{4.8}$$

so

$$|r(e^{i\theta})| = (1 + \lambda^2 \sin^2\theta)^{\frac{1}{2}}.$$

This means that if

$$\Delta t \le 2c(\Delta x)^2 \tag{4.9}$$

TABLE 3

(a) *Errors measured by* $\|u^n - \bar{u}\|/\|u^n\|$ *due to truncation for various bandwidths and first through fifth orders.*
(b) *Errors measured by* $\|u_{\text{exact}} - \bar{u}\|/\|u_{\text{exact}}\|$ *due to truncation for various bandwidths and first through fifth orders.*

(a)	Order 1	Order 2	Order 3	Order 4	Order 5
$B = 9$	.0248	.0331	.0330	.0378	.0374
$B = 11$	.0044	.0122	.0077	.0127	.0119
$B = 13$	.0025	.0075	.0055	.0124	.0081
$B = 15$	.0019	.0064	.0042	.0089	.0089
$B = 17$	.0009	.0028	.0022	.0071	.0062
(b)	Order 1	Order 2	Order 3	Order 4	Order 5
$B = 9$	.1771	.0333	.0330	.0379	.0374
$B = 11$	.1621	.0125	.0080	.0129	.0121
$B = 13$	.1602	.0079	.0056	.0125	.0082
$B = 15$	.1604	.0068	.0045	.0090	.0090
$B = 17$	.1600	.0034	.0024	.0072	.0063

for some $c > 0$, then

$$\|A^n\|_{l^2} \leq e^{cn\Delta t}. \tag{4.10}$$

The restriction (4.9) means that the operation count for this explicit method would be $O(N^3)$ if we were silly enough to use it. However, our compression method allows for an operation count of $O(N(\log N)^3)$ for the reasons described above.

Table 4 shows the number of elements in A^n and $(SAS^{-1})^n$ whose absolute values are greater than 10^{-3}. We choose a bigger threshold here since we took $\frac{\Delta t}{(\Delta x)^2} = 1$ and $n\Delta t = 2$, so $\|A^n\|$, as estimated in (4.10), grows to be roughly e when we are finished computing.

The error as measured by $\frac{\|u^n - \bar{u}^n\|}{\|u^n\|}$ (subband truncation using bandwidth 11) was 0.0136.

We also performed convergence studies as we refined the grid for this method. Figures (4a)–4(c) compare the numerical (untruncated) solution using dots versus exact solution for $m = 128, 256, 512$ grid points. The result indicates a second order method, as it should, since $\Delta t = (\Delta x)^2$. Figures (5a)–(5c) compare the truncated bandwidth (using dots) versus the untruncated bandwidth for this method for $m \neq 128, 256$, and 512 grid points.

The relative error decreases with mesh refinement. The truncation error equation associated with this scheme involves limited antidiffusion. Perhaps this accounts for this behavior.

4.3. System of hyperbolic equations. We apply the method to solving the system of hyperbolic equations:

$$\partial_t \begin{bmatrix} v \\ w \end{bmatrix} + \begin{bmatrix} a & 0 \\ 0 & -a \end{bmatrix} \partial_x \begin{bmatrix} v \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.11a}$$

on $0 \leq x \leq 1, t \geq 0$ with the boundary conditions and initial conditions:

$$\begin{aligned} v(0,t) &= w(0,t), \\ w(1,t) &= v(1,t), \\ v(x,0) &= v_0(x) = \sin(4\pi x), \\ w(x,0) &= w_0(x) = \cos(2\pi x). \end{aligned} \tag{4.11b}$$

TABLE 4

Hyperbolic equation "unstable scheme": the number of elements in A^n and $(SAS^{-1})^n$ whose absolute values are greater than 10^{-3}.

n	A^n	$(SAS^{-1})^n$
1	512	512
2	750	512
4	1024	1336
8	1024	1764
16	1024	2328
32	1024	3060
64	1024	4026
128	2048	5273
256	2048	6302
512	2560	7447
1024	3432	8360
2048	4566	9308
4096	6330	9266
8192	9362	10557
16384	14332	13346
32768	23872	19255
65536	41490	29649
131072	74750	48595
262144	132916	84566
524288	132454	106197
1048576	132304	110240
2097152	130164	113276

The coefficient a is chosen to be constant:

$$a = 0.115.$$

The numerical method used is the first order accurate upwind method described above. The results are similar to the scalar case, except the structure of the singularities in the matrices is more complicated. We must keep track of reflections of singularities at the boundaries, which is quite simple in this case. The number of elements in A^n and $(SAS^{-1})^n$ whose absolute values are greater than 10^{-4} is shown in Table 5 and is plotted on Fig. 6. The relative error due to the subband of width 11 truncation, measured by $\|u^n - \tilde{u}^n\|/\|u^n\|$, is 0.0149.

The structure of the elements whose absolute values are greater than 10^{-4} of A^{2048} and $(SAS^{-1})^{2048}$ is shown in Figs. 7(a), (c), while Fig. 7(b) shows the image of a subband of bandwidth 11 in $(SAS^{-1})^{2048}$.

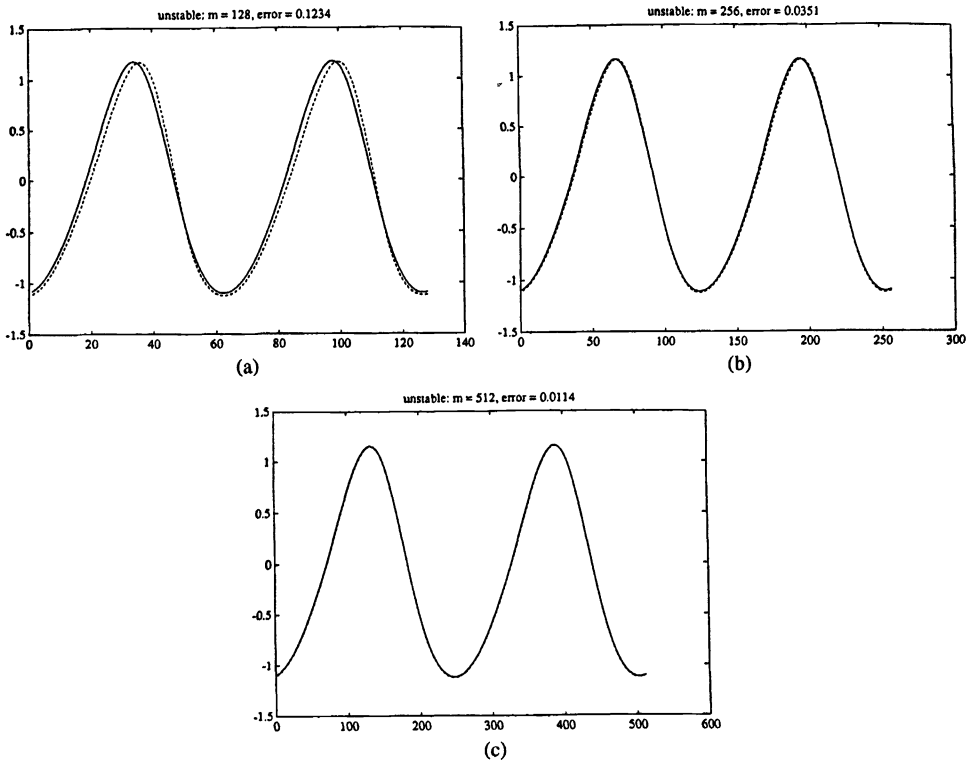


FIG. 4. (a) *Exact versus approximate solution,"unstable scheme," m* = 128. (b) *Exact versus approximate solution "unstable scheme," m* = 256. (c) *Exact versus approximate solution "unstable scheme," m* = 512.

TABLE 5

System of hyperbolic equations: The number of elements in A^n *and* $(SAS^{-1})^n$ *whose absolute values are greater than* 10^{-4}.

n	A^n	$(SAS^{-1})^n$
1	2048	19353
2	3074	22589
4	5126	25327
8	6154	26440
16	9228	25804
32	13332	25747
64	19488	22364
128	27692	18985
256	37945	14064
512	52308	10116
1024	72814	8110
2048	98456	5685

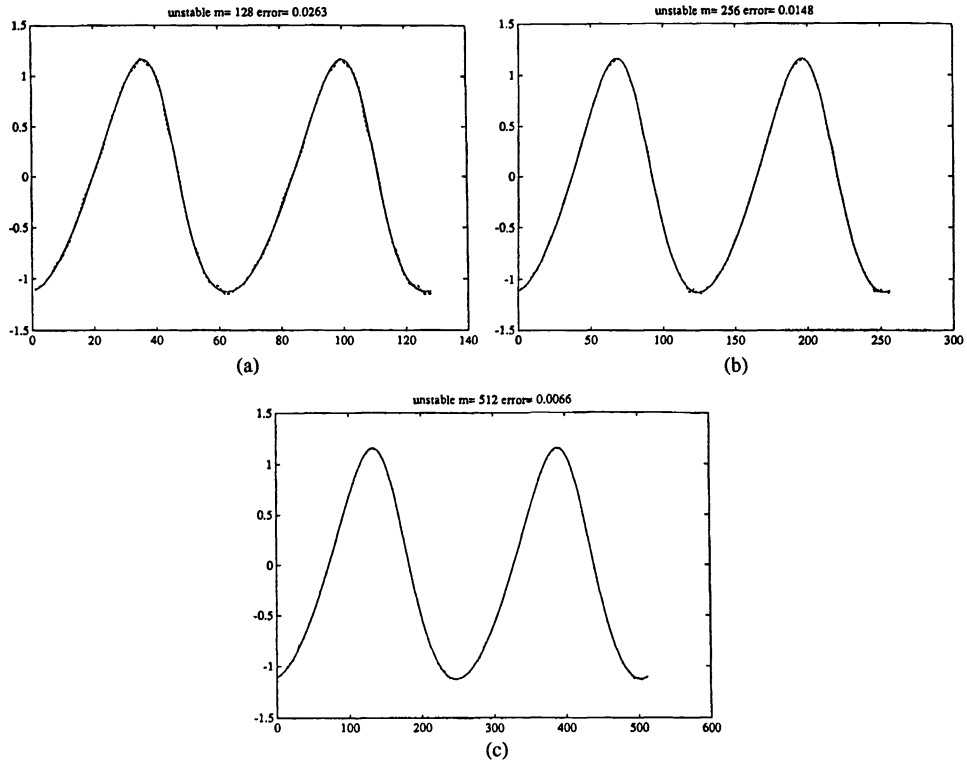


FIG. 5. (a) *Truncated bandwidth* 11 *versus untruncated for the "unstable scheme,"* $m = 128$. (b) *Truncated bandwidth* 11 *versus untruncated for the "unstable scheme,"* $m = 256$. (c) *Truncated bandwidth* 11 *versus untruncated for the "unstable scheme,"* $m = 512$.

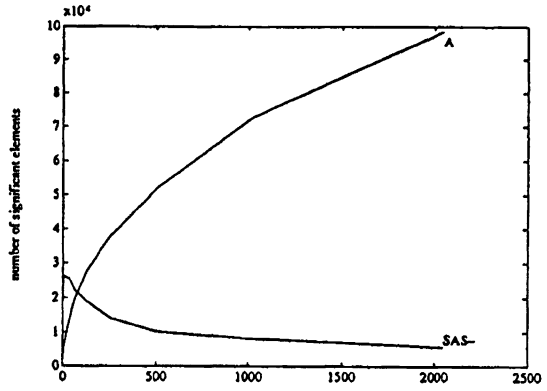


FIG. 6. *System of hyperbolic equations*: *The number of elements in* A^n *and* $(SAS^{-1})^n$ *whose absolute values are greater than* 10^{-4}.

4.4. Parabolic problems. We experiment on the following parabolic problem:

$$\begin{aligned} \partial_t u &= \partial_x(a(x)\partial_x u) + f(x), \\ u(x,0) &= u_0(x), \end{aligned} \tag{4.12}$$

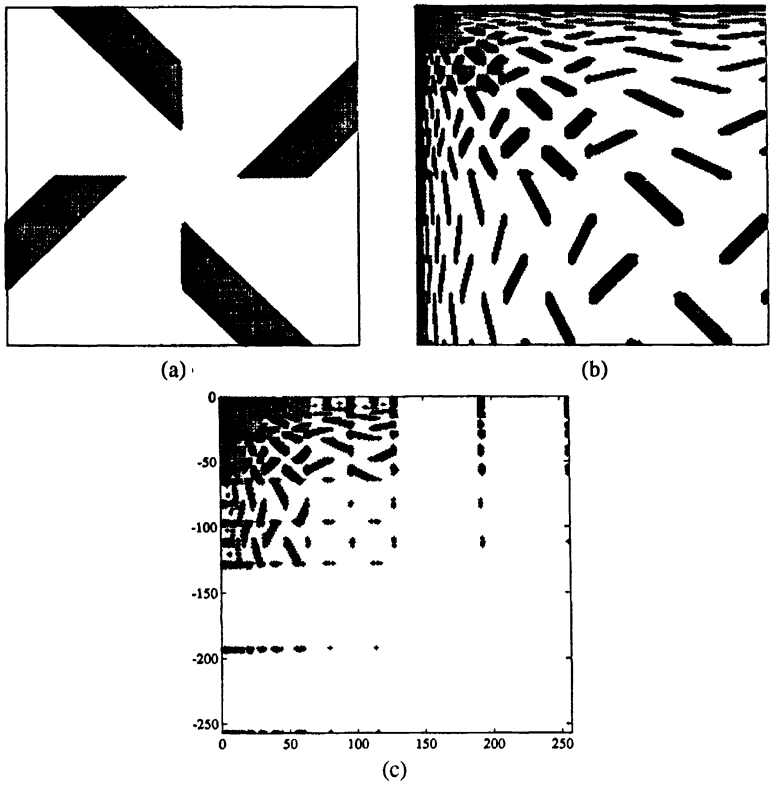


FIG. 7. (a) *System of hyperbolic equations pattern of significant elements* ($\geq 10^{-4}$) *for* A^n, $n = 2048$. (b) *System of hyperbolic equations pattern of significant elements for* $(SAS^{-1})^n$, $n = 2048$, *image of bandwidth* 11 *around singular support.* (c) *System of hyperbolic equations pattern of significant elements* ($\geq 10^{-4}$) *for* (SAS^{-1}), $n = 2048$.

with periodic boundary conditions ($0 \leq x \leq 1$). We made the following choices:

$$a(x) = 0.5 + 0.25\sin(2\pi x),$$
$$f(x) = -\pi^2\cos(2\pi x)^2 + \pi^2(0.5 + 0.25\sin(2\pi x))\sin(2\pi x),$$
$$u_0(x) = \sin(4\pi x).$$

The discrete setting and the wavelets are the same as in the hyperbolic problem. We use the simple explicit central difference scheme (4.13)

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{(\Delta x)^2}\Delta_-(a(x_j)\Delta_+ u_j) + \Delta t f(x_j), \tag{4.13}$$

where

$$\Delta_{\mp} u_j = \mp(u_{j\mp 1} - u_j)$$

with $\Delta t/(\Delta x)^2 = 0.25$. The number of significant elements in A^n and $(SAS^{-1})^n$ is shown in Table 6, and is plotted on Fig. 8.

For the parabolic problem, the large elements of A are in the neighborhood of the main diagonal. Their wavelet transform image is shown in Fig. 9. The relative error due to subband truncation was 0.0025.

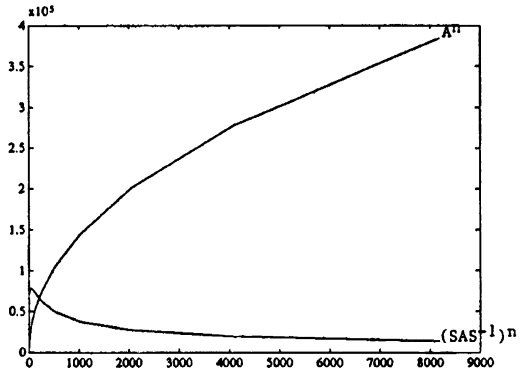


FIG. 8. *Parabolic equation: the number of elements in* A^n *and* $(SAS^{-1})^n$ *whose absolute values are greater than* 10^{-4}.

TABLE 6
Parabolic equation: the number of elements in A^n *and* $(SAS^{-1})^n$ *whose absolute values are greater than* 10^{-4}.

n	A^n	$(SAS^{-1})^n$
1	3072	15194
2	5120	17342
4	8462	19136
8	11682	19328
16	16214	18775
32	21900	17622
64	30126	14389
128	41434	10387
256	56756	7392
512	78078	5073
1024	106976	3554
2048	146466	2396
4096	199878	1658
8192	272050	1082

4.5. Two-dimensional parabolic problems. We consider the following problem:

$$\partial_t u = a_{11}\partial_{xx}u + 2a_{12}\partial_{xy}u + a_{22}\partial_{yy}u,$$
$$u(x, y, 0) = u_0(x, y)$$

with periodic boundary conditions ($0 \le x \le 1,\ 0 \le y \le 1$). We choose

$$a_{11}(x, y) = 0.5 + 0.25\sin(2\pi x),$$
$$a_{12}(x, y) = 0.115\sin(2\pi x)\cos(2\pi y),$$
$$a_{22}(x, y) = 0.5 + 0.25\cos(2\pi y),$$
$$u_0(x, y) = \sin(4\pi x) + \cos(8\pi x).$$

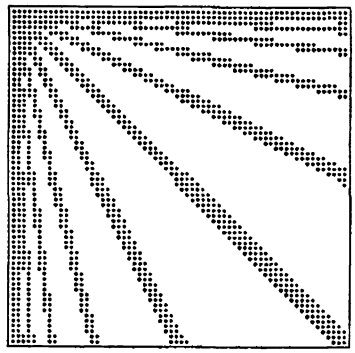


FIG. 9. *Parabolic equation: the pattern of significant elements in* $(SAS^{-1})^n$.

TABLE 7

Two-dimensional-parabolic equation: the number of elements in A^n *and* $(SAS^{-1})^n$ *whose absolute values are greater than* 10^{-4}.

n	A^n	$(SAS^{-1})^n$
1	6634	34190
2	16612	52941
4	40210	72420
8	72360	87381
16	105802	84827
32	146292	67912
64	198480	46856
128	269882	31925
256	365456	21497
512	491936	13653
1024	658800	8703
2048	891144	5271
4096	1048576	3373
8192	1048576	1981

We use a standard two-dimensional explicit central difference scheme. The two-dimensional data $u_{j,k}$, $j = 1, \ldots, N_1, k = 1, \ldots, N_2$ forms a one-dimensional vector in the following way

$$\{u_{1,1}, \ldots, u_{1,N_2}, u_{2,1}, \ldots, u_{2,N_2}, \ldots, u_{N_1,1}, \ldots, u_{N_1,N_2}\}.$$

To reduce the size of the problem, N_2 is much less than N_1. In particular we took $N_1 = 128$, $N_2 = 8$ that is, $\Delta x = \frac{1}{128}$, $\Delta y = \frac{1}{8}$.

The compression worked quite well. Table 7 shows the number of elements in A^n on $(SAS^{-1})^n$ whose absolute values are greater than 10^{-4}. The relative error due to subband truncation was 0.0066.

Acknowledgement. The authors would like to thank V. Rokhlin for suggesting both this problem and this approach to it, An Jiang for some help with the computations, and an anonymous referee for some constructive comments on the first draft.

REFERENCES

[1] G. BEYLKIN, R. COIFMAN, AND V. ROKHLIN, *Fast wavelet transforms and numerical algorithms* I, Comm. Pure Appl. Math., 44 (1991), pp. 141–184.

[2] I. DAUBECHIES, *Orthonormal basis of compactly supported wavelets*, Comm. Pure Appl. Math., 41 (1988), pp. 909–996.

[3] L. GREENGARD AND J. STRAIN, *A fast algorithm for the evaluation of heat potentials*, Comm. Pure Appl. Math., 43 (1990), pp. 949–964.

[4] ———, *The fast Gauss transform*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 79–94.

[5] G. HEDSTROM, *The rate of convergence of some difference schemes*, SIAM J. Numer. Anal., 5 (1968), pp. 363–406.

[6] Y. MADAY, V. PERRIER, AND J. C. RAVEL, *Adaptativité dynamique sur based'onde letts pour l'approximation d'équations aux dérivées partielles*, C. R. Acad. Sci., Paris, to appear.

[7] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial Value Problems*, Interscience Publishers, New York, 1967.

[8] V. ROKHLIN, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[9] A. ISERLES AND G. STRANG, *The optimal accuracy of difference schemes*, Trans. Amer. Math. Soc., 277 (1983), p. 779.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 776–797, July 1994

002

A SIMPLE ADAPTIVE GRID METHOD IN TWO DIMENSIONS*

WEIZHANG HUANG† AND DAVID M. SLOAN‡

Abstract. This paper gives an interpretation of the concept of equidistribution in the context of adaptive grid generation for multidimensional problems. It is shown that the equidistribution principle cannot be satisfied throughout the domain of the problem and, based on this recognition, a local equidistribution principle is developed. A discrete formulation is described for grid generation in two space dimensions and a smoothing mechanism is presented for improving mesh quality. The adaptive grid method that is constructed contains three grid-quality parameters. Numerical examples illustrate adaptive grid generation using a prescribed monitor function and grid generation for numerical solution of partial differential equations. Results show that the method produces high quality grids and that it is fairly insensitive to the choice of parameters.

Key words. adaptive grid, equidistribution

AMS subject classifications. 65L50, 65M50, 76M20

1. Introduction.

Numerical grid generation has become a valuable device for use in the numerical solution of partial differential equations. Many of the commonly used methods of generating computational grids are derivations of a technique first proposed by Winslow [15] in which a potential problem is solved, with mesh lines playing the role of equipotentials. Winslow's approach involves the solution of a nonlinear elliptic equation to generate a mapping from the computational domain to the physical domain. This idea was developed by J. F. Thompson and coworkers, among others, and some of these developments may be found in the text by Thompson, Warsi, and Mastin [14]. The initial developments from Winslow's method dealt with the construction of boundary-fitted coordinates for the solution of problems in irregular physical domains. The objective with these methods is to adjust the coordinate lines to match the geometry of the physical domain rather than adapt to the solution of the problem. This geometrical approach is not entirely satisfactory when a solution involves near-singular behaviour such as a boundary layer or a shock wave: in such cases the coordinate lines should be adapted to the features of the solution as well as to the geometry. Grid generators that are influenced by the solution are called adaptive grid generators.

Construction of adaptive grid generators is now an area of intense research activity, an activity that finds much of its motivation and application in computational fluid dynamics. A fairly common theme in methods used for adaptive grid generation is the idea of equidistribution, which seeks to distribute some function equally over the domain of the problem. The function is usually some measure of the computational error or the solution variation. Equidistribution has the effect of producing grids with spacing that is related to the local rate of change of the function.

The idea of equidistribution has been well defined for one-dimensional problems, but a proper description of the concept has not been produced for problems involving many dimensions. For two-dimensional problems useful progress has been made by Dwyer, Kee, and Sanders [7], Dwyer [8], and by Catherall [3]. They have used equidistribution by applying the concept in one dimension along sets of coordinate lines. In its simplest form this technique may produce grids of poor quality in terms of smoothness, skewness, and orthogonality. However, improvements in grid quality may be obtained if the equidistribution problem is

*Received by the editors August 12, 1992; accepted for publication (in revised form) April 22, 1993.

†Department of Mathematics, University of Strathclyde, Glasgow G1 1XH, United Kingdom and Institute of Applied Mathematics, Chinese Academy of Sciences, Beijing 100080, China (weizhang@cs.sfu.ca). The work of this author was supported by a Royal Fellowship Award from the Royal Society of London.

‡Department of Mathematics, University of Strathclyde, Glasgow G1 1XH, United Kingdom (caas10@ccsun.strath.ac.uk).

supplemented by conditions that deal with qualities such as smoothness and orthogonality. For example, Catherall [3] obtains good quality grids by conjoining his line-based equidistribution principle to Laplace and Poisson equations which control mesh spacings. Brackbill and Saltzman [1] used a variational approach to incorporate adaptivity (which may be regarded as a high-dimensional extension of the one-dimensional equidistribution principle) into Winslow's [15] method. They included terms in their variational formulation designed to improve grid properties such as smoothness and orthogonality. Dorfi and Drury [5], in considering one-dimensional problems, used a very effective device for incorporating smoothness into the equidistribution principle. Their one-dimensional device ensures that the ratio of adjacent grid intervals is restricted, thus controlling clustering and grid expansion. The power of the smoothing capability in [5] is clearly demonstrated in the valuable comparative study by Furzland, Verwer, and Zegeling [9].

The objective of this paper is to clarify the concept of equidistribution for multidimensional problems and to describe a simple method which produces high-quality meshes in two dimensions. The equidistribution principle is described in continuous form in §2.1 and in discrete form in §2.2. A mechanism for improving mesh quality, based on the ideas of Dorfi and Drury [5], is described in §2.3, and a simple 3-parameter generator is presented. Adaptive grid generation using a prescribed function is described in §3, and generation associated with the numerical solution of differential equations in two spatial dimensions is described in §4. Numerical examples are presented in §§3 and 4. Conclusions and comments on our 3-parameter method are contained in §5.

2. Simple grid adaption based on equidistribution principles.

2.1. Equidistribution principles. The equidistribution idea, first used by de Boor in 1974 [4], is the most important concept in the field of adaptive grid generation. Most adaptive grid methods are related in some way to this concept (see, for example, Hawken, Gottlieb, and Hansen [10]). However, although equidistribution is understood in one space dimension, it does not seem to have been well formulated mathematically in more than one dimension.

In this subsection we shall construct equidistribution principles based on the idea of equivariation of certain functions. Let $\mathbf{x} = [x_1, x_2, x_3]^T$ be the spatial coordinate in a three-dimensional physical domain, D_p, where the superscript T denotes transposition. Suppose we select a function u on D_p whose variation is being considered. To simplify the presentation, we introduce a one-to-one coordinate transformation from the computational domain D_c to the physical domain D_p as

$$\mathbf{x} = \mathbf{x}(\boldsymbol{\xi}), \qquad \boldsymbol{\xi} \in D_c, \tag{1}$$

where $\boldsymbol{\xi} = [\xi^1, \xi^2, \xi^3]^T$ denotes the spatial coordinate on the computational domain D_c. Obviously, for a prescribed mesh on the computational domain, the problem of finding a corresponding mesh on the physical domain will be that of determining a discrete transformation (1), which is defined for all $\boldsymbol{\xi}$ on the given mesh. Henceforth, the determination of a mesh, or a distribution of nodes, on D_p will be regarded as being equivalent to the construction of a (discrete) transformation (1). A distribution of the nodes of a mesh will hereinafter be referred to as a distribution.

It is well known (see, for example, Thompson, Warsi, and Mastin [14]) that the scaled arc-length measurement of variation of u along the element of arc from $\mathbf{x}$ to $\mathbf{x} + d\mathbf{x}$ can be expressed as

$$ds = [\alpha^2 (du)^2 + d\mathbf{x}^T\, d\mathbf{x}]^{1/2} = [d\mathbf{x}^T\, M\, d\mathbf{x}]^{1/2}, \tag{2}$$

where M is defined by

$$M = \alpha^2 \nabla u \cdot \nabla u^T + I, \tag{3}$$

$$\nabla u = \left[\frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial u}{\partial x_3} \right]^T, \tag{4}$$

and I is the unit matrix. An appreciation of scaled arc-length is more readily obtained by considering the one-dimensional situation, with x and ξ in D_p and D_c, respectively. An increment of arc-length, ds, on the solution curve $u = u(x)$ between locations x and $x + dx$ in D_p is given by

$$ds = [(du)^2 + (dx)^2]^{1/2} = \left[\left(\frac{du}{dx} \right)^2 + 1 \right]^{1/2} dx.$$

To obtain a scaled increment, which allows the relative effects of changes in u and x to be altered, we may replace the above ds by

$$ds = \left[\alpha^2 \left(\frac{du}{dx} \right)^2 + 1 \right]^{1/2} dx,$$

where $\alpha^2 \geq 0$ is a real parameter. If $\alpha^2 = 0$, the increment simply measures the change in x, and as α^2 increases the influence of the variation in u becomes more significant. Returning now to the multidimensional problem, if we use transformation (1), equation (2) can be expressed in computational coordinates as

$$ds = \left[d\boldsymbol{\xi}^T J^T M J d\boldsymbol{\xi} \right]^{1/2}, \tag{5}$$

where

$$J = \left[\frac{\partial \mathbf{x}}{\partial \xi^1}, \frac{\partial \mathbf{x}}{\partial \xi^2}, \frac{\partial \mathbf{x}}{\partial \xi^3} \right] \tag{6}$$

is the Jacobian of coordinate transformation (1).

The equidistribution principle follows from a simple observation on (5): if $u(\mathbf{x}(\boldsymbol{\xi}))$ is required to have the same variation ds along any element of arc in the computational domain which has fixed length $[d\boldsymbol{\xi}^T d\boldsymbol{\xi}]^{1/2}$, the term on the right-hand side of (5) must be independent of coordinate $\boldsymbol{\xi}$. Therefore, this equivariation requirement on u implies that $J^T M J$ should be independent of $\boldsymbol{\xi}$; that is,

$$[d\boldsymbol{\xi}^T J^T M J \, d\boldsymbol{\xi}]^{1/2} = [d\boldsymbol{\xi}^T \bar{M} d\boldsymbol{\xi}]^{1/2}, \tag{7}$$

where $\bar{M}$ is a 3×3 symmetric positive definite, and $\boldsymbol{\xi}$-independent, matrix. Thus, if a coordinate transformation (1) can be found to satisfy (7), u will have the same variation at any point on D_p along any arc that has length

$$\left[\left(\sum_{i=1}^{3} \frac{\partial \mathbf{x}}{\partial \xi^i} d\xi^i \right)^T \left(\sum_{j=1}^{3} \frac{\partial \mathbf{x}}{\partial \xi^j} d\xi^j \right) \right]^{1/2}.$$

In this sense, any one-to-one coordinate transformation (1) that satisfies (7) for some constant, symmetric positive definite matrix $\bar{M}$ will be called an *equidistribution*, and (7) will be called an *equidistribution principle* (in continuous form).

Now let's consider how we might use the equidistribution principle (7) to determine a distribution on D_p (physical mesh) corresponding to a given uniform mesh on D_c (computational mesh). The equidistribution principle (7) may be discretised by any suitable discretisation method, and this process yields an equation corresponding to each link between adjacent pairs of nodes on the computational mesh. The physical mesh may then be obtained by solving these equations, in conjunction with some discrete boundary conditions for the coordinate transformation (1). Some method has to be given for the determination of the constant matrix $\bar{M}$ in (7). In the following, we shall consider two common cases.

Case 1. The number of nodes is not restricted. This case often occurs in unstructured mesh generation and refinement mesh generation. Noting that M defined by (3) is a symmetric and positive definite matrix, we may choose the coordinate transformation (1) such that $\bar{M}$ has the special form

$$\bar{M} = c^2 I, \tag{8}$$

where $c > 0$ is a constant. For example, this can be achieved by choosing the computational coordinate system to coincide with the principal directions of $\bar{M}$. Therefore, (7) takes the form

$$\left[d\boldsymbol{\xi}^T J^T M J d\boldsymbol{\xi}\right]^{1/2} = c\left[d\boldsymbol{\xi}^T d\boldsymbol{\xi}\right]^{1/2}. \tag{9}$$

In this case, c may be given in advance. Similar ideas have been used for the adaptive generation of unstructured meshes (see, for example, Peraire et al. [13], Bristeau et al. [2]).

Case 2. The number of nodes along any coordinate line is given in advance and the global structure of the mesh is also required. This is the case in structured movement (or redistributed) mesh generation. Usually (7) cannot be satisfied by the coordinate transformation (1) (or by a mesh on D_p) on the whole computational domain. In fact, this can be seen easily from the following analysis.

Suppose (7) can be satisfied by some coordinate transformation on the whole computational domain. Then (7) must hold along any coordinate line. For example, along the coordinate line

$$l_1 : \xi^2 = \text{constant}, \qquad \xi^3 = \text{constant}, \tag{10}$$

(7) reads as

$$\left[\left(\frac{\partial \mathbf{x}}{\partial \xi^1}\right)^T M \left(\frac{\partial \mathbf{x}}{\partial \xi^1}\right)\right]^{1/2} = c_1, \tag{11}$$

where c_1 is a positive constant. On the other hand, the equidistribution along line l_1 can be regarded as a one-dimensional case. Therefore, the assumption that the number of nodes along l_1 is prescribed implies that c_1 can be determined by the equidistribution, and this gives

$$c_1 = \frac{\int \left[\left(\frac{\partial \mathbf{x}}{\partial \xi^1}\right)^T M \left(\frac{\partial \mathbf{x}}{\partial \xi^1}\right)\right]^{1/2} d\xi^1}{\int d\xi^1}, \tag{12}$$

where the integration is along l_1 in D_c. If the number of nodes on l_1 is prescribed in advance, then the denominator on the right-hand side of (12) is fixed at a value determined by this number. Hence, when the number of nodes along any coordinate line is given in advance, it may be seen that c_1 will usually be a function of ξ^2 and ξ^3. Since c_1 in (11) is assumed to be constant over the entire computational domain it follows that there does not exist an equidistribution on D_p for this case. However, if we weaken (7) and require the coordinate transformation (1) only to satisfy (7) locally, it will be possible to find an equidistribution on D_p. Of course, this equidistribution is only *locally-equidistributed.* Here we consider (7) to be satisfied along any coordinate line: for example, along line l_1 we have (11), but with c_1 now a function of ξ^2 and ξ^3. Similarly, along the other two coordinate lines analogous formulae can be obtained. Hence, we have

$$\begin{cases} \left[\left(\frac{\partial \mathbf{x}}{\partial \xi^1}\right)^T M \left(\frac{\partial \mathbf{x}}{\partial \xi^1}\right)\right]^{1/2} = c_1(\xi^2, \xi^3), \\ \left[\left(\frac{\partial \mathbf{x}}{\partial \xi^2}\right)^T M \left(\frac{\partial \mathbf{x}}{\partial \xi^2}\right)\right]^{1/2} = c_2(\xi^3, \xi^1), \\ \left[\left(\frac{\partial \mathbf{x}}{\partial \xi^3}\right)^T M \left(\frac{\partial \mathbf{x}}{\partial \xi^3}\right)\right]^{1/2} = c_3(\xi^1, \xi^2). \end{cases} \tag{13}$$

The equations in (13) give the essential formulae in our simple adaptive grid method. The idea of using a curve-based equidistribution principle is not new, having been used by Dwyer, Kee, and Sanders [7], Dwyer [8], and Catherall [3]. However, it does not seem to have been presented in a form as simple as that offered by (13).

It is instructive to look at the two- and one-dimensional forms of (13).

Two Dimensions

$$\begin{cases} \left[\begin{pmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \end{pmatrix}^T M \begin{pmatrix} \frac{\partial x}{\partial \xi} \\ \frac{\partial y}{\partial \xi} \end{pmatrix}\right]^{1/2} = c_1(\eta), \\ \left[\begin{pmatrix} \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \eta} \end{pmatrix}^T M \begin{pmatrix} \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \eta} \end{pmatrix}\right]^{1/2} = c_2(\xi), \end{cases} \tag{14}$$

where

$$M = \alpha^2 \begin{bmatrix} \left(\frac{\partial u}{\partial x}\right)^2 & \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} \\ \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} & \left(\frac{\partial u}{\partial y}\right)^2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{15}$$

and (ξ, η) denotes the spatial coordinate in D_c.

One Dimension

$$\left[\left(\frac{dx}{d\xi}\right)^2 M\right]^{1/2} = c, \tag{16}$$

where

$$M = 1 + \alpha^2 \left(\frac{du}{dx}\right)^2, \tag{17}$$

and ξ is the spatial coordinate in D_c. Equation (16) is the well-known, one-dimensional arc-length equidistribution principle.

The remainder of this paper deals with structured, redistributed, adaptive mesh generation in two dimensions.

2.2. Finite difference discretisation of the two-dimensional equidistribution principle. In this subsection, a finite difference discretisation of (14) will be presented. Suppose that a uniform mesh is given on D_c by

$$\begin{cases} \xi_i = i, & i = 0, \dots, n, \\ \eta_j = j, & j = 0, \dots, m, \end{cases} \tag{18}$$

where n and m denote the numbers of intervals in the ξ and η directions, respectively. Equations (14) are discretized on the internodal links containing $(i + \frac{1}{2}, j)$ and $(i, j + \frac{1}{2})$, respectively, to give

$$\left\{ \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ y_{i+1,j} - y_{i,j} \end{bmatrix}^T M_{i+\frac{1}{2},j} \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ y_{i+1,j} - y_{i,j} \end{bmatrix} \right\}^{1/2} = c_1(\eta_j), \tag{19}$$
$$i = 0, \dots, n-1, \qquad j = 1, \dots, m-1,$$

and

$$\left\{ \begin{bmatrix} x_{i,j+1} - x_{i,j} \\ y_{i,j+1} - y_{i,j} \end{bmatrix}^T M_{i,j+\frac{1}{2}} \begin{bmatrix} x_{i,j+1} - x_{i,j} \\ y_{i,j+1} - y_{i,j} \end{bmatrix} \right\}^{1/2} = c_2(\xi_i), \tag{20}$$
$$j = 0, \dots, m-1, \qquad i = 1, \dots, n-1,$$

where $M_{i+1/2,j} = M(\xi_{i+1/2}, \eta_j)$ and $M_{i,j+1/2} = M(\xi_i, \eta_{j+1/2})$. Equations (19) and (20) may be rewritten as

$$\left\{ \begin{bmatrix} x_{i,j} - x_{i-1,j} \\ y_{i,j} - y_{i-1,j} \end{bmatrix}^T M_{i-\frac{1}{2},j} \begin{bmatrix} x_{i,j} - x_{i-1,j} \\ y_{i,j} - y_{i-1,j} \end{bmatrix} \right\}^{1/2} - \left\{ \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ y_{i+1,j} - y_{i,j} \end{bmatrix}^T M_{i+\frac{1}{2},j} \begin{bmatrix} x_{i+1,j} - x_{i,j} \\ y_{i+1,j} - y_{i,j} \end{bmatrix} \right\}^{1/2} = 0, \tag{21}$$
$$i = 1, \dots, n-1, \qquad j = 1, \dots, m-1,$$

$$\left\{ \begin{bmatrix} x_{i,j} - x_{i,j-1} \\ y_{i,j} - y_{i,j-1} \end{bmatrix}^T M_{i,j-\frac{1}{2}} \begin{bmatrix} x_{i,j} - x_{i,j-1} \\ y_{i,j} - y_{i,j-1} \end{bmatrix} \right\}^{1/2} - \left\{ \begin{bmatrix} x_{i,j+1} - x_{i,j} \\ y_{i,j+1} - y_{i,j} \end{bmatrix}^T M_{i,j+\frac{1}{2}} \begin{bmatrix} x_{i,j+1} - x_{i,j} \\ y_{i,j+1} - y_{i,j} \end{bmatrix} \right\}^{1/2} = 0, \tag{22}$$
$$j = 1, \dots, m-1, \qquad i = 1, \dots, n-1.$$

Regarding the discretisation of matrix M, it needs to be transformed into computational coordinates and then discretised on the computational domain. To be more specific, M is transformed into

$$M(\xi, \eta) = \alpha^2 \begin{bmatrix} \frac{u_\xi y_\eta - u_\eta y_\xi}{\sqrt{g}} \\ \frac{-u_\xi x_\eta + u_\eta x_\xi}{\sqrt{g}} \end{bmatrix} \cdot \left[\frac{u_\xi y_\eta - u_\eta y_\xi}{\sqrt{g}}, \frac{-u_\xi x_\eta + u_\eta x_\xi}{\sqrt{g}} \right] + I, \tag{23}$$

where $\sqrt{g} = x_\xi y_\eta - x_\eta y_\xi$. $M(\xi, \eta)$ can now be discretised at point $(\xi_{i+1/2}, \eta_{j+1/2})$ by second-order central differences. For example, in M, u_ξ and u_η can be discretised by

$$u_\xi\left(\xi_{i+\frac{1}{2}}, \eta_{j+\frac{1}{2}}\right) = \frac{1}{2}\left(u_{i+1,j} - u_{i,j} + u_{i+1,j+1} - u_{i,j+1}\right)$$

and

$$u_\eta\left(\xi_{i+\frac{1}{2}}, \eta_{j+\frac{1}{2}}\right) = \frac{1}{2}\left(u_{i,j+1} - u_{i,j} + u_{i+1,j+1} - u_{i+1,j}\right),$$

respectively, where $u_{i,j} = u(x_{i,j}, y_{i,j}) = u\big(x(\xi_i, \eta_j), y(\xi_i, \eta_j)\big)$.

Then $M_{i+1/2,j}$ and $M_{i,j+1/2}$ can be obtained by

$$\begin{aligned} M_{i+\frac{1}{2},j} &= \frac{1}{2}\left(M_{i+\frac{1}{2},j+\frac{1}{2}} + M_{i+\frac{1}{2},j-\frac{1}{2}}\right), \\ M_{i,j+\frac{1}{2}} &= \frac{1}{2}\left(M_{i+\frac{1}{2},j+\frac{1}{2}} + M_{i-\frac{1}{2},j+\frac{1}{2}}\right). \end{aligned} \tag{24}$$

2.3. Smoothness of mesh and the simple grid adaption. It is known that the mesh quality, such as smoothness of mesh and orthogonality along boundaries, is another important aspect of grid adaption (see, for example, Thompson, Warsi, and Mastin [14]). As seen in previous subsections, the equidistribution principle does not offer any mechanism to handle the mesh quality. Some mechanism should therefore be incorporated into the equidistribution principle in order to control the quality. In the popular variational approach to grid adaption, the balance between grid adaptivity and mesh quality is introduced by using a linear combination of terms associated with adaptivity, smoothness, and orthogonality. Recently, the same idea was used by Catherall [3]. He combines the equidistribution principle linearly with a Laplace and a Poisson equation for the nodal locations, with the combination designed to improve the grid quality. Suitable choices of the balance parameters can give good quality meshes. However, the proper balance depends on the physical problem and the ideal selection strategy is still an open question. An insensitive way to smooth the mesh in one dimension has been used by Dorfi and Drury [5] (see also Furzland, Verwer, and Zegeling [9]). They achieve their objective by smoothing the matrix M given by (17).

Here we follow the approach proposed by Dorfi and Drury [5]. In this approach, matrices (21) and (22) are replaced by smoothing matrices

$$\begin{aligned} \tilde{M}_{i+\frac{1}{2},j} &= \sum_{k=i-1}^{i+1}\sum_{l=j-1}^{j+1} M_{k+\frac{1}{2},l}\left(\frac{\gamma}{\gamma+1}\right)^{|k-i|+|l-j|}, \\ \tilde{M}_{i,j+\frac{1}{2}} &= \sum_{k=i-1}^{i+1}\sum_{l=j-1}^{j+1} M_{k,l+\frac{1}{2}}\left(\frac{\gamma}{\gamma+1}\right)^{|k-i|+|l-j|}, \end{aligned} \tag{25}$$

where γ, called the smoothing parameter, is a positive constant. The summations in (25) are understood to contain only elements that are well defined. It is of interest to note that the smoothing introduced by (25) couples three coordinate lines in each direction ($l = j - 1, j, j + 1$ in the first equation of (25), for example). The smoothing process adds numerical dissipation, rather like an increase in viscosity in a fluid flow—a process that reduces the tendency for fluid shear. The analogy here is that the smoothing produces an effect that prevents the mesh from skewing. Using (25), equations (21), and (22) read as

$$
\begin{cases}
\left\{\begin{bmatrix} x_{i,j}-x_{i-1,j} \\ y_{i,j}-y_{i-1,j}\end{bmatrix}^T \tilde{M}_{i-\frac{1}{2},j} \begin{bmatrix} x_{i,j}-x_{i-1,j} \\ y_{i,j}-y_{i-1,j}\end{bmatrix}\right\}^{1/2} \\
\quad -\left\{\begin{bmatrix} x_{i+1,j}-x_{i,j} \\ y_{i+1,j}-y_{i,j}\end{bmatrix}^T \tilde{M}_{i+\frac{1}{2},j} \begin{bmatrix} x_{i+1,j}-x_{i,j} \\ y_{i+1,j}-y_{i,j}\end{bmatrix}\right\}^{1/2} = 0, \\
\left\{\begin{bmatrix} x_{i,j}-x_{i,j-1} \\ y_{i,j}-y_{i,j-1}\end{bmatrix}^T \tilde{M}_{i,j-\frac{1}{2}} \begin{bmatrix} x_{i,j}-x_{i,j-1} \\ y_{i,j}-y_{i,j-1}\end{bmatrix}\right\}^{1/2} \\
\quad -\left\{\begin{bmatrix} x_{i,j+1}-x_{i,j} \\ y_{i,j+1}-y_{i,j}\end{bmatrix}^T \tilde{M}_{i,j+\frac{1}{2}} \begin{bmatrix} x_{i,j+1}-x_{i,j} \\ y_{i,j+1}-y_{i,j}\end{bmatrix}\right\}^{1/2} = 0, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 1,\dots,n-1, \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 1,\dots,m-1.
\end{cases} \tag{26}
$$

The adaptive grid method presented here consists of (3), (25), (26), and suitable boundary conditions at $\xi = \xi_0, \xi_n$ and $\eta = \eta_0, \eta_m$. It should be noted that low-order discretisations are often used along the boundaries, and these should take the form of mesh orthogonality conditions.

In practical calculations involving very stiff problems it is found that dangerously small physical grid spacings and mesh tangling ("overkill") can occur when $|\nabla u|$ becomes too large somewhere. This problem can be avoided by imposing a cut-off condition on ∇u in (3). That is, (3) is replaced by

$$
M = \frac{\alpha^2 \nabla u \,.\, \nabla u^T}{1 + \beta \nabla u^T \nabla u} + I, \tag{27}
$$

where β, called a scaling parameter, is a nonnegative constant. Equation (27) implies that

$$
\|M\|_2 \le 1 + \frac{\alpha^2}{\beta} \tag{28}
$$

for positive β. Therefore, positive β imposes minimum values on the increments of x and y along every computational coordinate line and thus obviates "overkill."

Thus, the three parameters α (concentration), β (scaling), and γ (smoothing) are introduced in the current adaptive grid method. The sensitivity of choice of these parameters will be discussed in the following sections.

3. Grid generation for given analytic functions. Before we apply the adaptive grid method to numerical solutions of differential equations, it will be useful to look at how to choose values of the parameters and how good meshes can be produced for some given function u. Two functions defined on a rectangular physical domain will be considered in this section. Dirichlet and Neumann (orthogonal) boundary conditions for x and y are used, that is,

$$
\begin{cases}
\begin{cases} x_{0,j} = 0, \; x_{n,j} = 1, \\ y_{1,j} - y_{0,j} = 0, \; y_{n,j} - y_{n-1,j} = 0, \end{cases} & j = 0,\dots,m, \\
\begin{cases} x_{i,1} - x_{i,0} = 0, \; x_{i,m} - x_{i,m-1} = 0, \\ y_{i,0} = 0, \; y_{i,m} = 1. \end{cases} & i = 1,\dots,n-1,
\end{cases} \tag{29}
$$

The nonlinear system (29) and (26) is solved by homotopy and a Newton–Raphson method, wherein α serves as continuation parameter ($0 \to \alpha$). The Jacobian is calculated by forward

differences and linear systems arising from Newton linearisation are solved by Gaussian elimination. The uniform mesh is used as the initial mesh.

Example 3.1. In this example we use the analytic function

$$u(x, y) = [1 - e^{R(x-1)}] \sin(\omega y). \tag{30}$$

This function is the exact solution of the convection-diffusion equation which is considered in §4 (Example 4.1) and it has been used for grid generation by Dvinsky [6]. When R is large, $u(x, y)$ has a boundary layer near $x = 1$. In this example, the scaling effect has not been considered ($\beta = 0$).

First, let us consider the case $R = 15$ and $\omega = \pi$. The exact function (on physical domain), meshes, and functions evaluated on these meshes are shown in Fig. 3.1. From Fig. 3.1 we make some observations:

a. Adapted meshes can be obtained by the equidistribution principle. However, meshes may be very unsmooth and skewed if $\gamma = 0$ (without control of smoothness).
b. The method described in §2.2 offers a good mechanism for handling mesh smoothness. The choice of values of γ is not sensitive.
c. The larger the values of α, the more adaptive the mesh becomes. The choice of values of α depends on how much adaptivity is needed. Usually, $\alpha = 1$ will suffice.
d. The more grid points we use, the better the quality of the generated mesh.

Adapted meshes and functions on these meshes are shown in Figs. 3.2 and 3.3 for the case $R = 35, \omega = \pi$, and the case $R = 15, \omega = 1.5\pi$, respectively. These figures confirm our observations.

Note that with Neumann boundary conditions, orthogonality of the grid lines is imposed at the boundary, and this may lead to discontinuities in grid line slope at points adjacent to the boundary. This did not give rise to noticeable inaccuracies for the problems considered here since the resulting skewness at points adjacent to the boundaries was never too severe. The slope discontinuities could be avoided if the Neumann conditions were replaced by an imposition of the equidistribution principle on the boundary lines.

Example 3.2. Here we use the analytic function

$$u(x, y) = \tanh\left[R\left(\frac{1}{16} - \left(x - \frac{1}{2}\right)^2 - \left(y - \frac{1}{2}\right)^2\right)\right]. \tag{31}$$

This is a more difficult problem than Example 3.1 because the mesh needs to be adapted in all directions. The function, with several values of R, is plotted in Fig. 3.4.

Adapted meshes obtained by the adaptive grid method described in §2 with $\alpha = 1, \gamma = 2$, and $\beta = 0$ (no scaling) are shown in Fig. 3.5. Although a good mesh can be obtained for $R = 15$, the "overkill" phenomenon appears for larger $R(= 20)$ near the four corners, where $\nabla u^T \nabla u$ is very large. As suggested in §2.2, the "overkill" phenomenon may be avoided by cutting off (or scaling) ∇u. Figures 3.6 and 3.7 show adapted meshes obtained with $\beta = 0.01$ (five times smaller than $\frac{1}{n} = \frac{1}{m} = 0.05$) and with $\beta = 0.005$ (ten times smaller than $\frac{1}{n} = \frac{1}{m} = 0.05$), respectively. It can be seen that the "overkill" phenomenon disappears even for much larger $R(= 100)$ and good quality meshes can be obtained for a wide range of values of β. It is also interesting to notice from Figs. 3.5–3.7 that positive β provides a safe guarantee for large values of R.

In summary, it has been shown that adapted meshes with good quality can be obtained for given analytic monitor functions by using the simple adaptive grid method described in §2. The ideal values of the parameters α, β, and γ will depend on the monitor function. The method is fairly insensitive to the choice of values of the parameters, however, and we

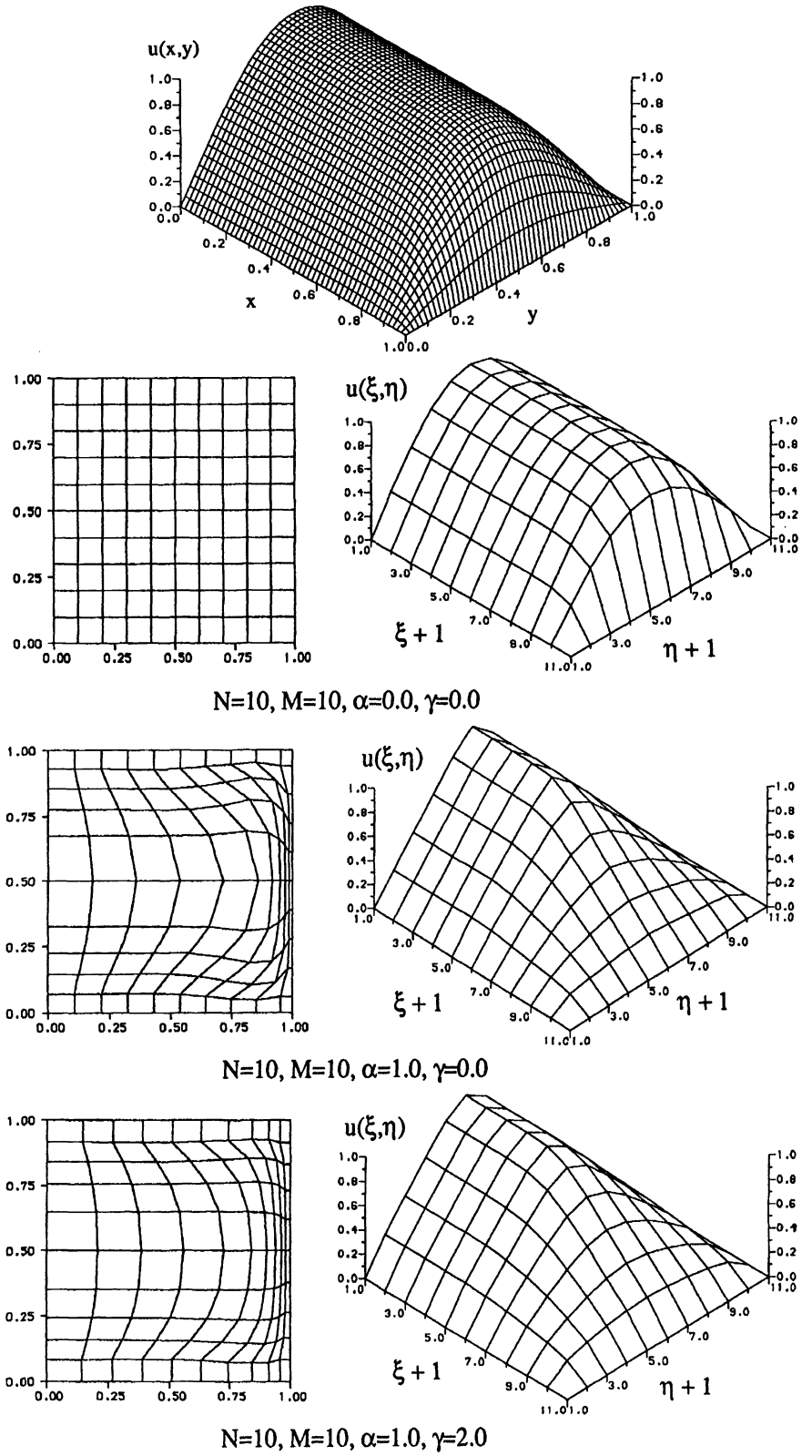
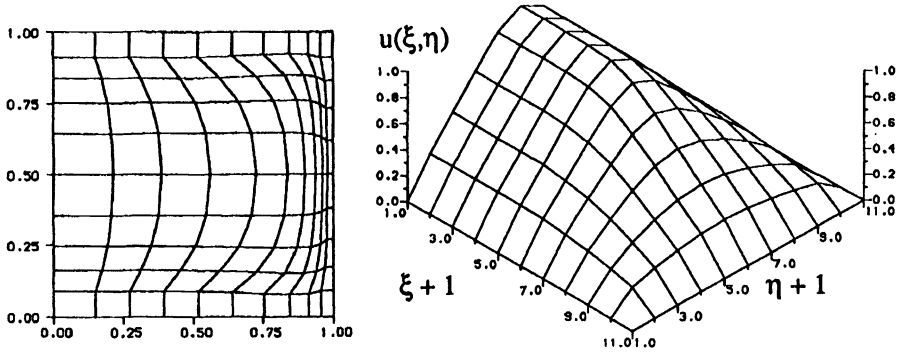
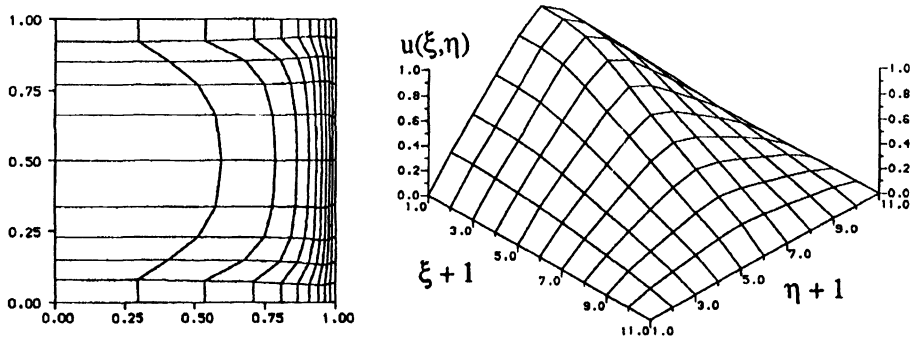


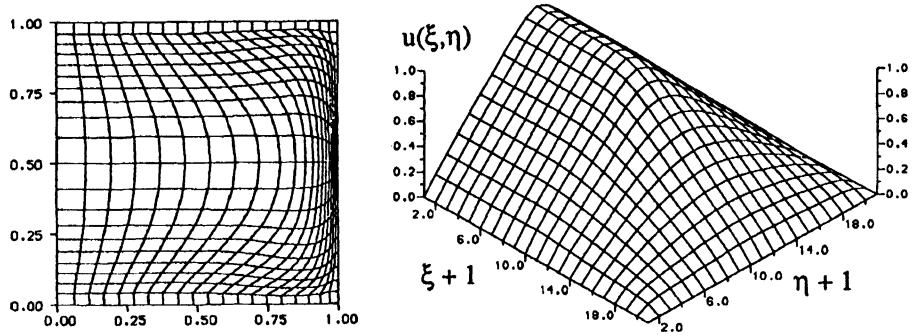
FIG. 3.1. *Solution and meshes for Example* 3.1 *with* $R = 15.0$, $\omega = \pi$.



N=10, M=10, α=1.0, γ=100.0



N=10, M=10, α=5.0, γ=2.0



N=20, M=20, α=1.0, γ=2.0

FIG. 3.1. *Continued.*

experienced no trouble in selecting suitable values, even in the difficult case of Example 3.2 with $R \geq 20$.

4. Grid adaption for numerical solutions of differential equations in two dimensions. In this section, the adaptive grid method described in §2 is applied to numerical solutions of the convection-diffusion equation given below.

Example 4.1. Consider

$$R\frac{\partial u}{\partial x} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \omega^2\left[1 - e^{R(x-1)}\right]\sin(\omega y), \qquad 0 < x, y < 1, \tag{32}$$

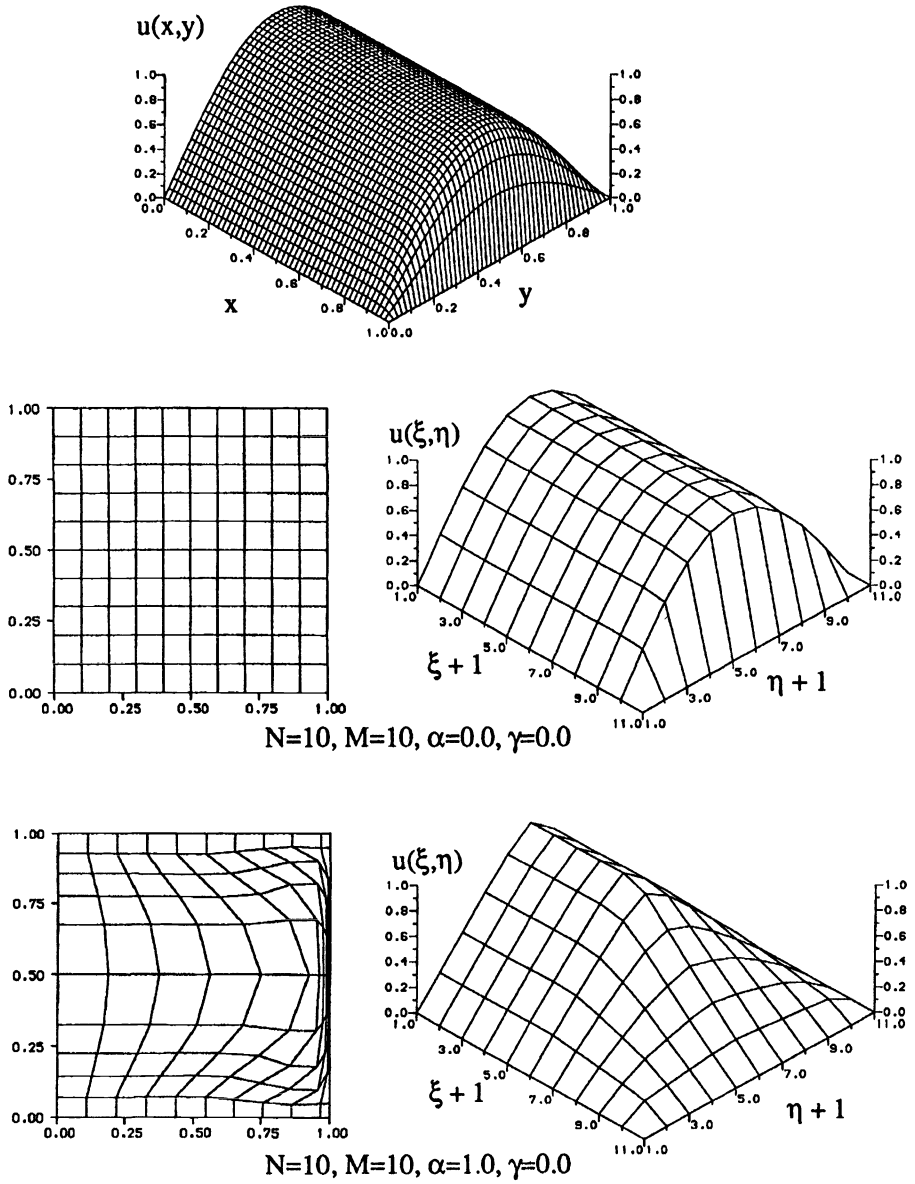


FIG. 3.2. *Solution and meshes for Example* 3.1 *with* $R = 35.0$, $\omega = \pi$.

subject to Dirichlet boundary conditions that are chosen such that (32) has the exact solution

$$u^e(x, y) = \left[1 - e^{R(x-1)}\right] \sin(\omega y). \tag{33}$$

This differential equation has been used by Dvinsky [6] to show the efficiency of his adaptive grid method.

In our calculations, (32) is first transformed into computational coordinates and then discretised on the uniform mesh (18) by first-order upwind differences. The resulting algebraic equations will hereinafter be called discrete convection-diffusion equations (DCDE) for simplicity of description. The equations (26), (29), and DCDE constitute the algebraic system

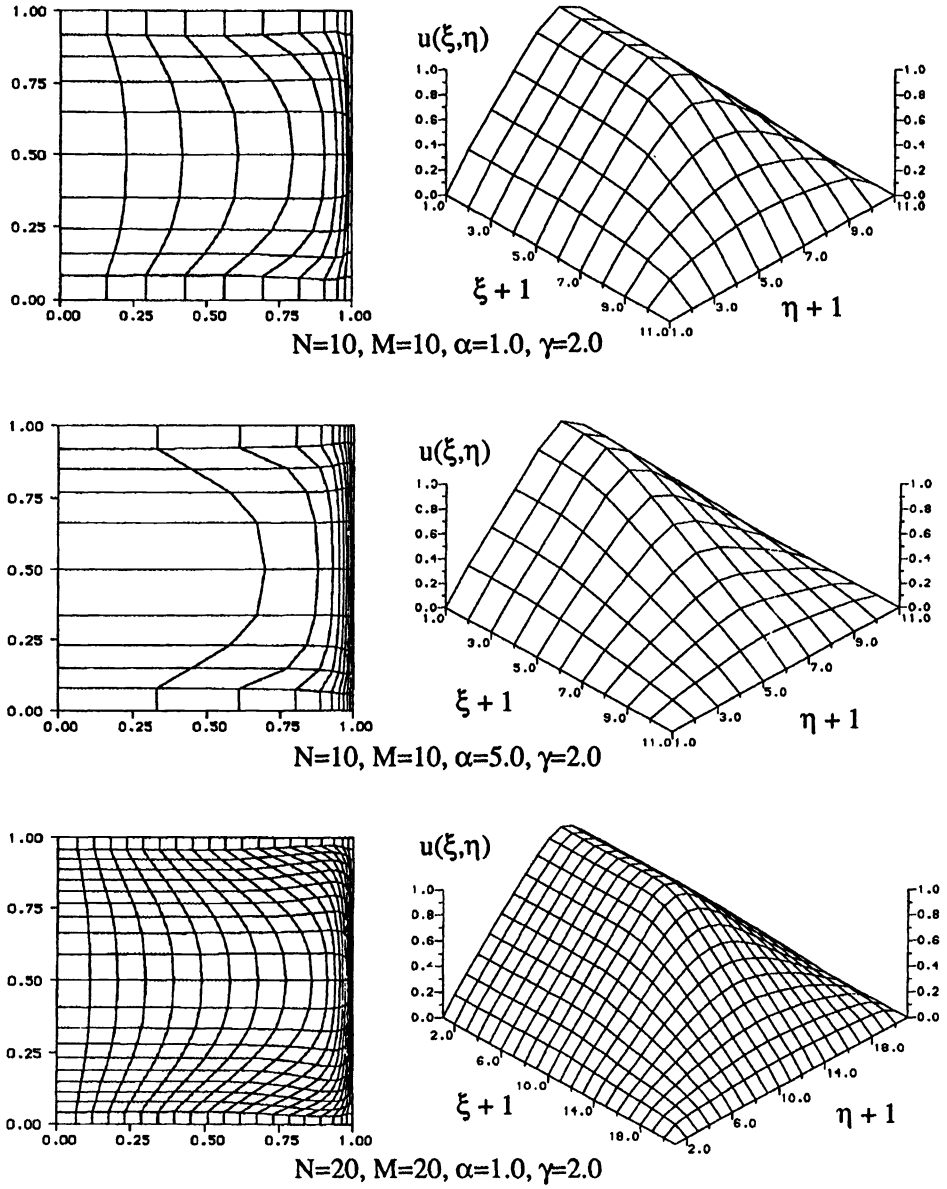


FIG. 3.2. *Continued.*

that is needed to determine the unknowns $u_{i,j}, x_{i,j}$, and $y_{i,j}$, $i = 0, \ldots, n$, $j = 0, 1, \ldots, m$. Obviously, the equations in this algebraic system should be solved simultaneously. However, the complete system is dealt with in an uncoupled way in this section in order to reduce the scales of algebraic systems to be solved. To be more specific, DCDE is solved by the Gaussian elimination method for given $x_{i,j}$ and $y_{i,j}$ $(i = 0, \ldots, n; j = 0, \ldots, m)$, and (26) and (29) are solved by the Newton–Raphson continuation method described in §3 for given $u_{i,j}$ $(i = 0, \ldots, n; j = 0, \ldots, m)$. It should be noticed that during the solution of (26) and (29), an interpolation procedure must be used for u because the computed values of u are given only on the previous mesh $(x_{i,j}, y_{i,j})$ $(i = 0, \ldots, n; j = 0, \ldots, m)$. Routines E02SAF and E02SBF from the NAG library are used for this purpose.

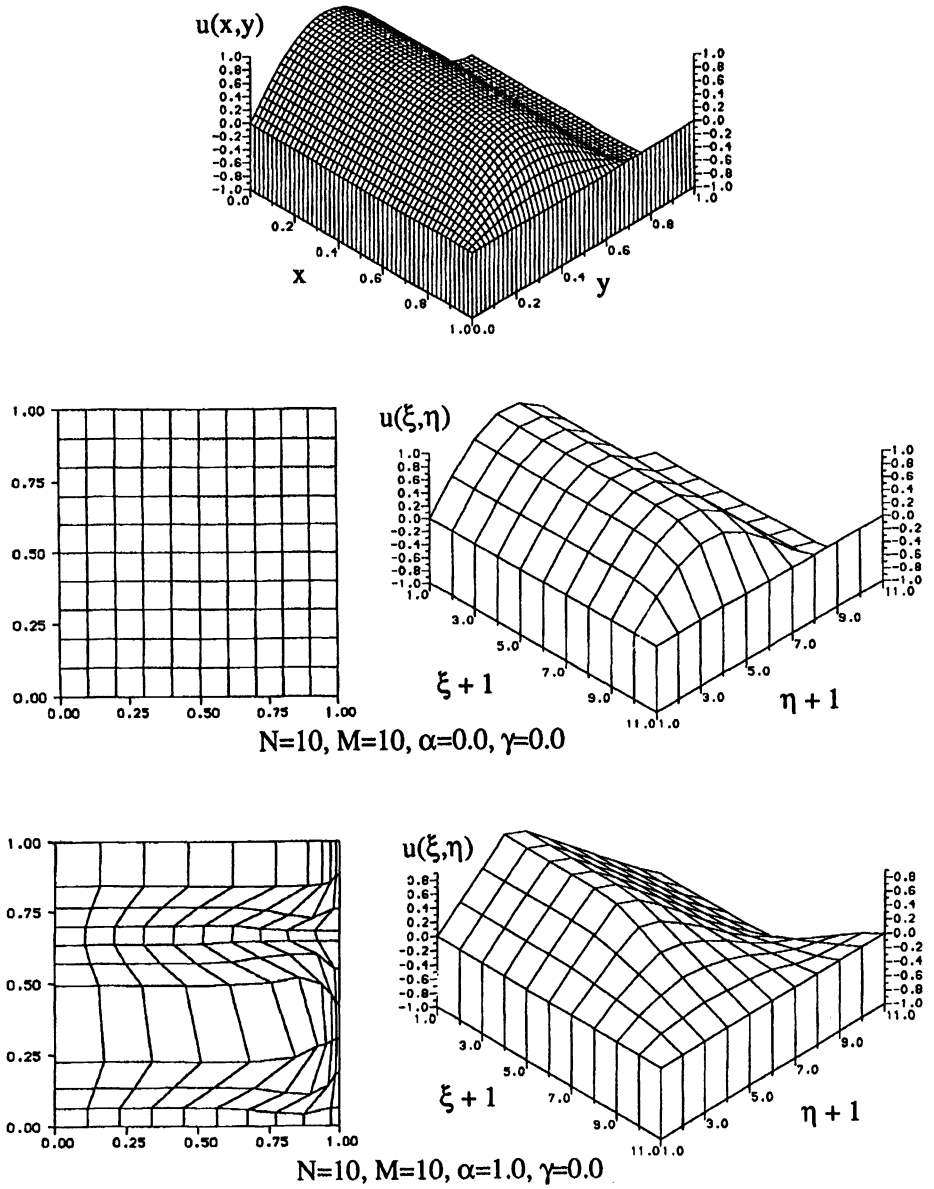


FIG. 3.3. *Solution and meshes for Example* 3.1 *with* $R = 15.0, \omega = 1.5\pi$.

In the solution procedure described below the subscripts i and j take the values $0 \leq i \leq n$ and $0 \leq j \leq m$. The process is initiated with the continuation parameter, α, set to zero and $u_{i,j}$ given by the solution of DCDE on a uniform mesh. α is increased by increments until it reaches the preselected value that is displayed with each set of results. At each value of α, the complete system is solved iteratively for $(x_{i,j}, y_{i,j})$ and $u_{i,j}$, and the superscript it denotes the iteration number. The algorithm may be written as:

(i) Set it:=0. Initial values $(x_{i,j}^{(0)}, y_{i,j}^{(0)})$ are values on a uniform mesh and initial values $u_{i,j}^{(0)}$ are given by solution of DCDE on the uniform mesh.

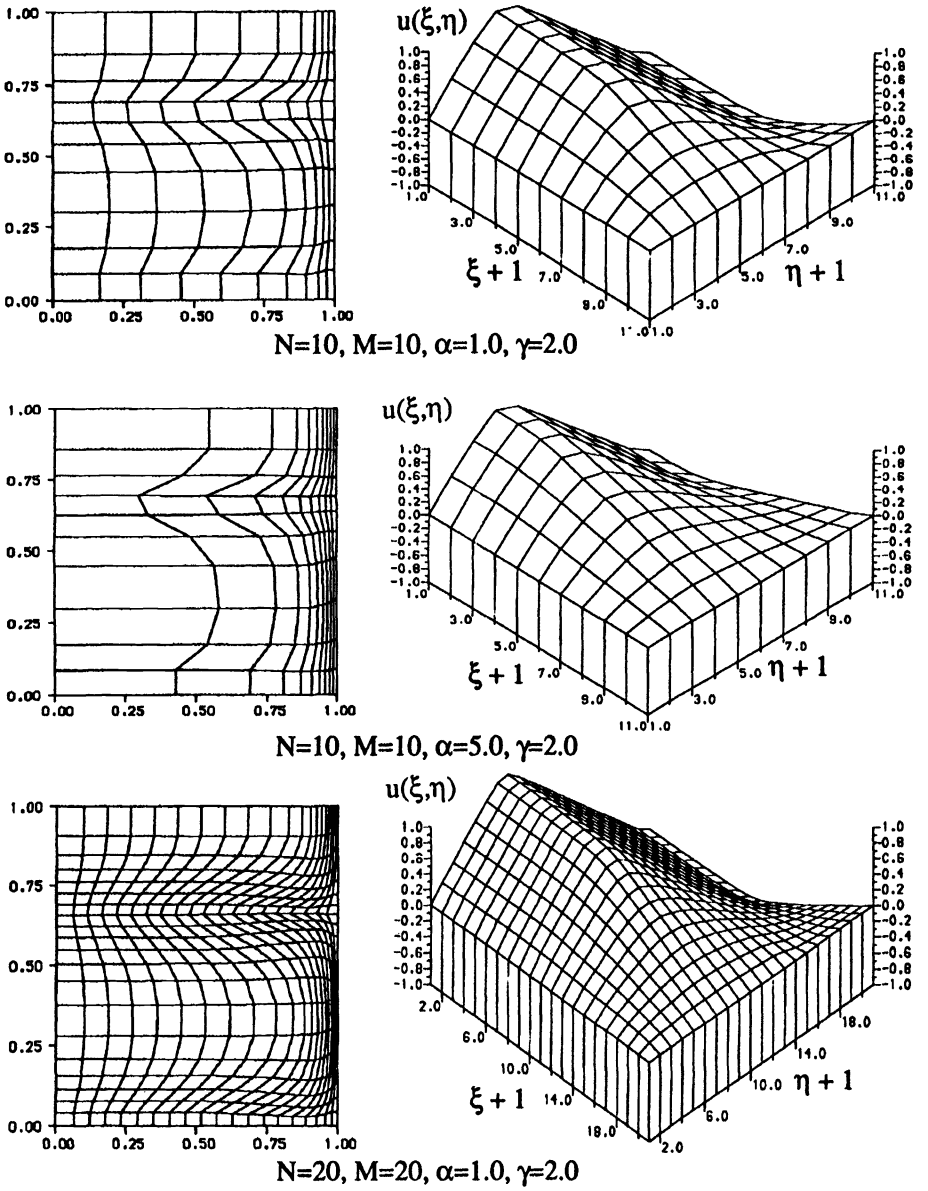


FIG. 3.3. *Continued.*

(ii) Based on current values $u_{i,j}^{(it)}$, solve (26) and (29) by homotopy and a Newton–Raphson method (α serves as the continuation parameter). The previous mesh $(x_{i,j}^{(it)}, y_{i,j}^{(it)})$ is used to initiate the continuation process. The new mesh obtained in this step is denoted by $(x_{i,j}^{(it+1)}, y_{i,j}^{(it+1)})$.

(iii) Solve DCDE on the updated mesh $(x_{i,j}^{(it+1)}, y_{i,j}^{(it+1)})$ to obtain $u_{i,j}^{(it+1)}$.

(iv) Check convergence criterion on $u_{i,j}^{(it+1)}$. If this is not satisfied, then set $it := it + 1$ and return to (ii).

(v) Obtain final mesh by solving (26) and (29).

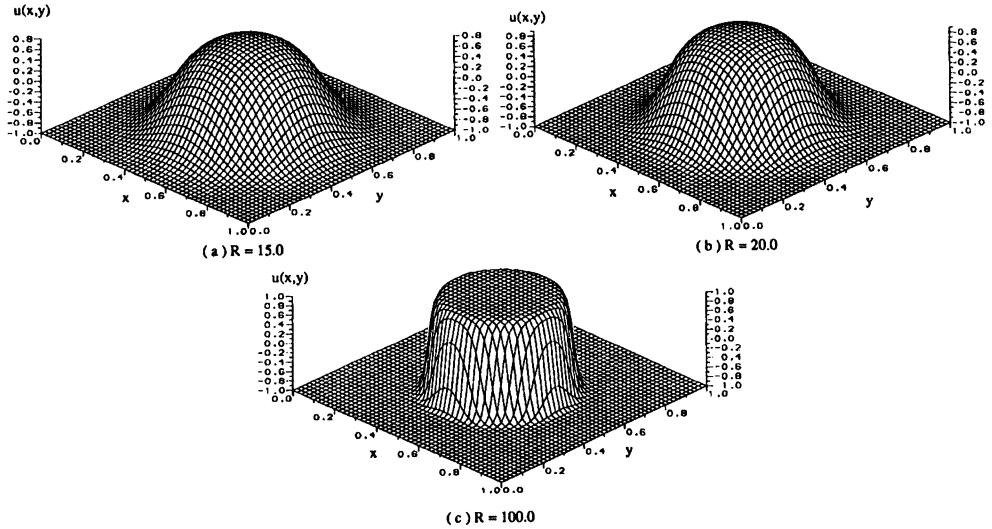


FIG. 3.4. *Exact solutions of Example* 3.2 *for different values of R.*

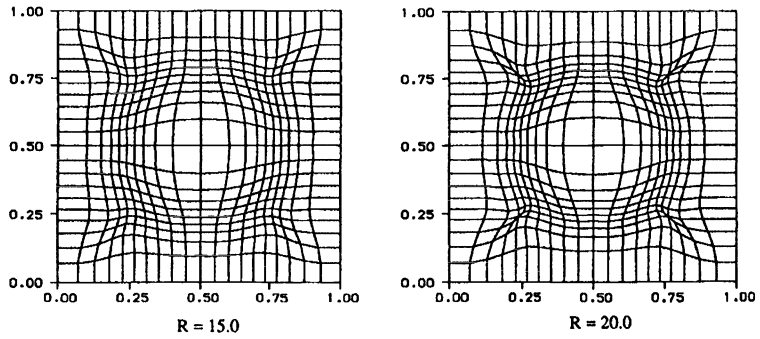


FIG. 3.5. *Adapted meshes obtained using* $\beta = 0.0$ *and* $N = 20$, $M = 20$, $\alpha = 1.0$, $\gamma = 2.0$ *for Example* 3.2.

The convergence criterion is given by

$$\frac{|E_{\max}^{(it+1)} - E_{\max}^{(it)}|}{E_{\max}^{(it)}} < 0.005, \tag{34}$$

where $E_{\max}$ is defined by

$$E_{\max} = \frac{1}{1 - e^{-R}} \max_{i,j} |u_{i,j} - u^e(x_{i,j}, y_{i,j})|. \tag{35}$$

Three cases have been considered in this section. In these cases, the parameter β is set to zero (without scaling).

Case 1. $R = 15$, $\omega = \pi$. The convergence histories, solutions at $y = 0.5$, and adapted meshes for several values of α and γ are shown in Fig. 4.1(a) and (b), respectively. It is observed easily from these figures that more accurate solutions can be obtained and the boundary layer can be resolved more sharply by grid adaption. These figures also show that the accuracy of the computed solution on a smooth mesh is better than that on an unsmooth mesh ($\alpha = 1.0$, $\gamma = 0.0$). By comparing Fig. 4.1(b) with Fig. 3.1, it can be seen that nearly the same mesh is obtained for given analytic function u and for computed solution $u_{i,j}$.

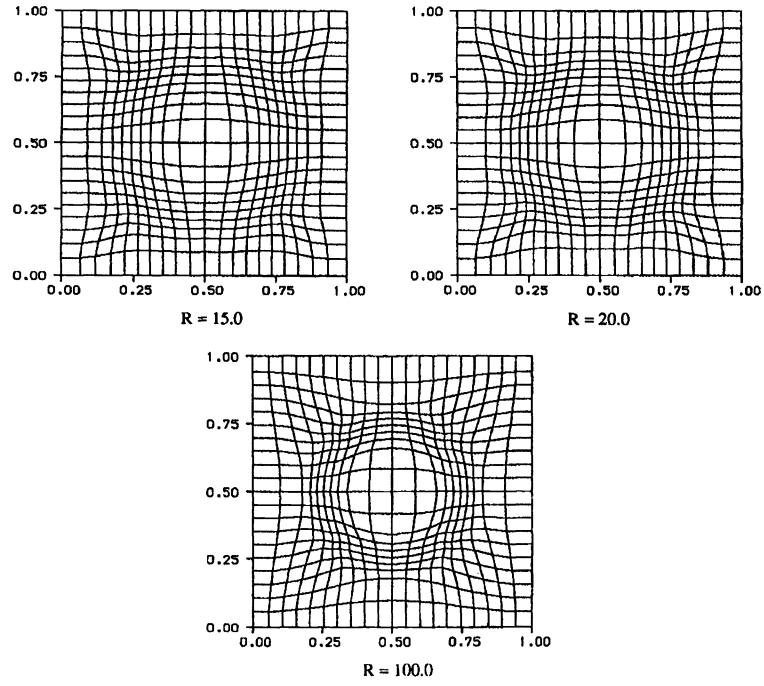


FIG. 3.6. *Adapted meshes obtained using* $\beta = 0.01$ *and* $N = 20$, $M = 20$, $\alpha = 1.0$, $\gamma = 2.0$ *for Example* 3.2.

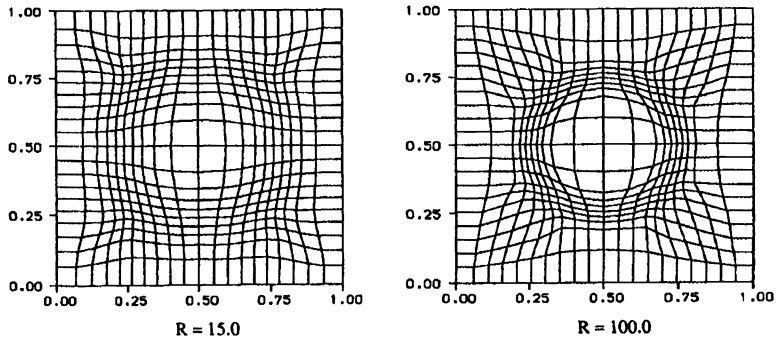


FIG. 3.7. *Adapted meshes obtained using* $\beta = 0.005$ *and* $N = 20$, $M = 20$, $\alpha = 1.0$, $\gamma = 2.0$ *for Example* 3.2.

Case 2. $R = 35$, $\omega = \pi$. In this case, the solution has a sharper boundary layer near $x = 1$. Figure 4.2 shows the convergence history, solutions at $y = 0.5$, and converged adapted mesh for $\alpha = 1.0$ and $\gamma = 2.0$. This figure consolidates the observations made for Case 1.

Case 3. $R = 15$, $\omega = 1.5\pi$. By comparing the graphs of exact solutions (Figs. 3.1–3.3), it is evident that this case is more difficult than the previous two cases because more adaption in the y direction is required. Figure 4.3 shows that although a less accurate solution is obtained for this case than for Cases 1 and 2, a more accurate solution is obtained on an adapted mesh than on a uniform mesh. It is also interesting to note in Fig. 4.3 that the solution obtained on the adapted mesh is less accurate than that obtained on a uniform mesh at points far from the boundary layer ($x \leq 0.7$) along $y = 0.5$, but the adapted mesh still gives a solution with much better resolution of the boundary layer.

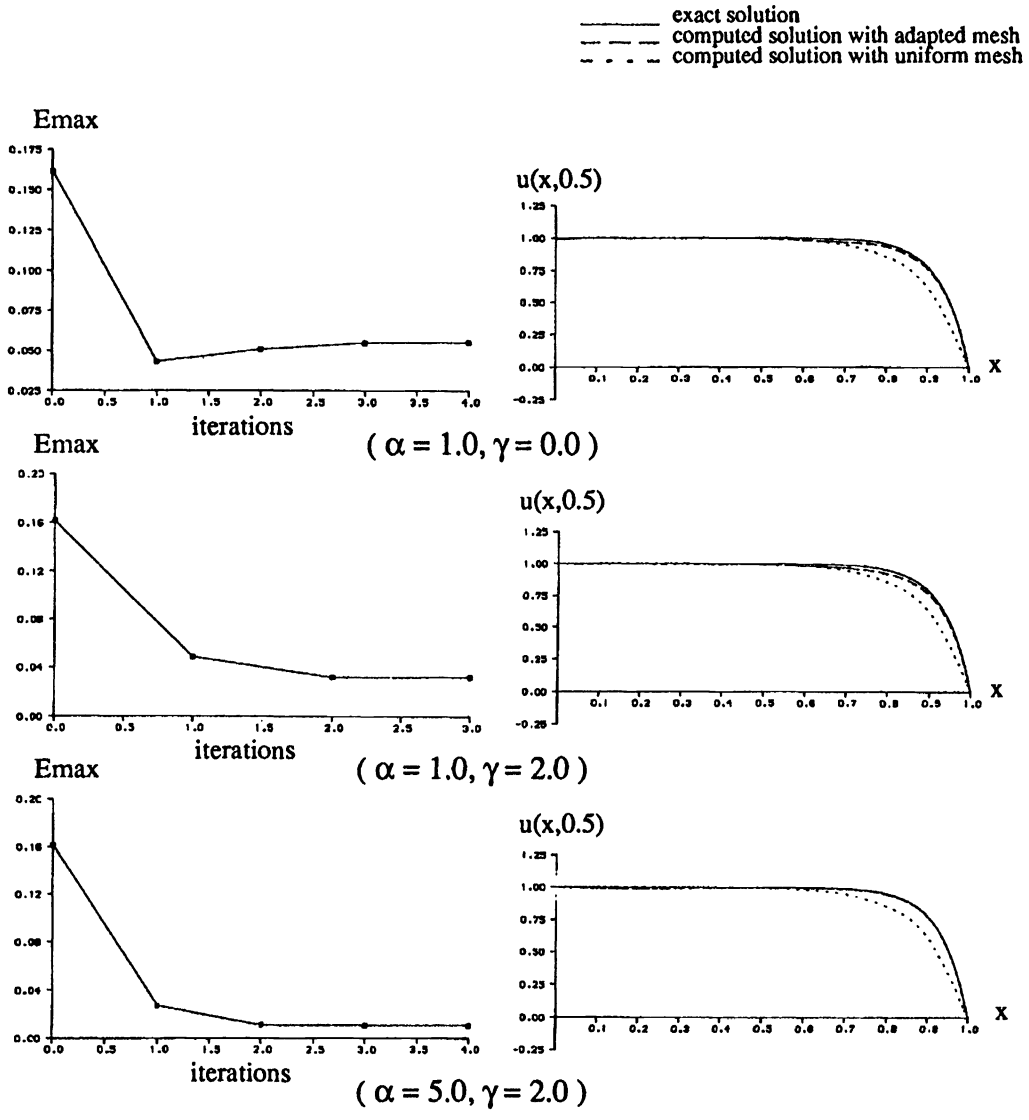


FIG. 4.1(a). *Convergence histories and solutions at* $y = 0.5$ *for Example* 4.1 ($R = 15.0, \omega = \pi$) *with* $N = 10, M = 10$.

Cases 1 and 3 have been considered by Dvinsky [6]. Indeed, our convergence criterion (34) has been chosen as in [6] to enable us to make comparisons with Dvinsky's work in terms of accuracy and efficiency. We notice that the present method gives results that are at least as accurate as those obtained by Dvinsky's method. But more flexibility is available in the present method, and much better results can be obtained by suitable choices of the parameters α, β, and γ. For example, as may be seen in Fig. 4.1, larger values of α can lead to better results.

5. Conclusions and comments. An understanding of the equidistribution principle is described and it is found that the equidistribution principle cannot be satisfied simultaneously on the whole structured mesh. Based on this understanding, a local (coordinate-line-based)

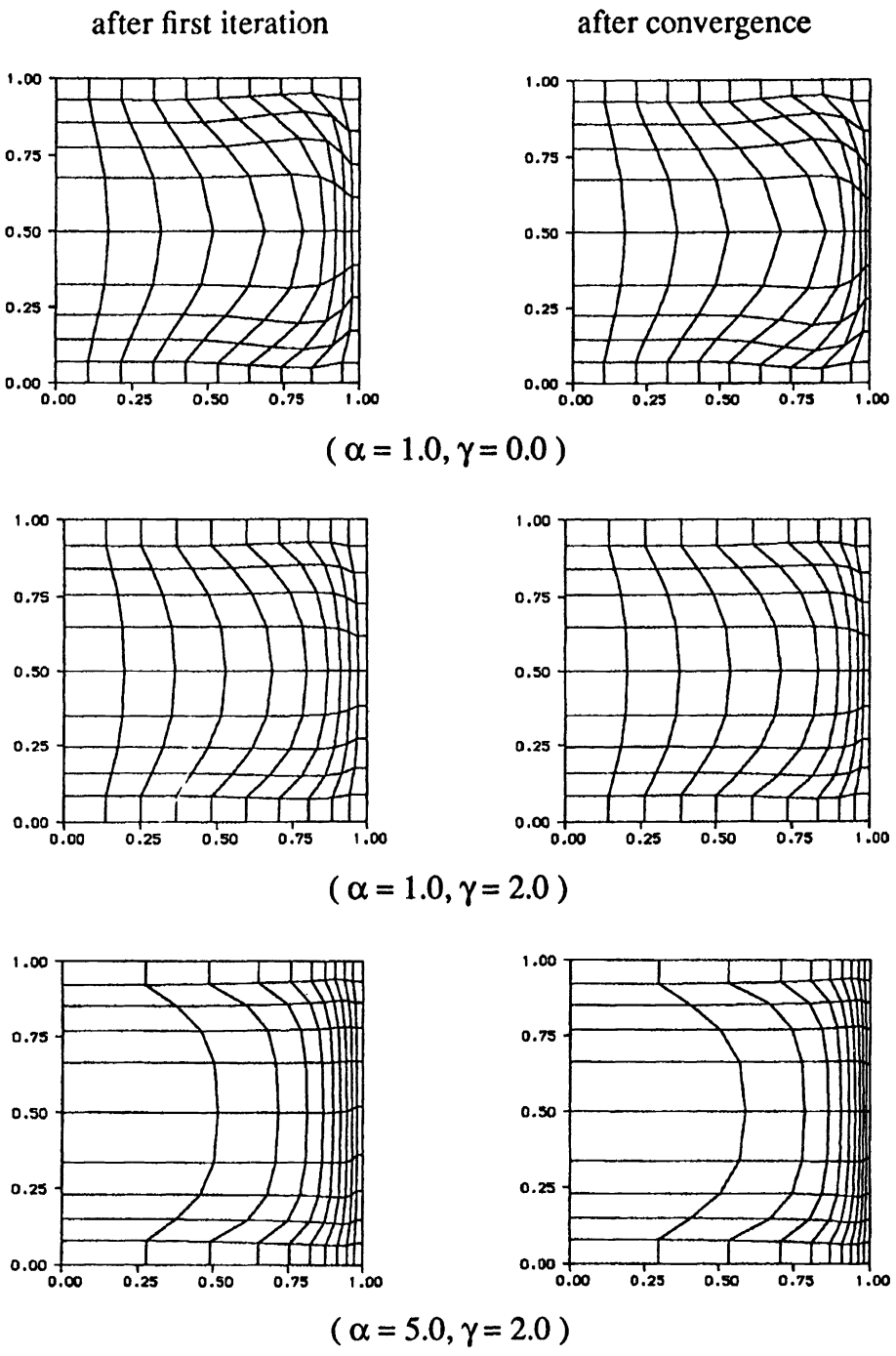


FIG. 4.1(b). *Adapted meshes for Example* 4.1 ($R = 15.0$, $\omega = \pi$) *with* $N = 10$, $M = 10$.

equidistribution principle is developed and a simple adaptive grid method with three parameters α (concentration), β (scaling), and γ (smoothness) is presented.

Numerical experiments have been performed on grid adaption, both for given analytic function and for numerical solutions of partial differential equations in two dimensions. They

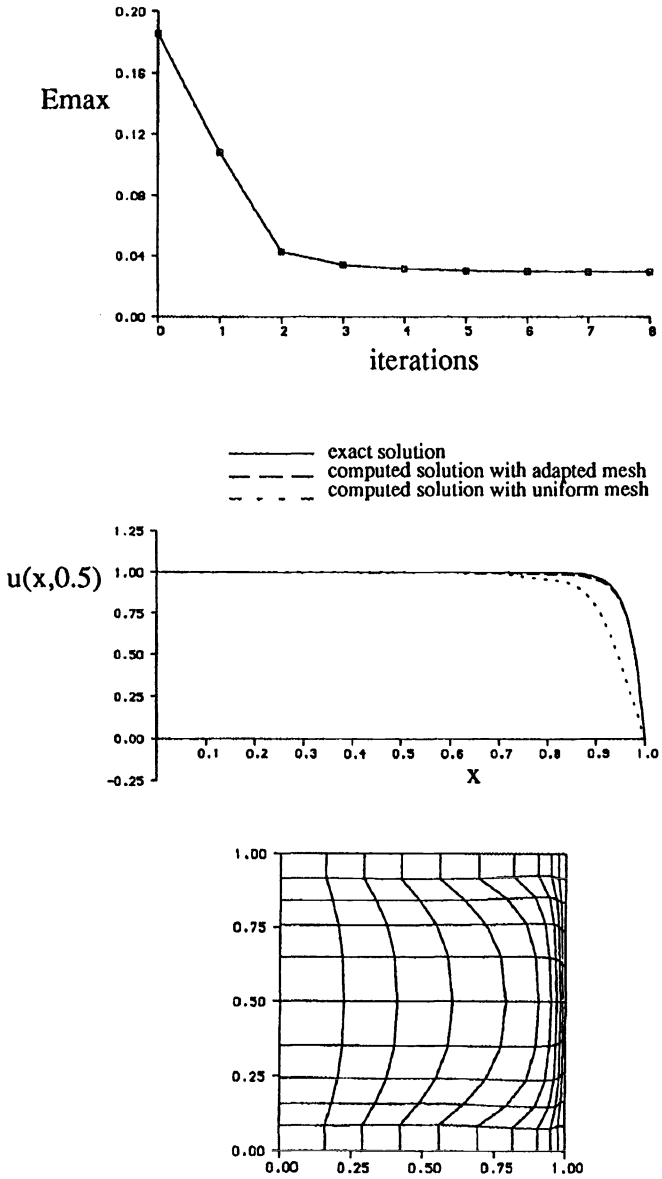


FIG. 4.2. *Convergence history, solutions at* $y = 0.5$ *and adapted mesh for Example* 4.1 ($R = 35.0, \omega = \pi$) *with* $N = 10, M = 10, \alpha = 1.0, \gamma = 2.0$.

show that the present adaptive grid method is fairly insensitive to the choice of values of parameters and it can lead to much better results than those obtained on uniform meshes. The equidistribution principle presented here has only been applied to a simple, steady, two-dimensional convection-diffusion equation. It should be possible, however, to apply the multidimensional adaptive grid method to time-dependent problems. A step in this direction has recently been made by Huang, Ren, and Russell [11], [12] with their moving mesh partial differential equations based on versions of the one-dimensional equidistribution condition (16). The moving mesh equations control the locations of the nodes in such a way that the equidistribution condition remains satisfied as time evolves.

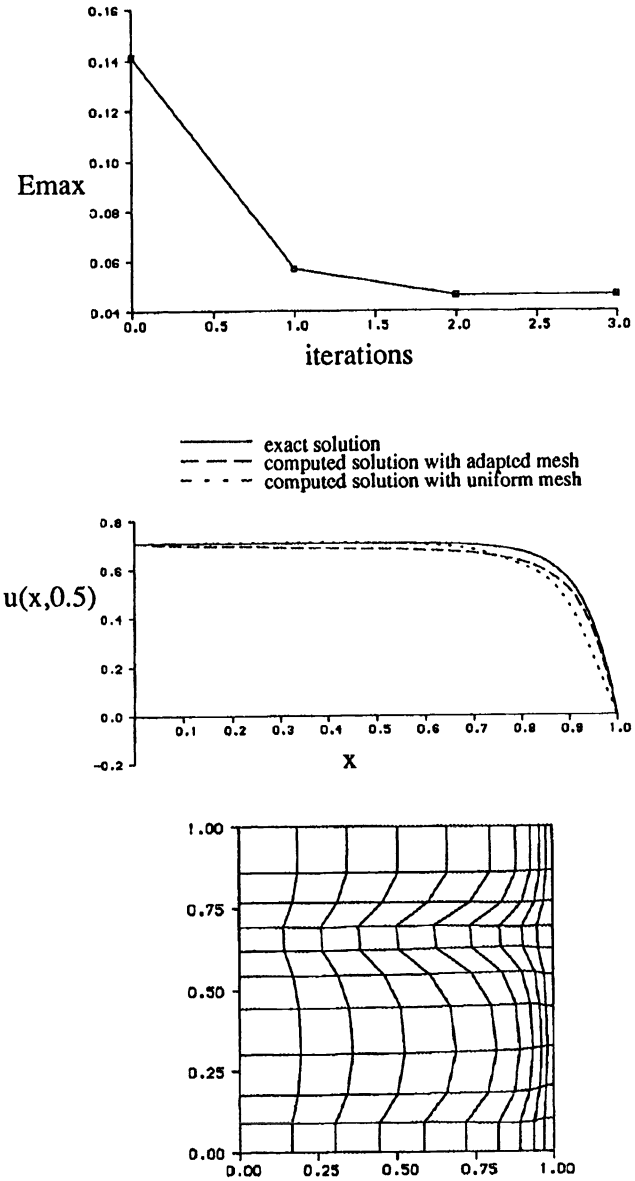


FIG. 4.3. *Convergence history, solutions at* $y = 0.5$ *and adapted mesh for Example* 4.1 ($R = 15.0, \omega = 1.5\pi$) *with* $N = 10, M = 10, \alpha = 1.0, \gamma = 2.0$.

Obviously, this method needs further investigation and it also needs to be applied to more problems. The efficiency of the proposed method relative to fixed-grid methods is also an area which has to be investigated. At this stage we can only say that for problems with steep gradients, adaptive methods are usually better in terms of computational efficiency. The reader is referred to the work by Hawken, Gottlieb, and Hansen [10] for further commentary on this matter. We hope that our presentation of the equidistribution principle in multidimensions may be useful for the development of other new adaptive grid methods.

Acknowledgements. The authors wish to thank the referees for their helpful comments.

REFERENCES

[1] J. U. BRACKBILL AND J. S. SALTZMAN, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982), pp. 342–368.

[2] M. O. BRISTEAU, R. GLOWINSKI, L. DUTTO, J. PÉRIAUX, AND G. ROGÉ, *Compressible viscous flow calculations using compatible finite element approximations*, Internat. J. Numer. Methods Fluids, 11 (1990), pp. 719–749.

[3] D. CATHERALL, *The adaption of structured grids to numerical solutions for transonic flow*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 921–937.

[4] C. DE BOOR, *Good approximation by splines with variable knots* II, in Lecture Notes in Mathematics 363, Springer-Verlag, New York, 1974, pp. 12–20.

[5] E. A. DORFI AND L. O'C. DRURY, *Simple adaptive grids for* 1-D *initial value problems*, J. Comput. Phys., 69 (1987), pp. 175–195.

[6] A. S. DVINSKY, *Adaptive grid generation from harmonic maps on Riemannian Manifolds*, J. Comput. Phys., 95 (1991), pp. 450–476.

[7] H. A. DWYER, R. J. KEE, AND B. R. SANDERS, *Adaptive grid method for problems in fluid mechanics and heat transfer*, AIAA J., 18 (1980), pp. 1205–1212.

[8] H. A. DWYER, *Grid adaption for problems in fluid dynamics*, AIAA J., 22 (1984), pp. 1705–1712.

[9] R. M. FURZLAND, J. G. VERWER, AND P. A. ZEGELING, *A numerical study of three moving-grid methods for* 1-D PDEs *which are based on the method of lines*, J. Comput. Phys., 89 (1990), pp. 349–388.

[10] D. F. HAWKEN, J. J. GOTTLIEB, AND J. S. HANSEN, *Review of some adaptive node-movement techniques in finite element and finite difference solutions of* PDEs, J. Comput. Phys., 95 (1991), pp. 254–302.

[11] W. HUANG, Y. REN, AND R. D. RUSSELL, *Moving mesh* PDEs *based on the equidistribution principle*, SIAM J. Numer. Anal., 31 (1994), pp. 709–730.

[12] ———, *Moving mesh methods based on moving mesh* PDEs, J. Comput. Phys., submitted.

[13] J. PERAIRE, M. VAHDATI, K. MORGAN, AND O. C. ZIENKIEWICZ, *Adaptive remeshing for compressible computations*, J. Comput. Phys., 72 (1987), pp. 449–466.

[14] J. F. THOMPSON, Z. U. A. WARSI, AND C. W. MASTIN, *Numerical Grid Generation*, North-Holland, New York, 1985.

[15] A. WINSLOW, *Numerical solution of the quasilinear Poisson equation in a nonuniform triangular mesh*, J. Comput. Phys., 2 (1967), pp. 149–172.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 798–818, July 1994

003

ERROR EQUIDISTRIBUTION AND MESH ADAPTATION*

KE CHEN†

Abstract. The author reviews a number of strategies on mesh adaptation from a mathematical point of view and compares their performances in solving nonlinear diffusion models from semiconductor process modeling. Some new strategies are then proposed that improve on the existing ones. The analysis of the strategies based on monitoring interpolation errors and local truncation errors leads to a new theory on error equidistribution. Several implementation methods for equidistributing an error monitor function are also reviewed and outstanding problems are highlighted.

Key words. error equidistribution, monitors, nonlinear parabolic PDEs, automatic mesh adaptation

AMS subject classifications. 65M50, 65N50, 35K57

1. Introduction.

Most numerical methods for solving partial differential equations (PDEs) demand a minimum number of discretization points (or subdomains) of the underlying domain so as to expect a required accuracy tolerance. However, this number is generally not available in advance. What one can realistically achieve is to obtain a numerical solution with optimal accuracy for a given number of discretization points. This implies that the numerical solution so obtained should have, to some extent, its error distributed uniformly throughout the domain, which follows from the study of error bounds. Here we investigate various methods of achieving uniform error distributions (i.e., error equidistributions) through appropriate mesh selection procedures.

Our main application is to the solution of parabolic PDEs. We concentrate on one-dimensional (1D) model equations, although extension to higher dimensions is also discussed. In particular three two-dimensional (2D) examples are given.

Research in the area has largely been centered on the numerical solution of boundary value problems (BVPs). Often a good mesh is obtained either by simultaneously finding the mesh points along with the numerical solution or by redistributing the mesh points through error equidistribution after an initial approximation has been obtained. See [4], [10], [36], [38], and [44] and the references therein.

When these mesh selection ideas are applied to solve time-dependent PDEs, the former category of methods becomes the so-called moving mesh techniques (see [36]); while the latter category aims to select a mesh for the new time level based on the mesh redistribution at the present time level.

The idea of using moving meshes is attractive and many successful applications have been reported. Refer to [36] and many useful references listed therein that include [17], [23], [29], [35], and [45]. More work can be found in [6] and [21]. However, the combination of a mesh selection equation with the PDEs as a system of PDEs often leads to a more 'stiff' problem. A good method of implementing the idea is still to be found.

We pursue the second category of methods with error equidistribution, which is sometimes called "static regridding." There are two common cases: the total number of grid points is either not restricted or fixed. The former case corresponds to the so-called local refinement methods that have been extensively studied in [7], [33], and [43] among others. These methods tend to require complex data structures for implementation. Here we shall restrict our work to the latter case of distributing a fixed number of nodes.

*Received by the editors December 16, 1991; accepted for publication (in revised form) June 16, 1993. This work was supported by the Department of Trade and Industry through the Science and Engineering Research Council.

†Department of Statistics and Computational Mathematics, The University of Liverpool, Liverpool L69 3BX, England (chen@scmp.scm.liv.ac.uk).

Equidistribution of some known quantity of the solution to select a suitable mesh usually corresponds to the use of a transformation of the original equation, so that the transformed equation has equal error distribution on the uniform mesh with the same number of nodal points. While the choice of such a quantity generally appears to be somewhat arbitrary and hard to identify (see [41]), we note that many known techniques do equidistribute some error measurement of the underlying solution. In theory, the best numerical strategy for an operator equation should be that of [34] by equidistributing the local truncation error of the method. This however has been proved to be false in practice. See [38], [41], and §6 here.

This paper attempts to do the following:

- Review some of the known methods for mesh selection.
- Show their connections and propose some new variants.
- Review some commonly used techniques for implementation.
- Test the effectiveness and robustness of these mesh selection strategies on a nonlinear diffusion model.
- Introduce a new class of robust methods.
- Present some test results on a few 2D examples.

Before proceeding, let us introduce some notation and formal definitions to be used later. We denote by

$$\mathcal{N}(u) = 0, \qquad x \in [a, b], \tag{1}$$

a nonlinear operator equation governing a BVP, satisfying some suitable boundary conditions and having a unique solution. Define a mesh of $(N+1)$ points by

$$\Pi : a = x_0 < x_1 < \cdots < x_N = b. \tag{2}$$

It is the purpose of this paper to address the problem of how to suitably select such a mesh determined by a given number N. Denote by S_Π^k the space of those functions that are piecewise kth degree polynomials in each interval $I_j = [x_{j-1}, x_j]$ for $j = 1, \ldots, N$. Let $h = \max\{h_j\}$. We shall seek the solution of (1) in space S_Π^k. Such an approximation, denoted by u_h, satisfies the discretized equation

$$\mathcal{N}_\Pi(u_h) = 0, \tag{3}$$

which follows from the application of a finite difference or finite element method. In §6, an example using the finite element method is presented.

Assuming that the numerical method adopted is stable, we have for h sufficiently small

$$\|u - u_h\|_p \le c\|\mathcal{N}_\Pi(u) - \mathcal{N}_\Pi(u_h)\|_p, \tag{4}$$

where the L_p norm also includes the case of $p = \infty$ and c is a generic constant independent of the mesh Π and N.

We now review some theory of mesh selection with regard to error minimization. As already discussed, each mesh selection method will be associated with a positive weight function $M(u, x)$ such that the mesh Π generated for variable x corresponds to the uniform mesh for the new variable ξ

$$\xi = \xi(x) = \int_a^x \rho(u, y)dy, \tag{5}$$

where

$$\rho(u, x) = M(u, x) \Big/ \int_a^b M(u, y)dy = \xi'(x),$$

that is,

$$\xi(x_j) - \xi(x_{j-1}) = c_1 \equiv \frac{1}{N}, \qquad j = 1, \ldots, N \tag{6}$$

or

$$\int_{x_{j-1}}^{x_j} \rho(u, y)dy = c_1. \tag{7}$$

We call this weight function M (often representing some error measurement of the solution) a *monitor* function, ρ the *density* function, and ξ the *grading* function (or a transformation) for the mesh selection strategy. We call the mesh Π *equidistributing* when (6) holds and *subequidistributing* when (6) holds with $\leq$, with respect to M and c_1. For similar definitions, see [26], [38], and [44].

2. Optimal monitors for operator equations. For the operator equation (1), we shall discuss the problem of selecting the mesh Π with a given fixed N which gives the optimal solution. For this purpose we shall try to sharpen, by mesh selection, the error bound (4), i.e., to

$$\|u - u_h\|_p \leq c\|\mathcal{N}_\Pi(u)\|_p. \tag{8}$$

Suppose that the local truncation error (LTE) has the following form (when $u_h \in S_\Pi^k$)

$$\mathcal{N}_\Pi(u)_i = h_i^{k+1}T(x_i) + O(h_i^{k+2}), \qquad i = 1, \ldots, N, \tag{9}$$

where $T(x)$ is known for a numerical method, independent of Π, and given usually by some function of derivatives of u. Furthermore, [34] has shown that the best choice of monitor functions is, for the L_∞ norm,

$$M_\infty = g^{1/(k+1)}, \qquad g = \max(\|T\|_\infty, \varepsilon) \tag{10}$$

and, for the L_p norms,

$$M_p = g^{p/[p(k+1)+1]}, \qquad g = \max(\|T\|_p, \varepsilon), \tag{11}$$

where ε is some small constant. Such a choice ensures that the global error is given by

$$\|u - u_h\| \leq c_2 N^{-k-1} + O(N^{-(k+2)}), \tag{12}$$

where the constant c_2 is minimal over all admissible choices of the mesh Π (in either norm).

3. Optimal monitors for interpolating polynomials. Although our underlying solution to be solved numerically is unknown, we now investigate the theoretical problem of finding the optimal monitor for polynomial interpolations to have the minimal interpolation error. Such a problem was first solved in [10].

3.1. H^m-seminorm. We again seek a polynomial approximation to u in the space of S_Π^k. We are only interested in those piecewise polynomials that interpolate the function u at one fixed node τ_i in each interval I_i for $i = 1, \ldots, N$, although any $v \in S_\Pi^k$ is a kth degree polynomial. This setting is a slight generalization of that of [10], where only interpolations at nodal points are assumed, but can be easily validated. Define the H^m-seminorm of a function v by

$$|v|_m^2 = \int_a^b [v^{(m)}]^2 dx. \tag{13}$$

We now seek the optimal monitor M (satisfying (7)) that minimizes $e = u - v_h$ for $v_h \in S_\Pi^k$ in the H^m-seminorm. The fundamental inequality given in [10] states the following:

$$|e|_m^2 \le \sum_{i=1}^{N} \left(\frac{h_i}{\pi}\right)^{2(j+1-m)} \int_{I_i} [u^{(j+1)}]^2 dx, \tag{14}$$

where $j \ge m$. Furthermore, for an arbitrary monitor satisfying (7), by taking $j = k$, it is shown that

$$|e|_m^2 \le \frac{1}{(\pi N)^{2(k+1-m)}} \int_a^b \frac{[u^{(k+1)}]^2}{[\rho]^{2(k+1-m)}} dx(1 + O(h)), \tag{15}$$

where $\rho = \xi'$. Minimization of the right-hand side of (15) over ξ leads to the solution of a Euler equation, whose solution produces the optimal monitor

$$M = [u^{(k+1)}]^{2/[2(k+1-m)+1]}. \tag{16}$$

3.2. Sobolev seminorm. Generalizations of the work in [10] may be obtained by the use of new error norms. Here we attempt to find the optimal monitor in the Sobolev (l_1, l_2)-seminorm defined here as

$$\|e\|_{l_1,l_2}^2 = \sum_{r=l_1}^{l_2} |e|_r^2 = \int_a^b \sum_{r=l_1}^{l_2} [e^{(r)}]^2 dx, \tag{17}$$

where $0 \le l_1 \le l_2 \le k$. Obviously, the previous H^m seminorm corresponds to the (m, m)-seminorm.

THEOREM 3.1. *Assume that $v_h \in S_\Pi^k$ interpolates u at the point τ_i of each interval I_i $(i = 1, \ldots, N)$. Then the optimal mesh for minimization of $e = u - v_h$ in the Sobolev seminorm* (17) *is given by the monitor function*

$$M = \left\{ \sum_{r=l_1}^{l_2} \left[u^{(k+r+1-l_1)}\right]^2 \right\}^{1/[2(k+1-l_1)+1]}. \tag{18}$$

Proof. Following §3.1, by taking $j = k + r - l_1$ for $r = l_1, \ldots, l_2$, we obtain

$$|e|_r^2 \le \sum_{i=1}^{N} \left(\frac{h_i}{\pi}\right)^{2(k+1-l_1)} \int_{I_i} [u^{(k+r+1-l_1)}]^2 dx. \tag{19}$$

Furthermore, in the similar way (15) was derived, we can verify the following for $r = l_1, \ldots, l_2$

$$|e|_r^2 \le \frac{1}{(\pi N)^{2(k+1-l_1)}} \int_a^b \frac{[u^{(k+r+1-l_1)}]^2}{[\rho]^{2(k+1-l_1)}} dx(1 + O(h)),$$

that is,

$$\|e\|_{l_1,l_2} \le \frac{1}{(\pi N)^{2(k+1-l_1)}} \int_a^b \frac{\sum_{r=l_1}^{l_2} \left\{[u^{(k+r+1-l_1)}]^2\right\}}{[\rho]^{2(k+1-l_1)}} dx(1 + O(h)), \tag{20}$$

where $\rho = \xi'$. We then can solve the Euler equation that minimizes the right-hand side of (20) with respect to ξ, giving the optimal monitor (18). Thus the theorem is proved. □

For instance, the choice of $l_1 = 0$ and $l_2 = 2$ (for the quadratic case $k = 2$) gives the optimal monitor

$$M = \left[u'''^2 + u''''^2 + u'''''^2\right]^{1/7}, \tag{21}$$

while the choice $l_1 = l_2 = 0$ (for the $k = 2$ case) gives the same monitor as that from using the H^0-seminorm, i.e., the L_2 norm,

$$M = \left[u'''\right]^{2/7}. \tag{22}$$

4. Equidistribution based on adaptive integration. The previous two sections introduced optimal monitors explicitly involving derivatives of the unknown solution function u. These are theoretical results. In practical applications (see §6), however, we usually replace u by extrapolation or recovery of the available approximate solution u_h. One natural question to be asked here is the following: Can we derive computable monitors involving the nodal values of u only? The answer is yes and there are several approaches. Refer to [16], [37], and [46] for methods using the residual for refinement.

Here we develop a simple approach based on adaptive integration. For the mesh Π given by (2), we consider the problem of approximating the integral $\int_a^b u dx$ using interpolating polynomials from the space S_Π^k. The error is measured by

$$E_i = \int_{x_{i-1}}^{x_i} (u - u_h) dx, \tag{23}$$

where $u_h \in S_\Pi^k$ is the approximation polynomial. For a given N, we wish to minimize $\max(E_i)$ over all admissible meshes. In the spirit of §§1–3, we propose the following equidistribution criterion for selecting Π

$$E_i \le c_3 \quad \text{for } i = 1, \ldots, N. \tag{24}$$

Using the Taylor expansion, it can be shown that E_i is determined by some high derivatives of u. At this point, instead of attempting to compute these derivatives, we use the extrapolation technique to estimate such quantities as we do in the context of adaptive integration; see [24, §6.3]. It turns out that generally

$$\int_{x_{i-1}}^{x_i} u_h dx - \left(\int_{x_{i-1}}^{x_{mid}} u_h dx + \int_{x_{mid}}^{x_i} u_h dx \right)$$

provides a good estimate for E_i, where x_{mid} denotes the midpoint of I_i. Therefore, given an appropriate constant c_3, the criterion (24) is ready to be applied.

For simplicity and practical use, we give the example of using the trapezoidal rule; high-order methods may be similarly discussed. Define

$$I(u) = \int_{x_{i-1}}^{x_i} u(x) dx, \quad I_1(u) = \int_{x_{i-1}}^{x_i} u_h(x) dx, \quad I_2(u) = \left(\int_{x_{i-1}}^{x_{mid}} + \int_{x_{mid}}^{x_i} \right) \bar{u}_h(x) dx,$$

where $\bar{u}_h$ denotes the piecewise linear approximation in $[x_{i-1}, x_{mid}]$ and $[x_{mid}, x_i]$. As is well known, the integration error is given by

$$I(u) - I_1(u) = -\frac{h_i^3}{12} u''(\eta) \quad \text{for some } x_{i-1} < \eta < x_i.$$

Furthermore, it can be shown that ([24, §6.3])

$$I(u) - I_2(u) \approx \frac{1}{4}[I(u) - I_1(u)] \approx \frac{1}{3}[I_2(u) - I_1(u)],$$

where $I_2 - I_1$ is computable. Therefore, we can define the equidistribution error in (24) as

$$E_i = |I_2(u) - I_1(u)| = h_i \frac{|2u_{mid} - u_i - u_{i-1}|}{4}. \tag{25}$$

In identifying (25) with (7) for monitor functions, we see that the error monitor from using the adaptive trapezoidal integration is

$$M = \left| \frac{2u_{mid} - u_i - u_{i-1}}{4} \right| \approx \frac{h_i^2}{4} \left| \frac{\partial^2 u}{\partial x^2} \right|. \tag{26}$$

5. The use of arc-length norm. Roughly speaking, the use of the arc-length norm puts an emphasis on the error control near rapid variations of the solution function, or, in other words, allows one to have more accuracy near the nonsmooth part of the solution.

Recall that the usual L_p norm for the error function e is defined by

$$\|e\|_p^p = \int_a^b |e|^p dx.$$

The modified norm using arc-length s is proposed as follows:

$$\|e\|_{ap}^p = \int_a^b |e|^p ds = \int_a^b |e|^p \sqrt{1 + u'^2} dx. \tag{27}$$

With such a modified norm, the definitions for the grading function (5) and the monitor function (7) become, respectively,

$$\xi = \xi(x) = \int_a^x M(u, y)\sqrt{1 + u'^2} dy \Big/ \int_a^b M(u, y)\sqrt{1 + u'^2} dy \tag{28}$$

and

$$\int_{x_{j-1}}^{x_j} M(u, y)\sqrt{1 + u'^2} dy = c_4 \equiv \int_a^b M(u, y)\sqrt{1 + u'^2} dy \Big/ N. \tag{29}$$

This automatically brings up a large collection of new monitors to be considered. Examples are illustrated in the next section.

We remark that (i) the arc-length monitor $\tilde{M} = \sqrt{1 + u'^2}$, as first advocated in [44] and used in [45], corresponds to the monitor

$$M = 1 \tag{30}$$

with our new modified arc-length norm. The monitor (30) alone defines a uniform mesh Π for variable x. (ii) The arc-length norm in the case of $p = 2$ (L_2-norm) was previously used in [29] and [30] in the so-called gradient-weighted moving finite element method (GWMFE). There the variational minimization was done for the L_2 norm of the *normal component* of the residual. The use of the arc-length norm helps to de-emphasize the steep portion of the solution so that the problem of excessive clustering of moving finite element nodes near steep fronts can be alleviated. See also [9] for the 2D generalization.

6. Numerical experiments. The test example taken here is the 1D nonlinear dopant diffusion equation (as tested in [14])

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial x}\left(D(e^f)\frac{\partial f}{\partial x}\right) + D(e^f)\left(\frac{\partial f}{\partial x}\right)^2 \quad \text{for } x \in [0, 1], \tag{31}$$

which is the equation transformed from its standard form (as studied in [25])

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial x}\left(D(C)\frac{\partial C}{\partial x}\right). \tag{32}$$

The diffusion coefficient and initial/boundary conditions are given, respectively, by

$$D(C) = \frac{1}{1+\beta}\left[1 + \frac{\beta}{2}\left(C + \sqrt{C^2+4}\right)\right],$$

$$C(x, 0) = H\exp\left[-S(x - R)^2\right],$$

$$\frac{\partial C}{\partial x} = 0 \quad \text{at } x = 0, \text{ and } 1$$

(see [25] for details). We solve (31) by a Petrov–Galerkin finite element method using quadratic interpolating B-splines ($k = 2$). The approximate local truncation error in space has been shown to be determined by (refer to (9))

$$T = -\frac{D}{4}\frac{\partial^4 f}{\partial x^4}$$

(see [14] for details).

We have selected the following three test problems of increasing difficulty:

1. H=100, S=100, R=0.25,
2. H=110, S=121, R=0.15,
3. H=500, S=100, R=0.25,

where β =100 is assumed. We set the diffusion time to be $\bar{t}$=20,000 seconds (corresponding to $t = 10^{-3}$) for the first two test problems and $\bar{t}$=4,000 seconds (or $t = 10^{-4}$) for the third test problem. Note that for a typical arsenic diffusion with intrinsic arsenic concentration of $n_i = 5.0 \times 10^{18}$ cm^{-3}, the above specifications correspond, respectively, to arsenic implantation doses of

1. $N = 6.25 \times 10^{19}$ cm^{-2},
2. $N = 6.2665 \times 10^{19}$ cm^{-2},
3. $N = 3.125 \times 10^{20}$ cm^{-2}.

Considering that $k = 2$ in our case with quadratic polynomials, we test the following monitors

$M_1 = [u'''']^{2/7}$ (L_2-norm of §3.1),

$M_2 = [u'''']^{2/5}$ (H^1-seminorm of §3.1),

$M_3 = [u'''']^{2/3}$ (H^2-seminorm of §3.1),

$M_4 = g^{1/3}$ (L_∞-norm of §2),

$$M_5 = g^{1/4} \qquad (L_1\text{-norm of §2}),$$
$$M_6 = g^{2/7} \qquad (L_2\text{-norm of §2}),$$
$$M_7 = [u'''^2 + u''''^2 + u'''''^2]^{1/7} \qquad ((0, 2)\text{-seminorm of §3.2}),$$
$$M_8 = [u'''^2 + u''''^2]^{1/7} \qquad ((0, 1)\text{-seminorm of §3.2}).$$

Incorporating the modified norm using arc-length (§5), we also test the monitors

$$M_{i+10} = M_i(1 + u'^2)^{1/2} \quad \text{for } i = 1, \ldots, 8.$$

In addition, we may wish to compare with those monitors that are only optimal for lower-order polynomials ($k = 0$ and 1)

$$M_{21} = [u'']^{2/5} \qquad (L_2\text{-norm and } k = 1 \text{ of §3.1}),$$
$$M_{22} = [u'']^{2/3} \qquad (H^1\text{-seminorm and } k = 1 \text{ of §3.1}),$$
$$M_{23} = [u']^{2/3} \qquad (L_2\text{-norm and } k = 0 \text{ of §3.1}),$$
$$M_{24} = [u''^2 + u'''^2]^{1/5} \qquad ((0, 1)\text{-seminorm and } k = 1 \text{ of §3.2}),$$

and their counterparts in the modified norm using arc-length (§5)

$$M_{i+5} = M_i(1 + u'^2)^{1/2} \quad \text{for } i = 21, \ldots, 24.$$

The idea presented in §4 offers a way of constructing a monitor function for the case of $k = 2$. But this monitor would be expensive to implement as more extrapolations or recoveries are required. Instead for testing we only use the monitor introduced for $k = 1$

$$M_{31} = \tfrac{h_i^2}{4}|u''| \qquad (\text{trapezoidal monitor of §4}),$$
$$M_{32} = M_{31}(1 + u'^2)^{1/2} \qquad (\text{arc-length norm of } M_{31}).$$

Finally, it may be argued that the optimal monitor is only for a particular polynomial space and it would be more robust to have monitors that are optimal for all low-order polynomial approximations. This is generally difficult to realize. One way to obtain suboptimal monitors is to use the averaging of known monitors. To be more specific, we take the following for tests:

$$M_{33} = [u'^{2/3} + u''^{2/5} + u'''^{2/7}]/3 \quad (L_2\text{-norm and average of } M_1, M_{21}, M_{23}),$$
$$M_{34} = M_{33}(1 + u'^2)^{1/2} \qquad (\text{arc-length norm of } M_{33}).$$

Once a monitor function has been chosen, there are several existing approaches that one can adopt to equidistribute the error. Since the exact equidistribution is difficult to realize, one usually looks for ways to subequidistribute an error monitor. Among the useful methods are those found in [10] and [44], which use Newton iterations, and those found in [26] and [39], which approximate the integrals numerically. Here we prefer the simple approach of removal and insertion introduced in [22] and [32]; see [12]. With this method, we remove points where the error is relatively small and insert points where the error is relatively large, while maintaining the total number of points.

In our tests, we take the number of intervals N=64 for the first two problems and N=200 for the third problem. We show the experimental results in Tables 1–3, where we denote by "Steps" the number of timesteps taken (using the same time stepping criterion), "CPU"

TABLE 1
Results for Test Problem 1.

Monitor	Steps	CPU	Updates	Error(RMS)	Comment
1	116	36	43	0.188	
2	116	37	43	0.222	
3	112	39	39	0.166	N+18
4	*	*	*	*	Diverge
5	196	61	69	0.169	
6	224	96	197	0.184	
7	145	39	41	0.202	
8	191	49	15	0.097	
11	116	37	43	0.229	
12	116	37	43	0.230	
13	113	45	40	0.145	N+2
14	64	37	48	0.389	
15	206	60	94	0.166	
16	63	23	61	0.358	
17	213	54	4	0.132	
18	195	51	33	0.089	
21	284	87	127	0.099	
22	328	98	129	0.062	
23	202	59	15	0.092	
24	284	86	127	0.099	
26	130	42	71	0.146	
27	267	88	143	0.039	
28	344	94	18	0.054	
29	130	46	71	0.146	
31	169	51	29	0.127	
32	321	91	62	0.080	
33	301	87	0	0.046	
34	246	74	11	0.079	

the CPU seconds used on a Sun 4, "Updates" the number of mesh adaptations that actually happened in the time history and "Error" the root mean square (rms) norm of the absolute error against an accurate solution at 101 evenly spaced points in [0, 1]. With the equidistribution algorithm adopted, some test runs may demand the increase of the number of points "N" in which case we denote in the "Comment" column the real number of points used. From the results in Tables 1–3, we can observe the following:

1. For Problem 1, only monitor M_4 (via the L_∞-norm of the LTE) failed to obtain a reasonable solution. The failure from using this monitor was previously reported in [44].

2. For the more difficult Problems 2 and 3, more monitors failed to solve the equations. In particular, note that these include M_1 and M_4.

3. The following monitors were successful for all three test problems of which the first two monitors are justified theoretically for quadratic approximations, (see §§2, 3.2, and 5).

(a) $M_5 = g^{1/4}$ (L_1-norm of §2),

(b) $M_{18} = [u''^2 + u''''^2]^{1/7}(1 + u'^2)^{1/2}$ ((0, 1)-seminorm of §3.2),

(c) $M_{21} = [u'']^{2/5}$ (L_2-norm and $k = 1$ of §3.1),

(d) $M_{22} = [u'']^{2/3}$ (H^1-seminorm and $k = 1$ of §3.1),

(e) $M_{23} = [u']^{2/3}$ (L_2-norm and $k = 0$ of §3.1),

(f) $M_{24} = [u''^2 + u''''^2]^{1/5}$ ((0, 1)-seminorm and $k = 1$ of §3.2),

TABLE 2
Results for Test Problem 2.

Monitor	Steps	CPU	Updates	Error(RMS)	Comment
1	104	33	15	0.415	
2	104	28	15	0.415	
3	104	28	15	0.415	
4	267	68	86	0.225	
5	500	117	108	0.104	
6	295	110	267	0.206	
7	*	*	*	*	Diverge
8	325	109	9	0.022	
11	104	28	15	0.415	
12	104	33	15	0.415	
13	104	30	15	0.415	
14	*	*	*	*	Diverge
15	479	177	134	0.160	
16	289	119	272	0.239	
17	*	*	*	*	Diverge
18	380	126	26	0.036	
21	500	132	241	0.068	
22	500	133	155	0.092	
23	405	98	18	0.167	
24	500	134	241	0.068	
26	500	122	111	0.094	
27	500	124	160	0.075	
28	312	75	18	0.051	
29	500	128	111	0.094	
31	279	73	39	0.131	
32	500	122	61	0.115	
33	*	*	0	*	Diverge
34	326	86	7	0.077	

(g) $M_{27} = [u'']^{2/3}(1 + u'^2)^{1/2}$ (H^1-seminorm and $k = 1$ of §3.1),

(h) $M_{28} = [u']^{2/3}(1 + u'^2)^{1/2}$ (L_2-norm and $k = 0$ of §3.1),

(i) $M_{32} = \frac{h_i^2}{4}|u''|(1 + u'^2)^{1/2}$ (trapezoidal monitor of §4, in arc-length norm),

(j) $M_{34} = \dfrac{u'^{2/3} + u''^{2/5} + u'''^{2/7}}{3(1 + u'^2)^{-1/2}}$ (average of M_1, M_{21}, M_{23}, in arc-length norm).

The final comparison is made in §8, where more monitors are tested.

7. A new theory on error equidistribution. We have introduced two main types of error equidistribution methods, those based on errors of interpolation polynomials approximating a "known" function (§§3 and 4) and those based on numerical solution errors of operator equations (§2). In the context of solving an operator equation such as (1), the underlying function u is obviously unknown and only given implicitly. It is natural to consider the equidistribution of the solution error of some kind for the particular numerical method used. Unfortunately, the use of local truncation errors as error monitors has not been found effective in §6. Our experiments also confirmed similar conclusions reached by a number of previous workers; see [38], [41], and [44].

The failure to compute local truncation errors accurately is believed to cause the problem. In such cases, the use of interpolation-based monitors (§§3 and 4), or data-dependent grids as

TABLE 3
Results for Test Problem 3.

Monitor	Steps	CPU	Updates	Error(RMS)	Comment
1	*	*	*	*	Diverge
2	*	*	*	*	Diverge
3	*	*	*	*	Diverge
4	*	*	*	*	Diverge
5	100	187	97	0.364	
6	*	*	*	*	Diverge
7	*	*	*	*	Diverge
8	*	*	*	*	Diverge
11	*	*	*	*	Diverge
12	*	*	*	*	Diverge
13	*	*	*	*	Diverge
14	*	*	*	*	Diverge
15	*	*	*	*	Diverge
16	*	*	*	*	Diverge
17	*	*	*	*	Diverge
18	91	134	80	0.348	
21	100	169	99	0.340	
22	100	185	99	0.358	
23	100	144	72	0.361	
24	100	170	99	0.340	
26	*	*	*	*	Diverge
27	100	176	99	0.374	
28	100	149	85	0.400	
29	*	*	*	*	Diverge
31	*	*	*	*	Diverge
32	100	172	99	0.342	
33	100	147	28	0.367	
34	96	161	85	0.405	

they are called in [40], is generally recommended. However, our experiments in §6 showed that this is not always sufficient, simply because such monitors also fail in some cases.

Our new idea in this section is to sharpen the error bounds from which previous error monitors were derived. Let us again consider the numerical solution of equation (1) in the space S_Π^k. Suppose that the L_2 norm is considered. Assume that $u \in C$ and $u_h \in S_\Pi^k$ is interpolated at one point $\tau_i \in I_i$ $(i = 1, \ldots, N)$. This leads to two error bounds that we can consider minimizing *at the same time* over admissible monitors. The first bound for any grading function ξ with monitor M is given by (refer to $m = 0$ in §3)

$$\|e\|_2^2 \leq \frac{1}{(\pi N)^{2(k+1)}} \int_a^b \frac{[u^{(k+1)}]^2}{[\rho]^{2(k+1)}} dx(1 + O(h)), \tag{33}$$

where $\rho = \xi'$. On the other hand, we have the following estimate:

$$\int_{x_{i-1}}^{x_i} e^2 dx \leq c^2 h_i^{2k+3} T^2(u(x_{i-1/2}))(1 + O(h)) + \text{h.o.t.} \tag{34}$$

where h.o.t. denotes the high-order terms in h (negligible), since $|e| \leq c\, h_i^{k+1} T(u(x_{i-1/2})) + O(h_i^{k+2})$; see (4). For the same grading function ξ with monitor M, we again have

$$h_i = \frac{1}{N\rho(x_{i-1/2})}(1 + O(h_i)),$$

and furthermore,

$$\|e\|_2^2 \leq \frac{c^2}{N^{2(k+1)}} \sum_{i=1}^{N} \frac{T^2(u(x_{i-1/2}))}{\rho^{2(k+1)}} h_i(1 + O(h)) + \text{h.o.t.}$$

Thus the second error bound is given as

$$\|e\|_2^2 \leq \frac{c^2}{N^{2(k+1)}} \int_a^b \frac{T^2}{\rho^{2(k+1)}} dx(1 + O(h)) + \text{h.o.t.} \tag{35}$$

To obtain a minimization solution for right-hand sides of both (33) and (35), we now combine the two error bounds, giving (since $\rho = \xi'$)

$$\|e\|_2^2 \leq \frac{c^2}{(1 + c^2\pi^{2k+2})N^{2(k+1)}} \int_a^b \frac{[u^{(k+1)}]^2 + T^2}{[\xi']^{2(k+1)}} dx(1 + O(h)) + \text{h.o.t.} \tag{36}$$

THEOREM 7.1. *Assume that $u_h \in S_\Pi^k$ is the numerical solution of* (1) *from using the numerical method satisfying* (10). *Then if u_h is also interpolatory at one point in each interval I_i of mesh Π, the optimal monitor function in the L_2 norm is*

$$M = \left\{[u^{(k+1)}]^2 + T^2\right\}^{1/[2(k+1)+1]}. \tag{37}$$

Proof. We can minimize the right-hand side of (36) over all admissible grading functions ξ by solving the Euler equation

$$\frac{d}{dx} \frac{[u^{(k+1)}]^2 + T^2}{[\xi']^{2(k+1)+1}} = 0,$$

which yields

$$\xi(x) = \int_a^x \rho(u, y)dx \quad \text{with } \rho = M(u, x) \Big/ \int_a^b M(u, z)dz,$$

where M is as in (37). Refer to the proof of Theorem 1. The proof is complete. □

As examples, the choices of $k = 0$ and 1 lead to the following optimal monitors, respectively,

$$M = [u'^2 + T^2]^{1/3},$$
$$M = [u''^2 + T^2]^{1/5}.$$

Remarks. (i) Note that the minimization of the right-hand side of (35) alone would result in the same optimal monitor as that given in (11) due to [34]. This may then be viewed as an alternative proof for (11). (ii) Other treatments of simultaneously minimizing (33) and (35) may lead to the similar "optimal" monitor functions, e.g.,

$$M = \{F\}^{1/[2(k+1)+1]}$$

where $F = \max\left\{[u^{(k+1)}]^2/\pi^{2(k+1)}, T^2\right\}$. These alternatives have also been shown to be useful in other experiments.

8. Further numerical experiments.

8.1. The 1D examples. Having presented a new theory in §7, we now test its practical performance by comparing it with the experiments carried out in §6. At this point, we include

TABLE 4
Results for the new monitors vs. the arc-length monitor.

Problem	M	Steps	CPU	Updates	Error(RMS)	Comment
1	35	322	110	263	0.129	
	36	469	209	345	0.129	
	37	301	127	0	0.055	No adaptation
2	35	476	193	339	0.179	
	36	401	224	359	0.176	
	37	*	*	*	*	Diverge
3	35	100	192	100	0.373	
	36	*	*	*	*	Diverge
	37	*	*	*	*	Diverge

the results from using the well-known arc-length monitor §5 (due to White [44], [45]), which is here denoted by M_{37}. Again we solve the model equation (31) using the same finite element method as in §6. Since $k = 2$, we choose to test the following monitors:

$$M_{35} = [u'''^2 + T^2]^{1/7} \qquad \text{(the new monitor for } k = 2 \text{ with } L_2\text{-norm)},$$

$$M_{36} = M_{35}(1 + u'^2)^{1/2} \qquad \text{(arc-length norm of } M_{35}),$$

$$M_{37} = (1 + u'^2)^{1/2} \qquad \text{(arc-length monitor from (30))}.$$

Solving the three test problems of §6, we show the results in Table 4. We observe that the new monitor M_{35} works well for all three problems, although M_{36} failed for Problem 3. The arc-length monitor, not theoretically favoured, failed for both Problems 2 and 3.

Now in terms of efficiency, accuracy, and theoretical justification, a comparison of results of Tables 1–4 suggests that the most robust monitors are

$$M_5 = g^{1/4},$$
$$M_{18} = [u'''^2 + u''''^2]^{1/7}(1 + u'^2)^{1/2},$$
$$M_{35} = [u'''^2 + T^2]^{1/7}$$

although the following monitors also performed well: M_{21-24}, M_{27-28}, M_{32}, M_{34}. (See §6.) We shall make further comparisons in the next subsection.

8.2. Some 2D examples. Our 1D tests have shown that some monitors may not perform well practically, although they prove to be useful theoretically. Now we further apply those good 1D monitors (for quadratic approximations) to some 2D examples and see how they behave.

We use the adaptive tensor-product meshes for our tests. The simple strategy of equidistributing the average errors in both coordinate directions is adopted. That is, the mesh generation is done in a "1D" fashion. Similar meshes were previously used in [5], [31], and [40] for higher dimensions, where in [40] the averaging process was referred to as "equidistributing in slabs." The precise details of our implementation are described in [12]. Thus a new mesh is obtained after one sweep of 1D error equidistribution in both x and y directions, resulting in new rectangular grids. Our simple choice of strategies has also made it possible to use an existing code that is based on a rectangular structured grid.

The advantages as well as disadvantages of using tensor-product meshes were first discussed in [18]. Basically "diagonal features" in the underlying solution (e.g., errors are large along one diagonal direction of the domain simultaneously) will force the resulting mesh to be

uniform, therefore rendering the whole mesh adaptation process meaningless. However, such cases are physically rare in our primary application area of semiconductor process simulation and the use of the tensor-product meshes is adequate. See §8.3 for using quadrilateral meshes.

The test problem, as described in [15], is the 2D dopant diffusion equations

$$\frac{\partial C_k}{\partial t} = \mathrm{Div}\left[D_k \mathrm{grad} C_k + Z_k C_k \mathrm{grad}\Phi\right] \quad k = 1, \ldots, r \quad \text{and} \quad (x, y) \in \Omega, \tag{38}$$

where $C_k = C_k(x, y, t)$ is the concentration of the kth dopant, D_k is the diffusion coefficient, $Z_k = \pm 1$ depends on the dopant used (-1 for singly ionized acceptors, $+1$ for donors), and Φ is the electrostatic potential. Taking the transform $f_k = \log C_k$, (38) becomes

$$\frac{\partial f_k}{\partial t} = \mathrm{Div}\left[D_k \mathrm{grad}(f_k + Z_k\Phi)\right] + \mathrm{grad} f_k \cdot \mathrm{grad}(f_k + Z_k\Phi). \tag{39}$$

The system (39) is now solved by the finite element code [28] using quadratic tensor-product B-splines. See [15] and [28] for details.

We have chosen to test the following three problems in particular where all spatial sizes are measured in microns.

TD-1. $\Omega = [0, 2] \times [0, 2]$, 100 Kev boron of dose 10^{15} cm^{-2} for 80-minute diffusion at 1000°C, using 16×16 finite element boxes.

TD-2. $\Omega = [0, 1] \times [0, 1]$, 200 Kev arsenic of dose 10^{15} cm^{-2} for 50-minute diffusion at 1000°C, using 32×32 finite element boxes.

TD-3. $\Omega = [0, 1] \times [0, 1]$, 20 Kev boron of dose 10^{12} cm^{-2} and 50 Kev phosphorus of dose 10^{15} cm^{-2} for 5-minute diffusion at 1000°C, using 32×32 finite element boxes.

The initial Pearson IV ion implantation profiles are shown in Figs. 1, 2, and 3, respectively. For comparison purposes, we denote by M_0 the monitor presently used in the finite element code of [28], which is

$$M_0 = |\Phi''|^{1/2}.$$

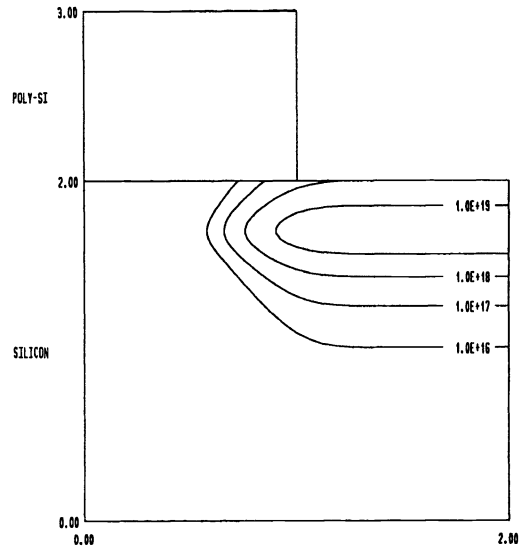


FIG. 1. *The initial profile for the* 2D *problem* TD-1.

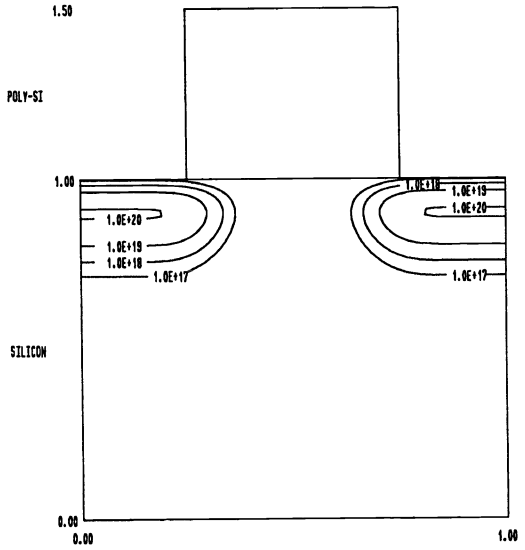


FIG. 2. *The initial profile for the* 2D *problem* TD-2.

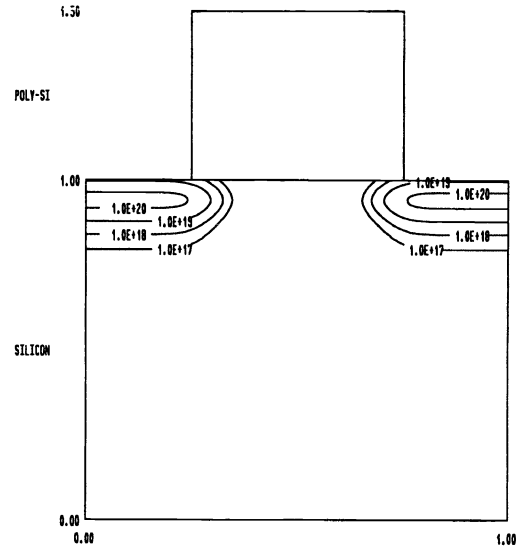


FIG. 3. *The initial profile for the* 2D *problem* TD-3.

As in [15], we again use as an accuracy indicator the relative mass balance error, defined by

$$Q_t = \frac{(M_t - M_0)}{M_0},$$

where M_t and M_0 are the initial and final total mass, respectively. Note that such a quantity is not necessarily decreasing in absolute magnitude as t increases.

The results from solving problems TD-1, TD-2, and TD-3 are presented in Tables 5–7, where the same notation is used as in §6. We observe that the error monitors M_5, M_{18}, and M_{35} performed rather well for all three problems, and also that the 2D test results are in good

TABLE 5
Results for the 2D *problem* TD-1.

Monitor	Steps	CPU	Updates	Mass error	Comment
0	20	85	3	3.0E-3	
5	20	73	2	2.3E-4	
18	29	132	20	1.6E-3	
21	18	82	12	6.3E-2	
22	21	89	9	-8.1E-4	
23	19	76	5	-1.5E-3	
24	25	101	10	3.5E-5	
34	20	79	3	9.6E-5	
35	25	119	23	2.4E-4	Figs. 4 & 5

TABLE 6
Results for the 2D *problem* TD-2.

Monitor	Steps	CPU	Updates	Mass error	Comment
0	*	*	*	*	Failed
5	137	1550	8	-2.4E-3	
18	33	615	31	1.9E-3	
21	45	728	41	9.4E-5	
22	*	*	*	*	
23	72	875	15	3.0E-4	
24	58	819	26	-3.8E-4	
34	54	753	20	-2.1E-3	
35	29	504	30	4.5E-4	Figs. 6 & 7

TABLE 7
Results for the 2D *problem* TD-3.

Error monitor	Time-steps	Sun CPU	Mesh updates	Mass error		Comment
				Boron	Phosphorus	
0	50	975	0	-1.1E-2	1.9E-3	
5	15	466	5	-2.9E-3	-3.0E-5	
18	20	790	16	-2.5E-3	1.7E-3	
21	13	539	13	-5.2E-4	1.2E-4	
22	16	639	14	-5.1E-4	-2.4E-5	
23	13	514	11	-2.2E-4	5.5E-3	
24	17	692	16	2.4E-4	2.5E-4	
34	17	588	8	1.1E-3	-1.9E-4	
35	13	532	13	-6.1E-4	-8.8E-4	Figs. 8 & 9

agreement with the 1D experiments of §§6 and 8.1. It perhaps requires more experiments to make conclusive remarks. However, monitors such as M_{35}, which take more account of the solution error, appear to be the overall winners and are therefore recommended. For further illustration, we show in Figs. 4–9 the final solution and the adapted finite element mesh from using the monitor M_{35}, where in all cases the p–n junctions of physical importance lie in the region well represented by the mesh adaptation strategy.

8.3. Discussion of 2D equidistribution strategies. Our implementation of 2D equidistribution in §8.2 is one of the simplest of many such strategies, although some may not give unique meshes. To allow generality, the tensor-product meshes may in practice be replaced by

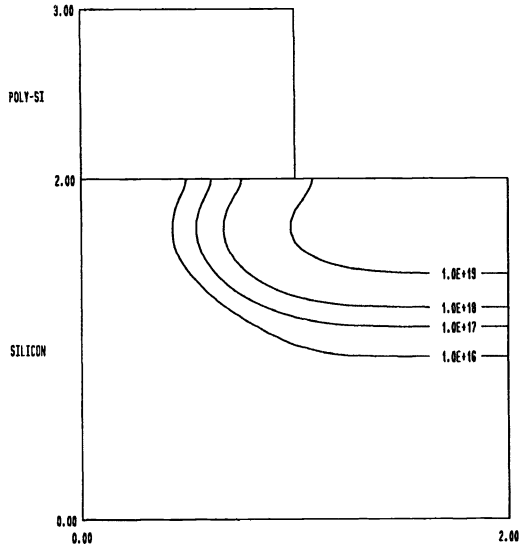


FIG. 4. *The final solution for the* 2D *problem* TD-1.

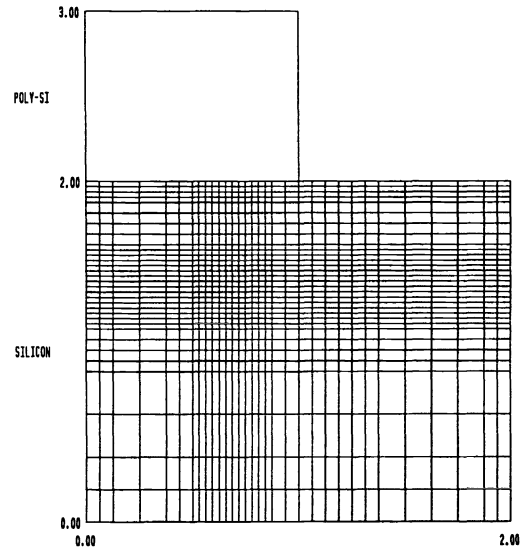


FIG. 5. *The final adapted mesh for the* 2D *problem* TD-1.

quadrilateral meshes. Recently, much work has been done on the use of adaptive quadrilateral meshes. Refer to [2], [3], [11], [41], [20], and the references therein.

However, there appears to be no analogous and elegant error equidistribution analysis (predicting good monitors) or methods (guiding the implementation) in the 2D case. Of course, one must deal with the problem of ideal error distribution and mesh (structure) requirement for a particular numerical method. As far as the generalization of equidistribution strategies is concerned, there are two main approaches: (i) the curve-by-curve grid line equidistribution approach in the computational space; see [19]. (ii) the weighted area (volume) variation minimization approach; see [8]. In fact, the grid system from the latter case will not have

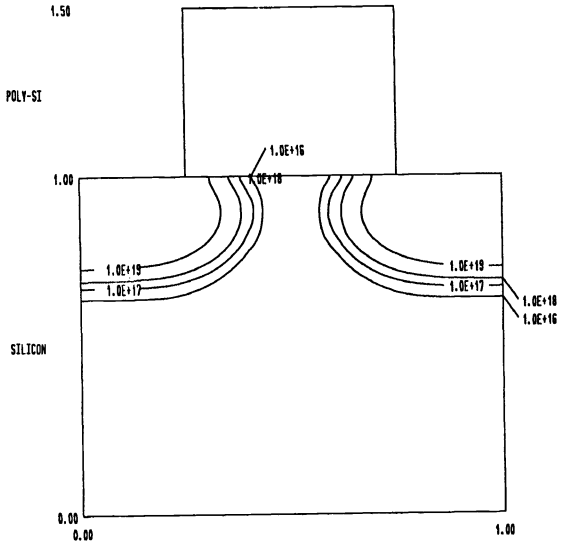


FIG. 6. *The final solution for the* 2D *problem* TD-2.

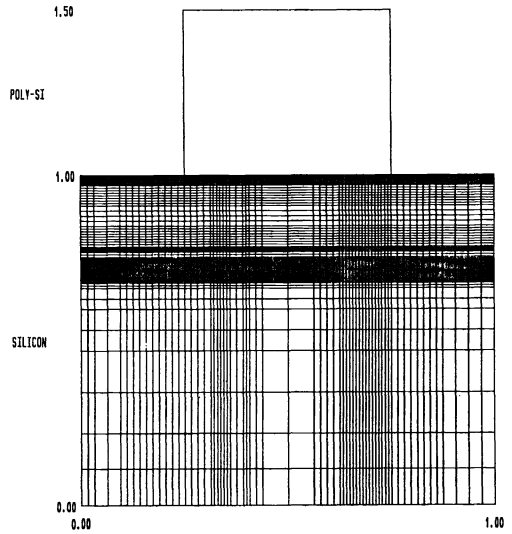


FIG. 7. *The final adapted mesh for the* 2D *problem* TD-2.

unique solutions (as also noted in [2]) and must be combined with some smoothness control measure to ensure uniqueness; see [8] and [27].

The performance of these adaptive quadrilateral meshes for solving nonlinear diffusion equations is to be observed and is under current investigation. Our recent work has provided a better understanding of the curve-by-curve grid line equidistribution approach and has further shown a new way of enhancing mesh smoothness; see [13] for details. Besides, there are also methods that one can adopt for direct 2D grid generation. Refer to [1], [42], and [47] among others.

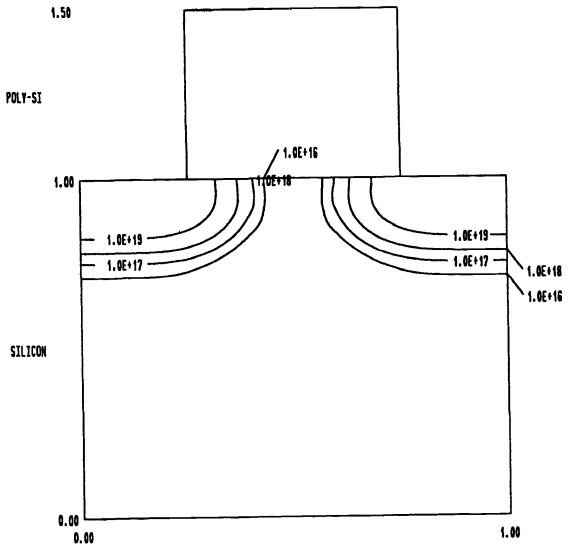


FIG. 8. *The final solution for the* 2D *problem* TD-3.

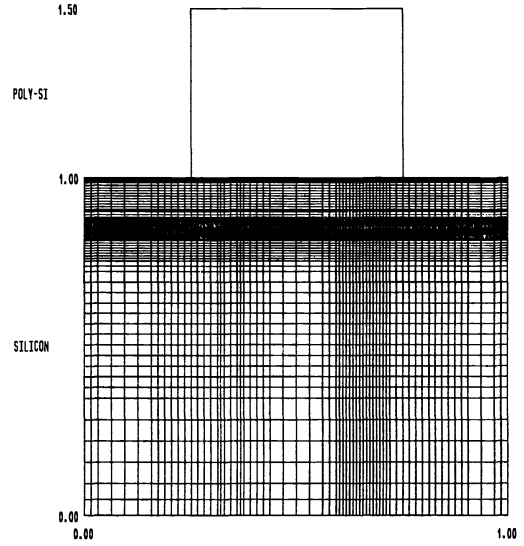


FIG. 9. *The final adapted mesh for the* 2D *problem* TD-3.

9. Conclusions. We have developed as well as reviewed a number of error equidistribution strategies for the purpose of mesh adaptation in the solution of parabolic PDEs. New and optimal monitors have been derived and tested in the context of solving 1D and 2D nonlinear diffusion test equations that show improvements over certain well-known monitors. The test results showing the success of the L_1-norm monitors using local truncation errors have not been reported before, as far as we are aware. Our work for the new equidistribution theory (§7) also points to a way of constructing more robust optimal error monitors once further information is available regarding the underlying numerical method. For the quadratic

approximations tested, the overall "best" monitor is found to be M_{35}, although certain monitors optimal for lower-order polynomials (constants and linears) also performed well. The new monitors proposed may be applied to solve more general operator equations. However, it remains to be seen how they compare with existing monitors for wider applications.

Acknowledgments. The author wishes to thank Drs. Mike J. Baines, Nancy K. Nichols, and Pete K. Sweby for many helpful discussions and the referees for constructive comments.

REFERENCES

[1] I. ALTAS AND J. W. STEPHENSON, *A two dimensional adaptive mesh generation method*, J. Comput. Phys., 94 (1991), pp. 201–224.

[2] D. C. ARNEY AND J. E. FLAHERTY, *A two-dimensional mesh moving technique for time-dependent partial differential equations*, J. Comput. Phys., 67 (1986), pp. 124–144.

[3] ———, *An adaptive mesh-moving and local refinement method for time-dependent partial differential equations*, ACM Trans. Math. Software, 16(1) (1990), pp. 48–71.

[4] U. M. ASCHER, R. M. M. MATTHEIJ, AND R. D. RUSSELL, *Numerical solution of boundary value problems for ordinary differential equations, Chap. 9, Mesh selection*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[5] J. V. ASHBY, E. M. AZOFF, R. F. FOWLER, R. J. FAWCETT, AND C. GREENOUGH, *The adaptive solution of three dimensional semiconductor device problems*, in Proc. of 3rd Internat. Conf. on Numerical Grid Generation in CFD and Related Fields, Barcelona, Spain, 1991.

[6] I. BABUSKA, J. CHANDRA, AND J. E. FLAHERTY, EDS., *Adaptive computational methods for partial differential equations*, Society of Industrial and Applied Mathematics, Philadelphia, PA, 1983.

[7] I. BABUSKA, O. C. ZIENKIEWICZ, J. GAGO, AND E. R. DE A. OLIVEIRA, EDS., *Accuracy Estimates and Adaptive Refinement in Finite Element Calculations*, Wiley-Interscience, New York, NY, 1986.

[8] J. U. BRACKBILL AND J. S. SALTZMAN, *Adaptive zoning for singular problems in two dimensions*, J. Comput. Phys., 46 (1982) pp. 342–368.

[9] N. CARLSON AND K. MILLER, *Gradient weighted moving finite elements in two dimensions*, in Finite element—theory and applications, Chap. 7, D. L. Dwoyer, M. Y. Hussaini, and R. G. Voight, eds., Springer-Verlag, New York, Berlin, 1988, pp. 151–164.

[10] G. F. CAREY AND H. T. DINH, *Grading functions and mesh redistribution*, SIAM J. Numer. Anal., 22 (1985), pp. 1028–1040.

[11] D. CATHERALL, *Solution adaptivity with structured grids*, in Numerical methods in Computational Fluid Dynamics, M. J. Baines, et al., eds., University of Reading, England, April 1992, Oxford University Press, 1993.

[12] K. CHEN, *Selection of optimal meshes for the solution of nonlinear dopant diffusion problems*, Internat. J. Comput. Math. EEE (COMPEL), 11 (1992), pp. 433–444.

[13] ———, *Two dimensional adaptive quadrilateral mesh generation*, submitted.

[14] K. CHEN, M. J. BAINES, AND P. K. SWEBY, *On time step selection for solving* 1D *nonlinear diffusion equations*, Numerical Analysis Report 10/91, Dept. of Mathematics, University of Reading, England, 1991.

[15] ———, *On an adaptive time stepping strategy for solving nonlinear diffusion equations*, J. Comput. Phys., 105 (1993), pp. 324–332.

[16] J. CHRISTIANSEN AND R. D. RUSSELL, *Error analysis for spline methods with applications to knot selection*, Math. Comp., 32 (1978), pp. 415–419.

[17] J. M. COYLE, J. E. FLAHERTY, AND R. LUDWIG, *On the stability of mesh equidistribution strategies for time-dependent partial differential equations*, J. Comput. Phys., 62 (1986), pp. 26–39.

[18] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, Berlin, 1978.

[19] P. R. EISEMAN, *Adaptive grid generation*, Comput. Methods Appl. Mech. Engrg., 64 (1987), pp. 321–376.

[20] P. EISEMAN AND G. ERLEBACHER, *Grid Generation for the Solution of Partial Differential Equations*, NASA ICASE Report No. 87-57, Langley Research Center, Hampton, VA, Aug., 1987.

[21] J. E. FLAHERTY, P. J. PASLOW, M. S. SHEPHARD, AND J. D. VASILAKIS, EDS., *Adaptive methods for partial differential equations*, Society of Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[22] D. O. GOUGH, E. A. SPIEGEL, AND J. TOOMRE, *Highly stretched meshes as functionals of solutions*, in Proc. 4th Internat. Conf. Numerical Methods in Fluid Dynamics, Lecture Notes in Physics, 35, R. D. Richmyer, ed., Springer-Verlag, New York, Berlin, 1974.

[23] J. M. HYMAN, *Adaptive moving mesh methods for partial differential equations*, in Advances in Reactor Computations, American Nuclear Society Press, La Grange Park, IL, 1983.

[24] L. W. JOHNSON AND R. D. RIESS, *Numerical Analysis*, 2nd ed., Addison-Wesley, Reading, MA, 1982.
[25] J. KING AND C. PLEASE, *Diffusion of dopant in crystalline silicon*, IMA J. Appl. Math., 37 (1986), pp. 185–197.
[26] J. KAUTSKY AND N. K. NICHOLS, *Equidistributing meshes with constraints*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 449–511.
[27] G. LIAO, *Variational approach to grid generation*, Numer. Meth. PDE's, 8 (1992), pp. 143–147.
[28] J. LORENZ AND M. SVOBODA, *ASWR – method for the simulation of dopant redistribution in silicon*, in Simulation of Semiconductor Devices and Processes, G. Baccarani and M. Rudan, eds., Tecnoprint, Bologna, Italy, 3 (1988), pp. 243–254.
[29] K. MILLER, *Alternate modes to control the nodes in the moving finite element method*, in Adaptive Computational Methods for Partial Differential Equations, Society of Industrial and Applied Mathematics, Philadelphia, PA, 1983, pp. 165–182.
[30] ———, *Recent results on finite element methods with moving nodes*, in Accuracy Estimates and Adaptive Refinement in Finite Element Calculations, Wiley-Interscience, New York, 1986, pp. 325–338.
[31] K. NAKAHASHI AND G. S. DEIWERT, *Three dimensional adaptive grid method*, AIAA J., 24 (1986), pp. 948–954.
[32] C. E. PEARSON, *On a differential equation of boundary layer type*, J. Math. Phys., 47 (1968), pp. 134–154.
[33] J. PERAIRE, M. VAHDATI, K. MORGAN, AND O. C. ZIENKIEWICZ, *Adaptive remeshing for compressible computations*, J. Comput. Phys., 72 (1987), pp. 449–466.
[34] V. PEREYRA AND E. G. SEWELL, *Mesh selection for discrete solution of boundary problems in ordinary differential equations*, Numer. Math., 23 (1975), pp. 261–268.
[35] L. R. PETZOLD, *Observations on an adaptive moving grid method for one dimensional systems of partial differential equations*, Appl. Numer. Math., 3 (1987), pp. 347–360.
[36] Y. H. REN AND R. D. RUSSELL, *Moving mesh techniques based on equidistribution, and their stability*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1265–1286.
[37] W. C. RHEINBODLT, *Adaptive mesh refinement processes for finite element solutions*, Internat. J. Numer. Meth. Engrg., 17 (1981), pp. 649–662.
[38] R. D. RUSSELL, *Mesh selection methods*, in Codes for boundary value problems, Lecture Notes in Computer Science 74, B. Childs, ed., Springer-Verlag, Berlin, 1979.
[39] W. H. A. SCHILDERS, *A novel approach to adaptive meshing for the semiconductor problem* in Simulation of Semiconductor devices and Processes, 3, G. Baccarani and M. Ruden, eds., Tecnoprint, Bologna, Italy, 1988, pp. 519–527.
[40] P. K. SWEBY, *An approximate equidistribution technique for unstructured Grids*, in Proc. Numerical Methods for Fluid Dynamics III, K. W. Morton and M. J. Baines, eds., Oxford University Press, 1988.
[41] J. F. THOMPSON, *A survey of dynamically adaptive grids in the numerical solution of partial differential equations*, Appl. Numer. Math., 1 (1983), pp. 3–27.
[42] J. F. THOMPSON, Z. U. A. WARSI, AND C. W. MASTIN, *Numerical Grid Generation: Foundations and Applications*, North-Holland, Amsterdam, 1985.
[43] J. M. VERWER AND R. A. TROMBERT, *Analysis of an adaptive finite difference method for time dependent PDEs*, in Numerical Analysis, D. F. Griffiths and G. A. Watson, eds., Longman Press, London, 1991, pp. 2267–2284.
[44] A. B. WHITE, *On selection of equidistribution meshes for two-point boundary value problems*, SIAM J. Numer. Anal., 16 (1979), pp. 472–502.
[45] ———, *On the numerical solution of initial/boundary-value problems in one space dimension*, SIAM J. Numer. Anal., 19 (1982), pp. 683–697.
[46] K. WRIGHT, A. H. A. AHMED, AND A. H. SELEMAN, *Mesh selection in collocation for boundary value problems*, IMA J. Numer. Anal., 11 (1991), pp. 7–20.
[47] O. C. ZIENKIEWICZ AND J. Z. ZHU, *Adaptivity and mesh generation*, Internat. J. Numer. Meth. Engrg., 32 (1991), pp. 783–810.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 819–845, July 1994

004

A SCHEME FOR CONSERVATIVE INTERPOLATION ON OVERLAPPING GRIDS*

G. CHESSHIRE† AND W. D. HENSHAW†

Abstract. This paper describes how to interpolate in a conservative manner when solving systems of conservative laws on overlapping grids. Overlapping grids are a flexible and efficient way to create grids for complicated regions. Before now, however, no general method had been developed for conservative interpolation. The basic idea consists of assuming that the interpolation coefficients are free parameters and then deriving constraints to ensure that the interpolation is conservative. A system of equations is then solved to determine the coefficients. A two-dimensional viscous Burger's equation is solved to demonstrate the conservative nature of the scheme.

Key words. conservative interpolation, overlapping grids, composite grids, finite difference methods

AMS subject classifications. 65M05, 65M50, 65N05, 65N50

1. Introduction.

Standard interpolation procedures for overlapping grids do not lead to conservative methods for the solution of conservation laws. To obtain conservative schemes, it is necessary to interpolate fluxes at the overlapping boundaries instead of interpolating solution values. In one dimension we show that any consistent interpolation scheme for the fluxes is conservative. In two or three dimensions the result is not as easy. We describe a scheme to interpolate conservatively in two or three space dimensions. We implement the method in two dimensions, on a viscous Burger's equation, and show results of shocks passing through interfaces between grids. These computations demonstrate the conservative nature of the interpolation.

We are interested in numerically solving a system of conservation laws using a finite-difference or finite-volume method on overlapping grids. An overlapping grid consists of a set of logically rectangular curvilinear grids that cover a domain and overlap where they meet. Grid functions defined on the domain are matched by interpolation at the overlapping boundaries between component grids. For solving problems that are smoothly represented on the grid, standard interpolation procedures result in accurate solutions to partial differential equations (PDEs), as shown, for example, in Chesshire and Henshaw [7]. In this case it is not necessary that the discrete scheme be exactly conservative. If one is willing to have a fine enough grid to resolve sharp features such as shocks, then standard interpolation will give accurate results. For solving problems whose solution is not smoothly represented on the grid, however, it may be necessary to interpolate in a conservative way to ensure that the discrete solution converges to a physically relevant solution and to ensure that the speeds and locations of shocks are correct. For many applications it is not feasible with current computers to have enough grid points to resolve shocks. Thus we are led to develop conservative schemes. Note that, in principle, conservative interpolation is required only where shocks actually cross the boundary between grids. In many cases the problem can be avoided by aligning grids with the shocks. In general, however, it is difficult to avoid having a shock cross a grid boundary, especially in three dimensions.

An excellent discussion of conservative interpolation can be found in the paper of M. Berger [2]. She is concerned with both spatial and timewise discontinuities, the latter occurring when the timestep is different on different grids. In one space dimension Berger shows how to interpolate the fluxes in a conservative fashion given any discrete approximation to the integral. In two dimensions she shows how to interpolate conservatively for a restricted

*Received by the editors July 1, 1991; accepted for publication (in revised form) June 24, 1993.

†IBM Research Division, T.J. Watson Research Centre, Yorktown Heights, New York 10598 (whensha@watson.ibm.com).

class of problems, when the overlapping grids are related to each other in particular ways. Although the paper suggests an algorithm for the general case, the details are somewhat lacking, and it was apparently never implemented. Also of interest is the work of M. Rai [14] who shows how to interpolate conservatively on patched grids. These grids are composite grids whose boundaries match exactly on certain curves. Conservative interpolation in this case is somewhat simpler since the boundary across which the flux must be matched is well defined and the interpolation is in one dimension lower than that of the PDE.

A conservative difference scheme exactly conserves some discrete approximation to the integral of the solution, up to contributions of fluxes through the boundaries. The conservation property must be local in the sense that the scheme should be conservative on any subdomain. For an overlapping grid the approximation to the integral is not clearly defined where the grids overlap. In general, the method of Berger's paper consists of starting with a discrete approximation to the integral and then determining the interpolation coefficients. In the approach presented here we leave both the weights in the approximation to the integral and the coefficients in the interpolation of the fluxes as free parameters. The condition that the scheme should be locally conservative imposes constraints on these parameters. In this paper we extend Berger's one-dimensional result to show that any interpolation of the fluxes is conservative. In two dimensions we describe and implement a general algorithm for conservative interpolation on overlapping grids, such as those created by the program CMPGRD [10], [5], [7].

We present numerical examples for the two-dimensional Burger's equation that show a (viscous) shock moving through an overlapping grid. The conservative interpolation weights that we generate do indeed conserve the solution to roundoff errors. In the future we will study the effectiveness of this approach to conservative interpolation when coupled with some of the state of the art methods for computing solutions to the Euler or Navier–Stokes equations. For now we remark that results from our preliminary investigations suggest that some care will be required in this regard. It is important that interpolated fluxes are independent of the local coordinate system of a component grid so that it makes sense to use the fluxes on a different component grid. Further details and numerical experiments can be found in [8].

One of our motivations for doing this work was the fact that it has been reported, by Pärt-Enander and Sjögreen [13], for example, that very slowly moving shocks can become stuck or distorted at the boundary between two overlapping grids. Whether this difficulty is solely due to the interpolation or to a coupling between the interpolation and the integration scheme is difficult to decide. Hopefully with the availability of a procedure for conservative interpolation it will be easier to isolate the source of the problem in such situations. It should be pointed out, however, that the problem only seems to occur for extremely slow moving shocks. Many researchers have successfully used nonconservative interpolation on overlapping grids to compute complex flows in two and three space dimensions, including Berger and Oliger [3], Steger et al. [6], [17], Brown [4], and Pärt-Enander and Sjögreen [12].

We have not considered the stability properties of time-stepping schemes combined with our conservative interpolation. Stability of difference schemes on overlapping grids has been considered by a number of authors including Starius [16], Reyna [15], Trefethen [19], Berger [1], and Thuné [18]. In practice we have not encountered problems due to instabilities at the overlapping boundaries.

1.1. Weak solutions and conservation. We first review some well-known concepts about conservation laws and weak solutions that will motivate our approach for developing a conservative interpolation scheme for overlapping grids. Consider the solution of the conservation equations

$$\mathbf{u}_t + \mathbf{f}_x + \mathbf{g}_y = 0. \tag{1}$$

We have in mind a nonlinear hyperbolic system such as the equations of gas dynamics. Solutions to this equation may develop discontinuities such as shocks. A *solution* with a shock is not a classical solution because it is not differentiable, but it is still physically relevant in the sense that it is the limiting behaviour as certain neglected quantities such as viscosity or heat conduction tend to zero. Thus we wish to extend our notion of solution to this more general class of solutions, called weak solutions. Typically such equations are derived from the consideration of the conservation of some quantity such as mass, momentum, or energy through a small control volume ΔV

$$\frac{d}{dt}\int_{\Delta V} \mathbf{u}\, d\mathbf{x} + \int_{\partial \Delta V} (\mathbf{f}, \mathbf{g}) \cdot \mathbf{n}\, ds = 0, \tag{2}$$

and state that the time rate of change of the integral of $\mathbf{u}$ over ΔV is equal to the integral of the normal component of the fluxes through the boundary $\partial \Delta V$. The PDE formulation follows by letting the size of this arbitrary control volume go to zero. The equivalence of the two formulations requires certain smoothness conditions on the solution. We can extend our class of solutions by looking for solutions that satisfy (2) for all possible control volumes. This formulation does not require $(\mathbf{f}, \mathbf{g})$ to be differentiable. For later reference we define two types of weak solutions.

DEFINITION 1.1. *We say that* $\mathbf{u}$ *is a weak solution of type* I *to the conservation equations* (1) *if it satisfies the integral form of the equations* (2) *for any (suitably smooth) control volume* ΔV.

The finite-volume method is the discrete version of this formulation. This method imposes a discrete form of (2) for each cell in the computational grid. There is another common way of defining a more general class of solutions.

DEFINITION 1.2. *We say that* $\mathbf{u}$ *is a weak solution of type* II *to the conservation equations* (1) *if*

$$\int_t \int_\Omega (\phi_t \mathbf{u} + \nabla \phi \cdot (\mathbf{f}, \mathbf{g}))\; d\mathbf{x}\, dt - \int_{\partial \Omega} \mathbf{u}(\mathbf{x}, 0)\phi(\mathbf{x}, 0) d\mathbf{x} = 0 \tag{3}$$

for all smooth test functions $\phi(\mathbf{x}, t)$ *of compact support.*

This expression can be formally obtained by multiplying the conservation law by ϕ and integrating by parts. The main point is that the possibly discontinuous functions $\mathbf{u}$ and $\mathbf{f}$ are not differentiated in this formulation.

1.2. Conserative schemes on a rectangular grid. Let us now consider the numerical solution of (1) on a single rectangular grid. A discrete scheme is said to be in conservation form if it can be written as

$$\frac{d}{dt}\mathbf{u}_{ij} + \frac{1}{J_{ij}}\left(\Delta_{+i}\mathbf{F}_{ij} + \Delta_{+j}\mathbf{G}_{ij}\right) = 0, \tag{4}$$

where

$$\begin{aligned} J_{ij} &= \text{approximate area of cell } ij, \\ \mathbf{F}_{ij} &= \text{approximation to } \mathbf{f} \text{ on left face of cell } ij, \\ \mathbf{G}_{ij} &= \text{approximation to } \mathbf{g} \text{ on bottom face of cell } ij, \end{aligned}$$

and the difference operators are defined by

$$\begin{aligned} \Delta_{+i}\mathbf{F}_{ij} &= \mathbf{F}_{i+1,j} - \mathbf{F}_{ij}, \quad \Delta_{+j}\mathbf{G}_{ij} = \mathbf{G}_{ij+1} - \mathbf{G}_{ij}, \\ \Delta_{-i}\mathbf{F}_{ij} &= \mathbf{F}_{ij} - \mathbf{F}_{i-1j}, \quad \Delta_{-j}\mathbf{G}_{ij} = \mathbf{G}_{ij} - \mathbf{G}_{ij-1}. \end{aligned}$$

In particular the finite-volume method of discretization generates a scheme in conservation form. If we consider any subdomain $B \subset \Omega$, then a scheme in conservation form conserves an approximation to the integral of $\mathbf{u}$ over B

$$S_B(t) = \sum_{ij \in B} \mathbf{u}_{ij} J_{ij}$$

except for the fluxes coming through ∂B, the boundary of B. To see this, simply take the time derivative of S_B and substitute (4)

$$\begin{aligned} \frac{dS_B}{dt} &= \sum_{ij \in B} \frac{d\mathbf{u}_{ij}}{dt} J_{ij} = -\sum_{ij \in B} \{\Delta_{+i}\mathbf{F}_{ij} + \Delta_{+j}\mathbf{G}_{ij}\} \\ &= \sum_{ij \in \partial B} \pm\mathbf{F}_{ij} \pm \mathbf{G}_{ij} \\ &= \text{fluxes crossing the boundary.} \end{aligned}$$

All of the fluxes in the interior of B cancel, leaving only boundary terms. In the discrete case we see that the equation in conservation form satisfies the discrete analogue of the weak solution of type I (Def. 1.1).

Suppose that we have a discrete approximation to the integral and a discrete scheme in conservation form. Multiplication of the discrete scheme by $\phi_{ij} J_{ij}$ and integrating (summing) gives

$$\int_t \sum_{ij} \left\{ \phi_{ij} \frac{d}{dt} \mathbf{u}_{ij} J_{ij} + \phi_{ij}(\Delta_{+i}\mathbf{F}_{ij} + \Delta_{+j}\mathbf{G}_{ij}) \right\} dt = 0. \tag{5}$$

Summation by parts gives

$$\begin{aligned} &\int_t \sum_{ij} \left\{ \frac{d}{dt} \phi(t)_{ij} \mathbf{u}_{ij} + \frac{1}{J_{ij}} \{(\Delta_{-i}\phi_{ij})\mathbf{F}_{ij} + (\Delta_{-j}\phi_{ij})\mathbf{G}_{ij}\} \right\} J_{ij}\, dt \\ &\quad - \sum_{ij} \phi(0)_{ij} \mathbf{u}(0)_{ij} J_{ij} = 0. \end{aligned} \tag{6}$$

Since ϕ is smooth, if the solution converges then the above sums will converge to the integrals appearing in the definition of a weak solution of type II. In this case the solution will be a weak solution of type II. Thus we see that a scheme in conservation form, a discrete version of the integrated form of the conservation laws, (2), is compatible with the definition of weak solution type II (Def. 1.2). This idea that a solution to the equations in conservation form will converge to a weak solution of type II, provided the solution converges at all, is due to Lax and Wendroff [11].

It is important to understand that a conservative scheme must have the local conservation property. The integral (or approximation to the integral) over any subdomain must be conserved up to the fluxes crossing the boundary of the subdomain. It is not enough that the integral over the entire domain be conserved since this alone does not ensure that the solution will be a weak solution. In particular, shocks may travel with the wrong speed. Thus, for example, the naive approach of adjusting all values by a fixed amount at each timestep to be globally conservative is not locally conservative. This naive approach is not wrong in itself, but rather it is a mistake to believe that it will generate a weak solution or that shocks must necessarily move with the correct speed.

2. Conservation on one-dimensional overlapping grids. We consider the solution of the conservation equation

$$\mathbf{u}_t + \mathbf{f}_x = 0$$

on a one-dimensional overlapping grid on the interval $[a, b]$; see Fig. 1.

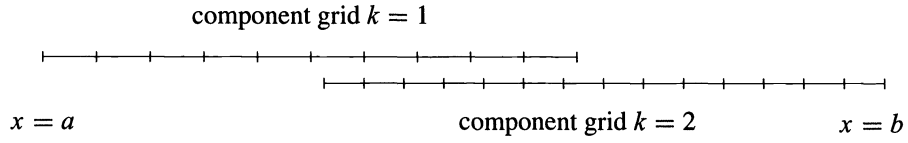


FIG. 1. *One-dimensional overlapping grid for the interval* $[a, b]$.

Solutions to this equation conserve the integral

$$I(t) = \int_a^b \mathbf{u}(x, t)dx.$$

That is,

$$\frac{dI}{dt} = \int_a^b \frac{\partial \mathbf{u}}{\partial t}dx = -\int_a^b \mathbf{f}_x dx = -[\mathbf{f}]_a^b.$$

The overlapping grid has two component grids labelled by $k = 1, 2$. The grid points on grid k are numbered $i = 0, \ldots, N_k + 1$; see Fig. 2.

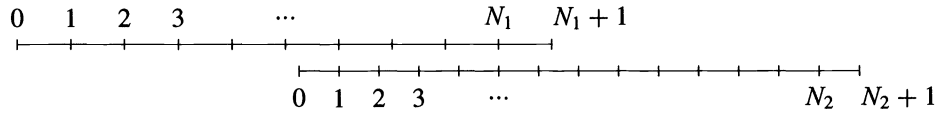


FIG. 2. *Numbering of grid points.*

Let $\mathbf{u}_{k,i}$ denote the discrete approximation to $\mathbf{u}$ on component grid $k = 1, 2$ for $i = 0, \ldots, N_k + 1$, defined at the midpoints of the cells and let $\mathbf{f}_{k,i}$ denote the discrete approximation to $\mathbf{f}$ on the cell edges; see Fig. 3.

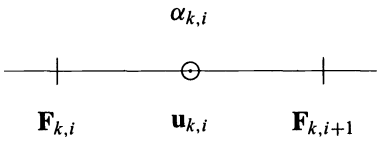


FIG. 3. *Location of discrete variables.*

The grid points will be denoted by $x_{k,i}$ with grid spacings $h_{k,i} = x_{k,i+1} - x_{k,i}$. Define the undivided difference operators

$$\Delta_+ \mathbf{F}_{k,i} = \mathbf{F}_{k,i+1} - \mathbf{F}_{k,i}, \qquad \Delta_- \mathbf{F}_{k,i} = \mathbf{F}_{k,i} - \mathbf{F}_{k,i-1},$$

and introduce the discrete scheme

$$\frac{d\mathbf{u}_{k,i}}{dt} + \frac{1}{h_{k,i}}\Delta_+ \mathbf{F}_{k,i} = 0 \tag{7}$$

on the two component grids $k = 1, 2$. Here the numerical flux $\mathbf{F}_{k,i}$ is some function of $\mathbf{u}_{k,i}$

$$\mathbf{F}_{k,i} = \mathbf{F}_N(\ldots, \mathbf{u}_{k,i-1}, \mathbf{u}_{k,i}, \mathbf{u}_{k,i+1}, \ldots)$$

such that $\mathbf{F}_N(\ldots, \mathbf{u}, \mathbf{u}, \mathbf{u}, \ldots) = \mathbf{f}(\mathbf{u})$. On an overlapping grid, extra information is needed on the boundary of a component grid that overlaps another component grid. The standard approach is to interpolate the solution value at such an *interpolation point* in terms of the solution values on the grid that is being overlapped. However, to achieve a conservative method, the flux is interpolated. (This comes from the fact that the flux function appears in a linear fashion in the conservation law while $\mathbf{u}$ has, in general, a nonlinear dependence.) The interpolation conditions are of the form (see Fig. 4)

$$\mathbf{F}_{1,N_1+1} = \sum_{j=M_1}^{M_2} \gamma_j \mathbf{F}_{2,j} \qquad 0 \le M_1 \le M_2 \le N_2, \tag{8}$$

$$\mathbf{F}_{2,0} = \sum_{j=L_1}^{L_2} \beta_j \mathbf{F}_{1,j} \qquad 0 \le L_1 \le L_2 \le N_1. \tag{9}$$

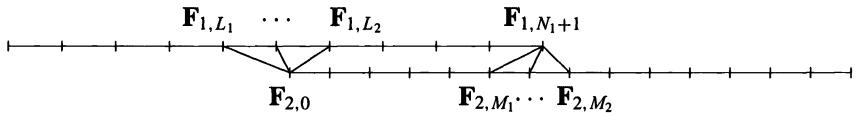


FIG. 4. *Interpolation of endpoints.*

Introduce the discrete approximation to the integral $I(t)$,

$$S(t) = \sum_{k=1}^{2} \sum_{i=0}^{N_k} \alpha_{k,i} \mathbf{u}_{k,i} h_{k,i}.$$

We will show that in one dimension, for any consistent interpolation of the fluxes (8), (9), the scheme is conservative. We first show that the scheme satisfies the discrete analogue of the local conservation property (Def. 1.1) since the algebra is somewhat simpler. We then show that the scheme is also consistent with the definition of a weak solution of type II.

THEOREM 2.1. *If the interpolation coefficients are first-order accurate,*

$$\sum_i \beta_i = 1 \text{ and } \sum_i \gamma_i = 1,$$

and are bounded,

$$\sum_i |\beta_i| = O(1) \text{ and } \sum_i |\gamma_i| = O(1),$$

then there exist $\alpha_{k,i}$ so that $dS/dt = -[\mathbf{f}]_a^b$ for any interval $[a, b]$ and $S(t) = I(t) + O(h)$, where $h = \max_{k,i} h_{k,i}$. If the interpolation is second-order accurate, then the integral is approximated to second order: $S(t) = I(t) + O(h^2)$. Note that the $\alpha_{k,i}$ must be chosen independent of $\mathbf{u}_k$ and $\mathbf{F}_k$.

Proof. It holds that

$$\frac{dS}{dt} = \sum_{i=0}^{N_1} \alpha_{1,i} \frac{d\mathbf{u}_{1,i}}{dt} h_{1,i} + \sum_{i=0}^{N_2} \alpha_{2,i} \frac{d\mathbf{u}_{2,i}}{dt} h_{2,i}$$

$$
\begin{aligned}
&= -\sum_{i=0}^{N_1} \alpha_{1,i}\Delta_+\mathbf{F}_{1,i} - \sum_{i=0}^{N_2} \alpha_{2,i}\Delta_+\mathbf{F}_{2,i} \\
&= \alpha_{1,0}\mathbf{F}_{1,0} + \sum_{1}^{N_1} \Delta_-(\alpha_{1,i})\mathbf{F}_{1,i} - \alpha_{1,N_1}\mathbf{F}_{1,N_1+1} \\
&\quad + \alpha_{2,0}\mathbf{F}_{2,0} + \sum_{1}^{N_2} \Delta_-(\alpha_{2,i})\mathbf{F}_{2,i} - \alpha_{2,N_2}\mathbf{F}_{2,N_2+1}.
\end{aligned}
$$

Using (8) and (9) for $\mathbf{F}_{1,N_1+1}$ and $\mathbf{F}_{2,0}$ gives

$$
\begin{aligned}
\frac{dS}{dt} = \alpha_{1,0}\mathbf{F}_{1,0} &+ \sum_{1}^{L_1-1} \Delta_-(\alpha_{1,i})\mathbf{F}_{1,i} + \sum_{L_1}^{L_2} \left\{\Delta_-(\alpha_{1,i}) + \alpha_{2,0}\beta_i\right\}\mathbf{F}_{1,i} \\
&+ \sum_{L_2+1}^{N_2} \Delta_-(\alpha_{1,i})\mathbf{F}_{1,i} + \sum_{1}^{M_1-1} \Delta_-(\alpha_{2,i})\mathbf{F}_{2,i} \\
&+ \sum_{M_1}^{M_2} \left\{\Delta_-(\alpha_{2,i}) - \alpha_{1,N_1}\gamma_i\right\}\mathbf{F}_{2,i} \\
&+ \sum_{M_2+1}^{N_2} \Delta_-(\alpha_{2,i})\mathbf{F}_{2,i} - \alpha_{2,N_2}\mathbf{F}_{2,N_2+1}.
\end{aligned}
$$

We now impose the conservation condition, $dS/dt = \mathbf{f}(a) - \mathbf{f}(b)$ for all $\mathbf{F}_{k,i}$. For simplicity, we assume that the boundary conditions are $\mathbf{F}_{1,0} = \mathbf{f}(a)$ and $\mathbf{F}_{2,N_2+1} = \mathbf{f}(b)$, so

$$
\begin{aligned}
\alpha_{1,0} &= 1, \\
\alpha_{2,N_2} &= 1, \\
\Delta_-(\alpha_{1,i}) &= 0 \quad i = 1, \ldots, L_1 - 1, L_2 + 1, \ldots, N_1, \\
\Delta_-(\alpha_{1,i}) + \alpha_{2,0}\beta_i &= 0 \quad i = L_1, \ldots, L_2, \\
\Delta_-(\alpha_{2,i}) &= 0 \quad i = 1, \ldots, M_1 - 1, M_2 + 1, \ldots, N_2, \\
\Delta_-(\alpha_{2,i}) - \alpha_{2,0}\gamma_i &= 0 \quad i = M_1, \ldots, M_2.
\end{aligned}
$$

These equations are equivalent to the following set of equations, where we have defined $\mu = \alpha_{2,0}$ and have used the fact that $\sum_j \beta_j = 1$ and $\sum_j \gamma_j = 1$:

$$
\begin{aligned}
\alpha_{1,i} &= 1 && i = 0, \ldots, L_1 - 1, \\
\alpha_{1,i} &= \alpha_{1,i-1} - \mu\beta_i && i = L_1, \ldots, L_2 - 1, \\
\alpha_{1,i} &= 1 - \mu && i = L_2, \ldots, N_1, \\
\alpha_{2,i} &= \mu && i = 0, \ldots, M_1 - 1, \\
\alpha_{2,i} &= \alpha_{2,i-1} + (1 - \mu)\gamma_i && i = M_1, \ldots, M_2 - 1, \\
\alpha_{2,i} &= 1 && i = M_2, \ldots, N_2.
\end{aligned} \tag{10}
$$

These last equations define the general solution to the problem in terms of the one free parameter μ. Thus we see that away from the overlap region and interpolation regions the integration weights $\alpha_{k,i}$ are equal to 1. In the region of overlap but away from the interpolation regions, the weights are also constant, with $\alpha_{1,i} = 1 - \mu$ and $\alpha_{2,i} = \mu$ (see Fig. 5).

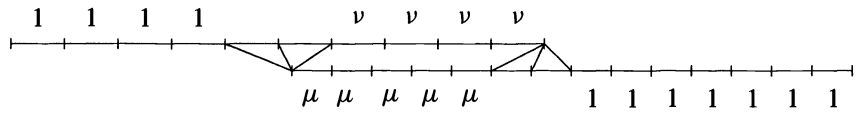


FIG. 5. *Solution for the integration weights $\alpha_{k,i}$. The weights are constant outside the interpolation regions. In the overlap region one grid has $\alpha_{1,i} = \nu$ and the other has $\alpha_{2,i} = \mu$ with $\mu + \nu = 1$.*

The parameter μ defines how to weight the two component grids in the sum $S(t)$ in the region where they overlap. Component grid $k = 1$ is weighted by $1 - \mu$ and component grid $k = 2$ is weighted by μ. Taking $\mu = \frac{1}{2}$ would give equal weight to each component grid in its contribution to the integral, although note that the numerical scheme (7), (8), (9) is independent of μ.

The equations (10) define a set of constraints on the interpolation coefficients γ_i, β_i and the weights α_i in the discrete approximation to the integral that must be satisfied for the scheme to be conservative. Provided we can choose coefficients that are consistent with these constraints and that lead to accurate interpolation and integration formulae then we will have a conservative scheme. In higher dimensions a similar result holds, although the constraints become much more complicated.

To complete the proof of the theorem it remains to show that $S(t)$ approximates the integral of $\mathbf{u}$, for smooth functions $\mathbf{u}$.

$$S(t) = \sum_{k=1}^{2} \sum_{i=0}^{N_k} \alpha_{k,i} \mathbf{u}_{k,i} h_{k,i}$$

$$= \sum_{0}^{L_1-1} \mathbf{u}_{1,i} h_{1,i} + \sum_{L_1}^{L_2-1} \alpha_{1,i} \mathbf{u}_{1,i} h_{1,i} + \sum_{L_2}^{N_1} (1-\mu) \mathbf{u}_{1,i} h_{1,i}$$

$$+ \sum_{0}^{M_1-1} \mu \mathbf{u}_{2,i} h_{2,i} + \sum_{M_1}^{M_2-1} \alpha_{2,i} \mathbf{u}_{2,i} h_{2,i} + \sum_{M_2}^{N_2} \mathbf{u}_{2,i} h_{2,i}.$$

The sums that apply to points outside the interpolation regions approximate integrals over the appropriate intervals:

$$S(t) = \int_a^{x_{1,L_1}} \mathbf{u}\, dx + \sum_{L_1}^{L_2-1} \alpha_{1,i} \mathbf{u}_{1,i} h_{1,i} + (1-\mu) \int_{x_{1,L_2}}^{x_{1,N_1+1}} \mathbf{u}\, dx$$

$$+ \mu \int_{x_{2,0}}^{x_{2,M_1}} \mathbf{u}\, dx + \sum_{M_1}^{M_2-1} \alpha_{2,i} \mathbf{u}_{2,i} h_{2,i} + \int_{x_{2,M_2}}^{b} \mathbf{u}\, dx + O(h^2),$$

where $x_{k,i}$ is the position of the ith grid point on grid k. Consider the sum over the interpolation interval $[L_1, L_2 - 1]$ on component grid 1

$$S_1(L_1, L_2 - 1) = \sum_{i=L_1}^{L_2-1} \alpha_{1,i} \mathbf{u}_{1,i} h_{1,i}.$$

The variables involved in the interpolation are shown in Fig. 6.

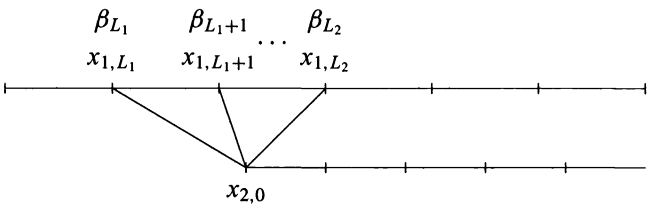


FIG. 6. *Grid variables in the interpolation region.*

Since

$$\alpha_{1,i} = 1 - \mu(\beta_{L_1} + \beta_{L_1+1} + \ldots + \beta_i) \quad \text{for } i = L_1, \ldots, L_2 - 1;$$

therefore

$$S_1(L_1, L_2 - 1) = \sum_{i=L_1}^{L_2-1} \mathbf{u}_{1,i} h_{1,i} - \mu \sum_{i=L_1}^{L_2-1} \left\{ \sum_{j=L_1}^{i} \beta_j \right\} \mathbf{u}_{1,i} h_{1,i}.$$

The first sum in this last expression approximates the integral of $\mathbf{u}$ from x_{1,L_1} to x_{1,L_2}. Changing the order of summation in the second sum gives

$$\sum_{i=L_1}^{L_2-1} \left\{ \sum_{j=L_1}^{i} \beta_j \right\} \mathbf{u}_{1,i} h_{1,i} = \sum_{i=L_1}^{L_2-1} \beta_i \left\{ \sum_{j=i}^{L_2-1} \mathbf{u}_{1,j} h_{1,j} \right\} = \sum_{i=L_1}^{L_2} \beta_i \int_{x_{1,i}}^{x_{L_2}} \mathbf{u}\, dx + O(h^2).$$

The β_i were chosen to interpolate discrete functions on the grid points $x_{L_1}, \ldots, x_{L_2}$ to obtain a value at $x_{2,0}$. In this last expression we recognize that we are interpolating the function that is equal to the integral of $\mathbf{u}$ from x to x_{L_2} and thus

$$\sum_{i=L_1}^{L_2-1} \left\{ \sum_{j=L_1}^{i} \beta_j \right\} \mathbf{u}_{1,i} h_{1,i} = \int_{x_{2,0}}^{x_{L_2}} \mathbf{u}\, dx + O(h^p),$$

where p is 1 or 2 depending on the order of accuracy of the interpolation and whence

$$S_1(L_1, L_2 - 1) = \int_{x_{1,L_1}}^{x_{1,L_2}} \mathbf{u}\, dx - \mu \int_{x_{2,0}}^{x_{1,L_2}} \mathbf{u}\, dx + O(h^p).$$

Similarly we estimate the sum $S_2(M_1, M_2 - 1)$

$$S_2(M_1, M_2 - 1) = \int_{x_{2,M_1}}^{x_{2,M_2}} \mathbf{u}\, dx - (1 - \mu) \int_{x_{2,M_1}}^{x_{1,N_1+1}} \mathbf{u}\, dx + O(h^p).$$

It follows that

$$S(t) = \int_a^b \mathbf{u}\, dx + O(h^p). \qquad \square$$

The scheme that we have constructed for one-dimensional overlapping grids satisfies the discrete local conservation property consistent with the definition of a weak solution of type I. We now show that solutions to this scheme will also converge to weak solutions of type II.

THEOREM 2.2. *If the discrete scheme for one-dimensional overlapping grids* (7,8,9) *satisfies the assumptions of Theorem* 2.1 *and the discrete solution converges as the mesh is refined, then the discrete solution converges to a weak solution of type* II (*Def.* 1.2).

Proof. Multiplying the discrete scheme, (7), by $\phi_{k,i}\, h_{k,j}$ and integrating (summing) gives

$$\int_t \sum_{k,i} \alpha_{k,i} \left\{ \phi_{k,i} \frac{d}{dt} \mathbf{u}_{k,i}\, h_{k,i} \;+\; \phi_{k,i} \Delta_{+i} \mathbf{F}_{k,i} \right\} dt \;=\; 0.$$

Integration by parts in time and summation by parts in space (proceeding in the same fashion as in the proof of Theorem 2.1) gives

$$\begin{aligned}
&\sum_{k,i} \phi_{k,i}(t_0)\mathbf{u}_{k,i}(t_0)h_{k,i} - \int_t \sum_{k,i} \frac{d\phi_{k,i}}{dt}\mathbf{u}_{k,i}h_{k,i}\, dt \\
&\quad - \int_t \Bigg\{ \alpha_{1,0}\phi_{1,0}\mathbf{F}_{1,0} + \sum_{1}^{L_1-1} \Delta_-(\alpha_{1,i}\phi_{1,i})\mathbf{F}_{1,i} + \sum_{L_1}^{L_2} \Big\{ \Delta_-(\alpha_{1,i}\phi 1, i) + \alpha_{2,0}\phi_{2,0}\beta_i \Big\} \mathbf{F}_{1,i} \\
&\qquad\qquad + \sum_{L_2+1}^{N_2} \Delta_-(\alpha_{1,i}\phi_{1,i})\mathbf{F}_{1,i} + \sum_{1}^{M_1-1} \Delta_-(\alpha_{2,i})\phi_{2,i}\mathbf{F}_{2,i} \\
&\qquad\qquad + \sum_{M_1}^{M_2} \Big\{ \Delta_-(\alpha_{2,i}\phi_{2,i}) - \alpha_{1,N_1}\phi_{1,N_1}\gamma_i \Big\} \mathbf{F}_{2,i} \;+\; \sum_{M_2+1}^{N_2} \Delta_-(\alpha_{2,i}\phi_{2,i})\mathbf{F}_{2,i} \\
&\qquad\qquad - \alpha_{2,N_2}\phi_{2,N_2}\mathbf{F}_{2,N_2+1} \Bigg\} dt \;=\; 0.
\end{aligned}$$

Using $\Delta_-(f_i g_i) = \Delta_-(f_i)\, g_i + f_{i-1}\, \Delta_-(g_i)$ gives

$$\begin{aligned}
\Delta_-(\alpha_{1,i}\phi_{1,i}) + \alpha_{2,0}\phi_{2,0}\beta_i \;&=\; \Big\{ \Delta_-(\alpha_{1,i}) + \alpha_{2,0}\beta_i \Big\} \phi_{1,i} \\
&\quad + \alpha_{1,i-1}\Delta_-(\phi_{1,i}) + \alpha_{2,0}\beta_i(\phi_{2,0} - \phi_{1,i}) \\
&=\; \alpha_{1,i-1}\Delta_-(\phi_{1,i}) + \alpha_{2,0}\beta_i(\phi_{2,0} - \phi_{1,i}).
\end{aligned}$$

Define

$$I(L_1, L_2) = \sum_{L_1}^{L_2} \alpha_{2,0}\beta_i(\phi_{2,0} - \phi_{1,i})\mathbf{F}_{1,i}$$

and note that for smooth functions ϕ of compact support, $\phi_{2,0} - \phi_{1,i} = O(h)$ and thus

$$\int_t I(L_1, L_2)dt = O(h)$$

provided $|L_2 - L_1| = O(1)$, $\alpha_{2,0} = O(1)$, and $\beta_i = O(1)$. If $\mathbf{F}_{1,i}$ converges to a smooth function then $I(L_1, L_2) = O(h^p)$, but generally we can only assume that $\mathbf{F}_{1,i}$ is bounded. Collecting terms, we get

$$\begin{aligned}
&\sum_{k,i} \phi_{k,i}(t_0)\mathbf{u}_{k,i}(t_0)h_{k,i} - \int_t \sum_{k,i} \frac{d\phi_{k,i}}{dt}\mathbf{u}_{k,i}h_{k,i}\, dt \\
&- \int_t \Bigg\{ \alpha_{1,0}\phi_{1,0}\mathbf{F}_{1,0} + \sum_{1}^{N_1} \alpha_{1,i}\Delta_-(\phi_{1,i})\mathbf{F}_{1,i} + \sum_{1}^{N_2} \alpha_{2,i}\Delta_-(\phi_{2,i})\mathbf{F}_{2,i} \\
&\qquad - \alpha_{2,N_2}\phi_{2,N_2}\mathbf{F}_{2,N_2+1} \Bigg\} dt \;+ O(h) = 0.
\end{aligned}$$

Since the test function ϕ is smooth (and of compact support so that the boundary terms disappear) and since the α_i were chosen so the sum converges to the integral, this final expression converges to

$$\int_t \int_x (\phi_t \mathbf{u} + \phi_x \mathbf{F})\, dx\, dt + \int_x \phi(x, t_0)\mathbf{u}(x, t_0)\, dx = 0$$

and thus the discrete solution converges to a weak solution of type II. □

3. Conservation in two dimensions. We now describe how to interpolate in a conservative way in two space dimensions. The algorithm we describe generalizes easily to three space dimensions. In two dimensions we consider the solution of the conservation equation

$$\mathbf{u}_t + \mathbf{f}_x + \mathbf{g}_y = 0 \tag{11}$$

on a domain Ω. The equation is conservative in the sense that the integral

$$I(t) = \int_\Omega \mathbf{u}(x, y, t) dx\, dy$$

has the property that

$$\frac{dI(t)}{dt} = \int_\Omega \frac{\partial \mathbf{u}}{\partial t} dx\, dy = -\int_\Omega (\mathbf{f}_x + \mathbf{g}_y)\, dx\, dy = -\int_{\partial\Omega} (\mathbf{f}, \mathbf{g}) \cdot \mathbf{n}\, ds.$$

The domain Ω is discretized using overlapping grids as described in Chesshire and Henshaw [7]. The domain is completely covered by a set of curvilinear grids. For each component grid $k = 1, 2, \ldots, n_g$ there is a mapping

$$\mathbf{d}_k : [0, 1]^2 \to \Omega_k$$

from the unit square to domain Ω_k, $\mathbf{x} = \mathbf{d}_k(\mathbf{r}) = (x_k(\mathbf{r}), y_k(\mathbf{r}))$. It will be convenient to introduce a related mapping for which the grid spacing in each direction will be one. Suppose component grid k is discretized with N_r and N_s grid lines in the r and s directions. Then define

$$\mathbf{D}_k(\mathbf{R}) = \mathbf{d}_k(R/(N_r - 1), S/(N_s - 1))$$

where

$$\mathbf{R} = (R, S), \quad R \in [0, N_r - 1], \quad S \in [0, N_s - 1].$$

The Jacobian of this mapping is

$$J_k = \left| \frac{\partial \mathbf{D}_k}{\partial \mathbf{R}} \right| = \frac{\partial x_k}{\partial R}\frac{\partial y_k}{\partial S} - \frac{\partial x_k}{\partial S}\frac{\partial y_k}{\partial R}.$$

We transform the conservation equation (11) on each component grid to

$$\mathbf{u}_t + \frac{1}{J_k}\left(\frac{\partial \mathbf{F}_k}{\partial R} + \frac{\partial \mathbf{G}_k}{\partial S}\right) = 0, \tag{12}$$

where

$$\begin{aligned} (\mathbf{F}_k, \mathbf{G}_k) &= \left(\frac{\partial y_k}{\partial S}\mathbf{f} - \frac{\partial x_k}{\partial S}\mathbf{g}, -\frac{\partial y_k}{\partial R}\mathbf{f} + \frac{\partial x_k}{\partial R}\mathbf{g}\right) \\ (\mathbf{f}, \mathbf{g}) &= \frac{1}{J_k}\left(\frac{\partial x_k}{\partial R}\mathbf{F} + \frac{\partial x_k}{\partial S}\mathbf{G}, \frac{\partial y_k}{\partial R}\mathbf{F} + \frac{\partial y_k}{\partial S}\mathbf{G}\right). \end{aligned} \tag{13}$$

Let $\mathbf{u}_{k,ij}$ denote the grid function values on grid k. The $\mathbf{u}_{k,ij}$ is cell-centred, while $\mathbf{F}_{k,ij}$ and $\mathbf{G}_{k,ij}$ is face-centred (Fig. 7).

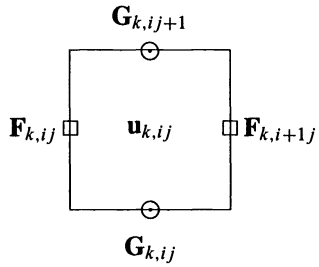


FIG. 7. *Cell variables in two dimensions.*

The discrete approximation is given by

$$\begin{aligned}
\frac{d\mathbf{u}_{k,ij}}{dt} + \frac{1}{J_{k,ij}}\left(\Delta_{+i}\mathbf{F}_{k,ij} + \Delta_{+j}\mathbf{G}_{k,ij}\right) &= 0,\\
\mathbf{F}_{k,ij} &= \mathbf{F}_N(\ldots, \mathbf{u}_{k,i-1j}, \mathbf{u}_{k,ij}, \ldots),\\
\mathbf{G}_{k,ij} &= \mathbf{G}_N(\ldots, \mathbf{u}_{k,ij-1}, \mathbf{u}_{k,ij}, \ldots).
\end{aligned}$$

The numerical fluxes are approximated by some functions $\mathbf{F}_N$ and $\mathbf{G}_N$ (N stands for numerical approximation) which for consistency should satisfy

$$\mathbf{F}_N(\ldots, \mathbf{u}, \mathbf{u}, \ldots) = \mathbf{F}_k(\mathbf{u}) \quad \text{and} \quad \mathbf{G}_N(\ldots, \mathbf{u}, \mathbf{u}, \ldots) = \mathbf{G}_k(\mathbf{u}).$$

To completely specify the discrete scheme, we need some appropriate boundary conditions at true boundaries and some interpolation conditions at the interpolation boundaries. The latter will be specified shortly.

Introduce the discrete approximation to the integral

$$S(t) = \sum_{k,ij} \alpha_{k,ij}\mathbf{u}_{k,ij}J_{k,ij}.$$

Note that by the definition of R and S, the Jacobian $J_{k,ij}$ will be approximately the area of the cell. Taking the time derivative of $S(t)$ and substituting the conservation equations gives

$$\frac{dS}{dt} = \sum_{k,ij} \alpha_{k,ij}\frac{d\mathbf{u}_{k,ij}}{dt}J_{k,ij} = -\sum_{k,ij} \alpha_{k,ij}(\Delta_{+i}\mathbf{F}_{k,ij} + \Delta_{+j}\mathbf{G}_{k,ij}).$$

Discrete integration by parts gives

$$\begin{aligned}
\frac{dS}{dt} &= \sum_{k,ij}(\Delta_{-i}(\alpha_{k,ij})\mathbf{F}_{k,ij} + \Delta_{-j}(\alpha_{k,ij})\mathbf{G}_{k,ij})\\
&\quad + \sum_{\partial\Omega_L} \alpha_{k,ij}\mathbf{F}_{k,ij} - \sum_{\partial\Omega_R} \alpha_{k,ij}\mathbf{F}_{k,i+1j} + \sum_{\partial\Omega_B} \alpha_{k,ij}\mathbf{G}_{k,ij} - \sum_{\partial\Omega_T} \alpha_{k,ij}\mathbf{G}_{k,ij+1},
\end{aligned} \tag{14}$$

where the boundary terms involve points on the Left ($\partial\Omega_L$), Right ($\partial\Omega_R$), Bottom ($\partial\Omega_B$), and Top ($\partial\Omega_T$) of the boundary cells. Some of the boundary terms will be obtained from boundary conditions on the PDE, while others will be obtained from interpolation.

We take the following approach to determine a conservative scheme.

1. All boundary terms appearing in (14) are defined using boundary conditions on the PDE or by interpolation equations. In the case of interpolation we define the flux on the interpolation boundary in terms of interior flux values on other component grids.

2. We substitute the boundary conditions and interpolation equations and collect like terms in the fluxes $(\mathbf{F}_{k,ij}, \mathbf{G}_{k,ij})$. We then set to zero the coefficients of all interior fluxes since we require that dS/dt converge to the integral of the fluxes through the boundary.

3.1. Interpolation of fluxes on overlapping grids. Certain details of the approach we take are dependent on the way in which overlapping grids are generated by the grid generation program, CMPGRD [7]. CMPGRD can generate either cell-centred or vertex-overlapping grids. For a cell-centred grid, values of $\mathbf{u}_{k,ij}$ are assumed to lie at cell-centres and interpolation is performed at cell-centres. In a vertex grid, function values lie at vertices and interpolation is done at vertices. Neither of these choices is optimal for implementing conservative interpolation since we would prefer to interpolate fluxes on cell faces. As a compromise we use a vertex grid but think of $\mathbf{u}_{k,ij}$ as being located at cell-centres. The fluxes are interpolated from the faces of one grid to the vertices of the second grid and these vertex values are then averaged to obtain face-centred values (see Fig. 8).

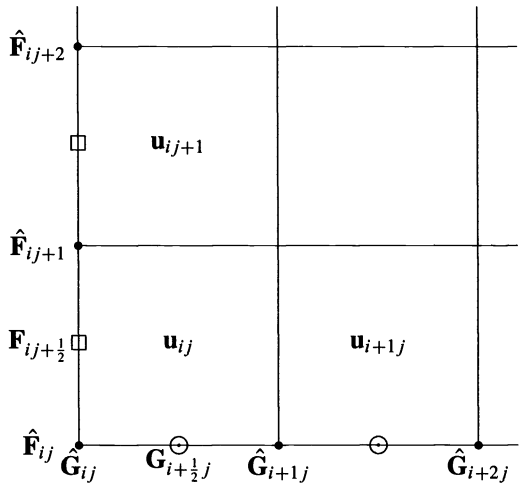


FIG. 8. *Fluxes are interpolated to vertices and then averaged to obtain values on faces.*

Suppose we wish to interpolate a point on component grid k from points on component grid k'. CMPGRD supplies the information needed to perform this interpolation, such as the points to be interpolated from and the (r, s) coordinates on grid k' of the point to be interpolated. The interpolation equations are of the form

$$\hat{\mathbf{F}}_I = \sum_{mn} \gamma^F_{k,ij,k',mn} \mathbf{F}_{k',mn}, \qquad \hat{\mathbf{G}}_I \sum_{mn} \gamma^G_{k,ij,k',mn} \mathbf{G}_{k',mn}, \tag{15}$$

where the interpolation weights γ^F and γ^G are left unspecified for the time being (see Fig. 9).

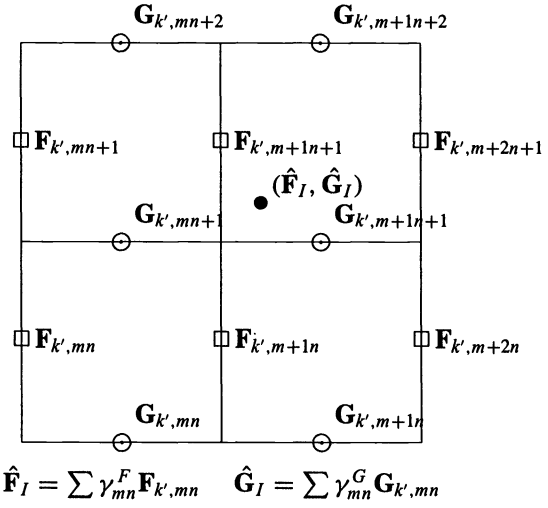


FIG. 9. *The interpolated fluxes* $(\hat{\mathbf{F}}_I, \hat{\mathbf{G}}_I)$ *for grid* k *are interpolated from face-centred fluxes of grid* k'.

The fluxes $(\hat{\mathbf{F}}_I, \hat{\mathbf{G}}_I)$ now have to be transformed from the coordinate system of grid k' to the coordinate system of grid k. This transformation is defined by (13). Introduce the matrix transformations

$$A_{k'}^{-1} = \frac{1}{J_{k'}} \begin{bmatrix} \partial x_{k'}/\partial R & \partial x_{k'}/\partial S \\ \partial y_{k'}/\partial R & \partial y_{k'}/\partial S \end{bmatrix}, \qquad A_k = \begin{bmatrix} \partial y_k/\partial S & -\partial x_k/\partial S \\ -\partial y_k/\partial R & \partial x_k/\partial R \end{bmatrix}.$$

The interpolated vertex flux is given by

$$\begin{bmatrix} \hat{\mathbf{F}}_{k,ij} \\ \hat{\mathbf{G}}_{k,ij} \end{bmatrix} = A_k A_{k'}^{-1} \begin{bmatrix} \hat{\mathbf{F}}_I \\ \hat{\mathbf{G}}_I \end{bmatrix} = A_k A_{k'}^{-1} \begin{bmatrix} \sum \gamma^F \mathbf{F}_{k'} \\ \sum \gamma^G \mathbf{G}_{k'} \end{bmatrix}. \tag{16}$$

The matrix $A_{k'}^{-1}$ is the transformation from the fluxes in grid k' coordinates to the fluxes $(\mathbf{f}, \mathbf{g})$ in physical coordinates. The matrix A_k is the transformation matrix from the physical fluxes $(\mathbf{f}, \mathbf{g})$ to the fluxes in grid k coordinates. The matrix $A_{k'}^{-1}$ should be evaluated at the same position as $(\hat{\mathbf{F}}_I, \hat{\mathbf{G}}_I)$; thus the matrix coefficients should be interpolated from the vertices.

To obtain a flux on the face, we average the values from the two adjacent vertices obtaining either

$$\mathbf{F}_{k,ij+1/2} = \frac{1}{2}\left(\hat{\mathbf{F}}_{k,ij} + \hat{\mathbf{F}}_{k,ij+1}\right) \tag{17}$$

for a vertical face or

$$\mathbf{G}_{k,i+1/2j} = \frac{1}{2}\left(\hat{\mathbf{G}}_{k,ij} + \hat{\mathbf{G}}_{k,i+1j}\right) \tag{18}$$

for a horizontal face.

Let us define some sets of indices:

1. Ω_I: Set of interior cells.
2. Ω_F: set of all faces which are used to interpolate $\hat{\mathbf{F}}_I$.
3. Ω_G: set of all faces which are used to interpolate $\hat{\mathbf{G}}_I$.
4. Ω_{BF}: Points on true boundaries that require a value for $\mathbf{F}$.
5. Ω_{BG}: Points on true boundaries that require a value for $\mathbf{G}$.

Substituting the interpolation equations defined by (15)–(18) into (14) and collecting like terms in $\mathbf{F}_{k,ij}$ and $\mathbf{G}_{k,ij}$ gives

$$\begin{aligned}\frac{dS}{dt} &= \sum_{\Omega_I}(\Delta_{-i}(\alpha_{k,ij})\mathbf{F}_{k,ij} + \Delta_{-j}(\alpha_{k,ij})\mathbf{G}_{k,ij}) \\ &+ \sum_{\Omega_F}\left(\Delta_{-i}(\alpha_{k,ij}) - \beta^F_{k,ij}\right)\mathbf{F}_{k,ij} \;+\; \sum_{\Omega_G}\left(\Delta_{-j}(\alpha_{k,ij}) - \beta^G_{k,ij}\right)\mathbf{G}_{k,ij} \\ &+ \sum_{\Omega_{BF}} \pm\alpha_{k,ij}\mathbf{F}_{k,ij} + \sum_{\Omega_{BG}} \pm\alpha_{k,ij}\mathbf{G}_{k,ij}.\end{aligned}$$

Here $\beta^F_{k,ij}$ and $\beta^G_{k,ij}$ are the sum of all terms which contribute to $\mathbf{F}_{k,ij}$ and $\mathbf{G}_{k,ij}$, respectively,

$$\beta^F_{k,ij} = \sum_{mn} c^{FF}_{k',mn,k,ij}\,\alpha_{k',mn}\,\gamma^F_{k',mn,k,ij} + c^{FG}_{k',mn,k,ij}\,\alpha_{k',mn}\,\gamma^G_{k',mn,k,ij}, \tag{19}$$

$$\beta^G_{k,ij} = \sum_{mn} c^{GF}_{k',mn,k,ij}\,\alpha_{k',mn}\,\gamma^F_{k',mn,k,ij} + c^{GG}_{k',mn,k,ij}\,\alpha_{k',mn}\,\gamma^G_{k',mn,k,ij}, \tag{20}$$

where

$$\begin{aligned}c^{FF}_{k',mn,k,ij} &= \pm K\,\{+\partial y_k/\partial S\;\partial x_{k'}/\partial R - \partial x_k/\partial S\;\partial y_{k'}/\partial R\}\frac{1}{J_{k'}}, \\ c^{FG}_{k',mn,k,ij} &= \pm K\,\{+\partial y_k/\partial S\;\partial x_{k'}/\partial S - \partial x_k/\partial S\;\partial y_{k'}/\partial S\}\frac{1}{J_{k'}}, \\ c^{GF}_{k',mn,k,ij} &= \pm K\,\{-\partial y_k/\partial R\;\partial x_{k'}/\partial R + \partial x_k/\partial R\;\partial y_{k'}/\partial R\}\frac{1}{J_{k'}}, \\ c^{GG}_{k',mn,k,ij} &= \pm K\,\{-\partial y_k/\partial R\;\partial x_{k'}/\partial S + \partial x_k/\partial R\;\partial y_{k'}/\partial S\}\frac{1}{J_{k'}}, \\ K &= \frac{1}{2} \quad \text{or } 1.\end{aligned}$$

The constant K is equal to 1 except at a boundary where it will be equal to $\frac{1}{2}$ for the following reason: The flux that is interpolated at the vertex is averaged to the adjoining faces by (17) and (18). There will normally be two faces adjoining each vertex that is interpolated, in which case the flux at the vertex is used twice and $K = 1$. When the interpolation boundary intersects a true boundary, however, the vertex that lies on the true boundary will only have one face adjacent to it, in which case $K = \frac{1}{2}$.

We require that dS/dt be equal to some approximation to the integral of the fluxes through the boundary of Ω. Thus we set to zero the coefficients of the interior fluxes. Thus the constraints on the $\alpha_{k,ij}$ are of the following types

1. For those α belonging to interior cells

$$\Delta_{-i}(\alpha_{k,ij}) = 0 \quad \text{or} \tag{21}$$

$$\Delta_{-j}(\alpha_{k,ij}) = 0 \quad \text{for } k, ij \in \Omega_I. \tag{22}$$

2. For those cells that are interpolated from

$$\Delta_{-i}(\alpha_{k,ij}) = \beta^F_{k,ij} \quad \text{for } k, ij \in \Omega_F, \tag{23}$$

$$\Delta_{-j}(\alpha_{k,ij}) = \beta^G_{k,ij} \quad \text{for } k, ij \in \Omega_G. \tag{24}$$

We will call (21)–(24) the *conservation constraints*. These equations impose conditions on the integration weights $\alpha_{k,ij}$ and the interpolation coefficients (that appear implicitly through

β^F and β^G). As long as these equations are satisfied we will have a conservative scheme. We are free to add extra equations to specify the interpolation coefficients, as long as these new equations are consistent with the conservation constraints.

3.2. Interpolee cells and the conservation constraints. In Fig. 10(a) we show an overlapping grid consisting of a rotated square (grid $k = 2$) sitting on an unrotated square (grid $k = 1$). The edge of the rotated square is interpolated from points on the unrotated square.

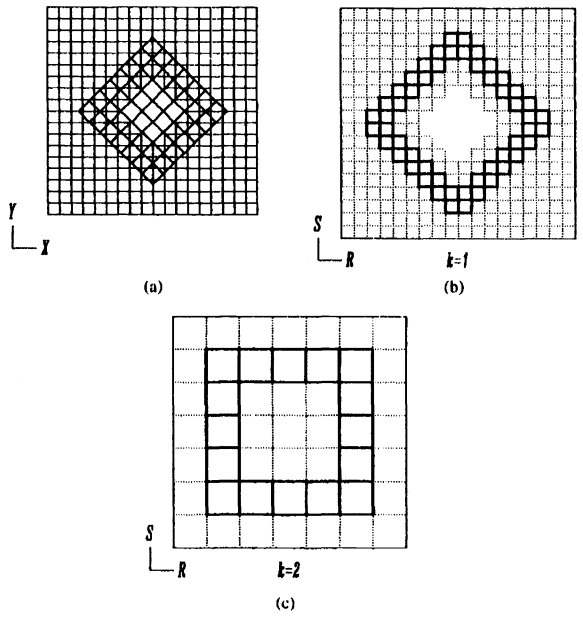


FIG. 10. *Overlapping grid and interpolee cells on each component grid.*

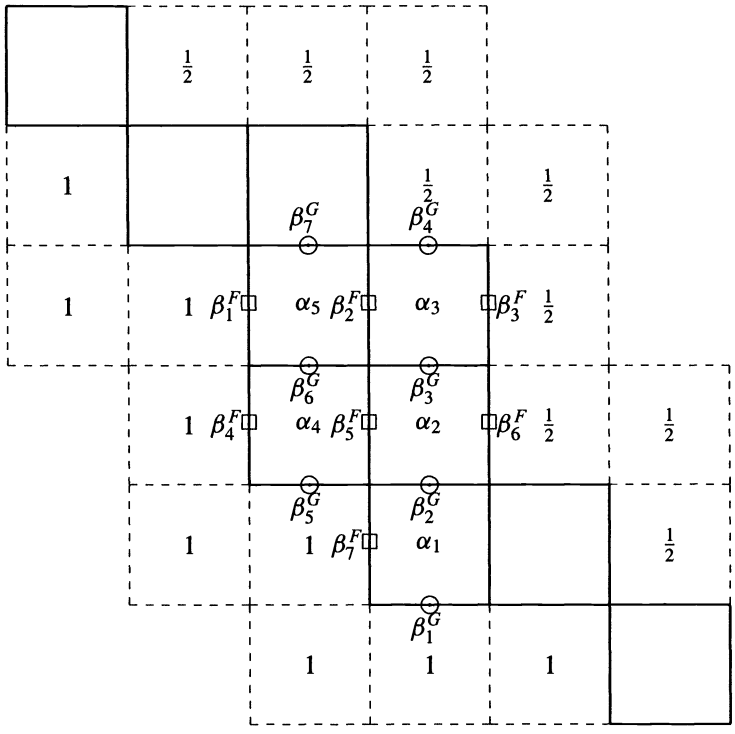
The unrotated square has points near the centre that are not used. The points on the boundary of this cut-out portion of grid 1 are interpolated from the rotated square. Since we often refer to those points or cells that are used in the interpolation of another point, we define such a point or cell to be an interpolee point or interpolee cell.

DEFINITION 3.1. *A point on grid k that is used to obtain an interpolated value on grid k′ is called an interpolee point of grid k. Cells that are completely surrounded by interpolee points are called interpolee cells.*

In Figs. 10(b) and 10(c), we show the location of the interpolee cells for component grid 1 and component grid 2 in the (R, S) unit squares. The interpolee cells are outlined with thick lines. Interpolee cells are those cells that are used by another grid for interpolation. Thus the values on the vertices of the interpolee cells on grid 1 are used to interpolate a value to the interpolation points of grid 2. Note that the set of interpolee cells partitions the grid into distinct regions. This partitioning property is a necessary condition for the existence of a conservative interpolation scheme. A cell that is not an interpolee cell will have a value of $\alpha = 1$ if it is not in an overlapping region and a value of $\alpha = \frac{1}{2}$ if it is in a region where two grids overlap. Thus the dotted cells exterior to the region bounded by the interpolee cells on grid 1 will have $\alpha = 1$ while the dotted cells inside this region will have $\alpha = \frac{1}{2}$. Conversely, the dotted cells interior to the region bounded by interpolee cells on grid 2 will have $\alpha = 1$ while outside this region $\alpha = \frac{1}{2}$. An interpolee cell may have a value for α that is different from 1 or $\frac{1}{2}$. This

value is determined in the process of determining the conservative interpolation coefficients.

In Fig. 11 we show, in detail, a set of interpolee cells. For some of these cells we have marked the unknown α's and β's that must be determined. It is assumed that the region to the upper right of the figure lies in an overlapping region and thus $\alpha = \frac{1}{2}$. The region to the lower left is a non-overlapping region and $\alpha = 1$. Recall that each interpolee cell must satisfy the conditions given by the conservation constraints, (21)–(24). At the bottom of the figure we list some of these equations that must be satisfied in order that the interpolation be conservative.



$$\alpha_4 - 1 = \beta_5^G \quad \alpha_2 - \alpha_1 = \beta_2^G \alpha_3 - \alpha_2 = \beta_3^G,$$
$$\alpha_4 - 1 = \beta_4^F \quad \alpha_2 - \alpha_4 = \beta_5^F \alpha_3 - \alpha_5 = \beta_2^F,$$
$$\tfrac{1}{2} - \alpha_2 = \beta_6^F \quad \tfrac{1}{2} - \alpha_3 = \beta_3^F,$$
$$\tfrac{1}{2} - \alpha_3 = \beta_4^G.$$

FIG. 11. *Some interpolee cells and some sample conservation constraints.*

3.3. Accuracy and smoothness of the interpolation coefficients. The enforcement of conservation constraints impose certain restrictions on the interpolation coefficients. To fully specify the interpolation coefficients it is necessary to add further constraints. These constraints can be chosen for accuracy requirements. We may also want to impose smoothness requirements on the interpolation coefficients since coefficients that have large negative and positive values do not enhance the stability characteristics of a scheme.

Accuracy. It is natural to impose conditions to increase the accuracy of the interpolation. Suppose we want to interpolate a function $f(r, s)$ at some point (r_c, s_c) in terms of the values of f at a set of points $\{f(r_{mn}, s_{mn})\}$:

$$f(r_c, s_c) \approx \sum_{mn} \gamma_{mn} f(r_{mn}, s_{mn}).$$

By Taylor's formula

$$f(r_{mn}, s_{mn}) = f(r_c, s_c) + (r_{mn} - r_c) f_r(r_c, s_c) + (s_{mn} - s_c) f_s(r_c, s_c) + (r_{mn} - r_c)^2 \frac{1}{2} f_{rr}(r_c, s_c) + (r_{mn} - r_c)(s_{mn} - s_c) f_{rs}(r_c, s_c) + \cdots$$

and thus

$$f(r_c, s_c) = \left\{ \sum_{mn} \gamma_{mn} \right\} f(r_c, s_c) + \left\{ \sum_{mn} \gamma_{mn} (r_{mn} - r_c) \right\} f_r(r_c, s_c) + \left\{ \sum_{mn} \gamma_{mn} (s_{mn} - s_c) \right\} f_s(r_c, s_c) + \left\{ \sum_{mn} \gamma_{mn} (r_{mn} - r_c)^2 \right\} \frac{1}{2} f_{rr}(r_c, s_c) + \cdots.$$

Thus to interpolate to an accuracy of $O(h^P)$ we require that the interpolation coefficients satisfy

$$\sum_{mn} \gamma_{mn} = 1 + O(h^P),$$

$$\sum_{mn} \gamma_{mn} (r_{mn} - r_c)^p (s_{mn} - s_c)^q = O(h^P) \quad \text{for } p + q < P.$$

Smoothness. There are a number of ways to impose smoothness constraints on the interpolation coefficients. Here is one reasonable way: if we think of representing the interpolation coefficients as a Fourier sine series then we can choose to make the coefficients of the highest frequencies in this series vanish. In particular, consider the coefficients $\gamma^F_{k',mn,k,ij}$ for a given interpolation point (i.e., for fixed (k', k, ij)). Then we write

$$\gamma^F_{mn} = \sum_{i=1}^{M} \sum_{j=1}^{N} \hat{\gamma}^F_{ij} \sin\left(\frac{i\,m\pi}{M+1}\right) \sin\left(\frac{j\,n\pi}{N+1}\right),$$

where for clarity we drop some of the subscripts. The Fourier coefficients $\hat{\gamma}^F_{ij}$ are given by

$$\hat{\gamma}^F_{ij} = \frac{4}{M\,N} \sum_{m=1}^{M} \sum_{n=1}^{N} \gamma^F_{mn} \sin\left(\frac{i\,m\pi}{M+1}\right) \sin\left(\frac{j\,n\pi}{N+1}\right).$$

The first smoothness constraint is obtained by setting $\hat{\gamma}^F_{MN} = 0$, followed by $\hat{\gamma}^F_{M-1N} = 0$, $\hat{\gamma}^F_{MN-1} = 0$, and so on.

3.4. Determination of the conservative interpolation coefficients. Here is an outline of the algorithm we use to compute the conservative interpolation coefficients. We work on one component grid at a time. We determine the

$$\alpha_{kij},\ \beta^F_{kij},\ \beta^G_{kij},\ \gamma^F_{k',mn,k,ij} \quad \text{and} \quad \gamma^G_{k',mn,k,ij}$$

for a given component grid k. Note that we determine the interpolation coefficients for points which interpolate from grid k as opposed to the interpolation points on grid k.

Step 1. *Mark the interpolee cells.* Mark the cells used for interpolation. Make sure that these interpolee cells divide the grid into separated regions. The algorithm for marking the interpolee cells follows.

1. For each interpolation point on grid $k' \neq k$, mark the cells on grid k which are used for interpolation.

2. For neighbouring interpolation points, check whether the set of interpolee cells used by one interpolation point is connected to the set of interpolee cells used by the second interpolation point.

3. If the sets of interpolee cells are not connected, add extra interpolation coefficients to each of the two interpolation formulae so that the interpolee cells used by the added coefficients will connect the two sets of interpolee cells.

Step 2. *Form the* β *equations.* Determine the equations for $\beta_{kij}^{F,G}$ using (19) and (20). This defines the β's in terms of the interpolation coefficients γ:

$$\begin{aligned} \beta^F &= B^F(\gamma^F, \gamma^G), \\ \beta^G &= B^G(\gamma^F, \gamma^G), \end{aligned}$$

or in matrix form,

$$\begin{bmatrix} I & 0 & A_{13} \\ & & \\ & & \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \\ \gamma \\ \vdots \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 \\ \\ \end{bmatrix}.$$

Step 3. *Conservation constraints.* Form the conservation constraints (23)–(24) for interpolee points

$$\begin{aligned} \Delta_{-i}(\alpha_{k,ij}) &= \beta_{k,ij}^F \quad \text{or} \\ \Delta_{-j}(\alpha_{k,ij}) &= \beta_{k,ij}^G \quad \text{for interpolee cells on grid } k \end{aligned}$$

with boundary conditions

$$\begin{aligned} \alpha_{k,ij} &= 1 \quad \text{in nonoverlapping regions,} \\ \alpha_{k,ij} &= \frac{1}{2} \quad \text{where two grids overlap.} \end{aligned}$$

By taking $\alpha_{k,ij} = \frac{1}{2}$ where two component grids overlap we choose to equally weight cells of the same area on any two grids. This corresponds to choosing $\mu = \nu = \frac{1}{2}$ in the one-dimensional case. If three component grids all overlap in a common region then the boundary condition would be taken as $\alpha_{k,ij} = \frac{1}{3}$. At this point we have a system of equations, which has the form

$$\begin{bmatrix} I & 0 & A_{13} \\ A_{21} & A_{22} & 0 \\ & & \\ & & \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \\ \gamma \\ \vdots \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 \\ a \\ \\ \end{bmatrix}.$$

Step 4. *Initial factor.* Factor the matrix to eliminate A_{21} and to put A_{22} into upper triangular form:

$$\begin{bmatrix} I & 0 & A_{13} \\ 0 & U_{22} & A_{23} \\ 0 & 0 & A_{33} \end{bmatrix} \begin{bmatrix} \beta \\ \alpha \\ \gamma \\ \vdots \\ \gamma \end{bmatrix} = \begin{bmatrix} 0 \\ a \\ b \end{bmatrix}.$$

After this step the β's have been eliminated. U_{22} is a square upper triangular matrix that defines the area weights α in in terms of the γ's. The rectangular matrix A_{33} multiplies the γ's and represents the constraints that must be satisfied by the γ's so that the interpolation is conservative. At this point we need only to consider the constraint equations for the γ's:

$$A_{33}x = b \quad : \quad \begin{bmatrix} A_{33} \end{bmatrix} \begin{bmatrix} \gamma \\ \vdots \\ \gamma \end{bmatrix} = \begin{bmatrix} b \end{bmatrix}. \tag{25}$$

The problem now is to choose accurate interpolation coefficients that are consistent with the above conservation constraints.

Step 5. *Interpolation accuracy and smoothness.* For each point (k', ij) interpolating from grid k there is a set of interpolation coefficients $\gamma^F_{k',ij,mn}$ and $\gamma^F_{k',ij,mn}$ defining the interpolation of the fluxes. For each interpolation formula we construct sufficiently many equations based on accuracy and smoothness to determine the interpolation coefficients completely (ignoring the fact that the coefficients will be constrained by conservation). Our choice of equations is described in §3.3. If we write the equations that determine all γ's in one large matrix equation it will have the form of a block diagonal system:

$$Gx = r \quad : \quad \begin{bmatrix} G_1 & 0 & 0 & \cdots & 0 \\ 0 & G_2 & 0 & \cdots & 0 \\ 0 & 0 & G_3 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \cdots & 0 & G_N \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_N \end{bmatrix}. \tag{26}$$

Each diagonal block G_i is square and invertible. A typical dimension for a block G_i is 6×6, corresponding to the interpolation shown in Fig. 9.

Step 6. *Solution method.* It is now necessary to find a *good* solution to the over-determined system consisting of the conservation constraints, (25), and (26) defining the interpolation coefficients in terms of accuracy and smoothness. There are various possible approaches to this problem; we present two here.

Solution method I. In method I we augment the constraint equations $A_{33}x = b$ by sufficiently many accuracy and smoothness equations from the matrix G so that the resulting system is nonsingular. This system can be solved to determine the interpolation coefficients. This method is harder to implement in practice than it sounds. The equations taken from G may be dependent on those found in A_{33} or even slightly inconsistent with those equations. Thus it is not easy to know which set of equations should be used. We have been unsuccessful at devising a stable and efficient technique for this approach and thus we have devised method II.

Solution method II. In method II we solve the constrained least squares problem

$$\min_{x\in\mathbb{R}^n} \|Gx - r\| \quad \text{subject to } A_{33}x = b. \tag{27}$$

To be specific, assume that $A_{33} \in \mathbb{R}^{m\times n}$ and $G \in \mathbb{R}^{n\times n}$. The number of constraints m is assumed to be less than the number of interpolation coefficients n. This method generates a solution that is exactly conservative but only satisfies the accuracy and smoothness equations in a least squares sense. However, by weighting the accuracy equations more than the smoothness equations it is possible, in practice, to obtain accurate interpolation equations with no large interpolation coefficients. If the interpolation is intended to be accurate to $O(h^p)$ then we scale the smoothness constraints so that the l_2 norms of the rows corresponding to the smoothness equations are $O(h^p)$. The accuracy equations, as defined by (25), are naturally scaled in the proper way. Note that if we were only interested in weighting all the equations in $Gx = r$ the same then we could replace $\min_{x\in\mathbb{R}^n} \|Gx - r\|$ in (27) with $\min_{x\in\mathbb{R}^n} \|x - G^{-1}r\|$. The solution to this new problem is somewhat simpler than the solution to (27) that is outlined below.

We can solve problem (27) as follows. Since the matrix G is nonsingular we can make the change of variables $x = G^{-1}y$ to give an equivalent formulation of the problem:

$$\min_{y\in\mathbb{R}^n} \|y - r\| \quad \text{subject to } A_{33}G^{-1}y = b.$$

Recall that G is block diagonal and thus its inverse is quickly computed. Letting $A = A_{33}G^{-1} \in \mathbb{R}^{m\times n}$ be the new constraint matrix, then our problem becomes

$$\min_{y\in\mathbb{R}^n} \|y - r\| \quad \text{subject to } Ay = b. \tag{28}$$

The transpose of A can be factored using the QR decomposition method to give

$$A^T = Q\begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $R \in \mathbb{R}^{m\times m}$ is an upper triangular matrix, which is nonsingular if A has full rank. If we partition the orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ into $Q = [Q_1\ Q_2]$, with $Q_1 \in \mathbb{R}^{n\times m}$ consisting of the first m columns, then the general solution to the underdetermined system $Ay = b$ is

$$y = Q_1(R^T)^{-1}b + Q_2 w,$$

where $w \in \mathbb{R}^{n-m}$ is arbitrary. Thus the solution of the constrained least squares problem (28) is found when w is the solution of the following least squares problem:

$$\min_{w\in\mathbb{R}^{n-m}} \|Q_2 w - (r - Q_1(R^T)^{-1}b)\|.$$

The solution to this problem is

$$w_{\text{ls}} = Q_2^T(r - Q_1(R^T)^{-1}b) = Q_2^T r,$$

since $Q_2^T Q_2 = I$ and $Q_2^T Q_1 = 0$. Therefore the constrained least squares solution y_{cls} is

$$\begin{aligned} y_{\text{cls}} &= Q_1(R^T)^{-1}b + Q_2 w_{\text{ls}} \\ &= [Q_1 Q_2]\begin{bmatrix} (R^T)^{-1}b \\ Q_2^T r \end{bmatrix} = Q\begin{bmatrix} (R^T)^{-1}b \\ Q_2^T r \end{bmatrix}, \end{aligned}$$

and the solution to (27) is

$$x_{\text{cls}} = G^{-1} y_{\text{cls}} = G^{-1} Q \begin{bmatrix} (R^T)^{-1} b \\ Q_2^T r \end{bmatrix}.$$

Although the matrix $Q \in \mathbb{R}^{n \times n}$ is large, it can be stored in an efficient manner as a product of m Householder matrices. See, for example, the discussion in the LINPACK Users' Guide [9]. The cost to compute x_{cls} is $O(m^2 n)$ floating-point operations and $O(mn)$ words of storage.

4. Numerical experiments with conservative interpolation. In this section we describe a method for discretizing conservation laws in a conservative manner on overlapping grids, and we present numerical results that show the propagation of a shock through an overlapping grid. We study a two-dimensional analogue of Burger's equation,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{29}$$

with the initial condition

$$u(x, y, 0) = u_0(x) := c - \tanh\frac{x - x_0}{2\nu}.$$

The exact solution to this problem is a shock layer moving to the right with speed c,

$$u(x, y, t) = u_0(x - ct).$$

To compute a numerical solution to this problem on the rectangle $-\frac{3}{2} \le x \le \frac{3}{2}$, $-1 \le y \le 1$, we choose $x_0 = -1$ so that the shock is initially within the domain under consideration. We use periodic boundary conditions at $y = \pm 1$ and Dirichlet boundary conditions

$$u\left(\pm\frac{3}{2}, y, t\right) = u_0\left(\pm\frac{3}{2} - ct\right)$$

at $x = \pm\frac{3}{2}$. In particular, we consider the case $c = \frac{1}{50}$ and compute the solution for $0 \le t \le 100$, so that the shock should move from $x = -1$ at $t = 0$ to $x = 1$ at $t = 100$. We discretize the problem on overlapping grids consisting of an overall uniform rectangular grid covering the entire domain and a smaller "obstacle" grid located approximately in the centre of the domain. The shock starts to the left of the obstacle at $t = 0$ and should have passed entirely through it when $t = 100$.

4.1. Discretization on a curvilinear grid. The two-dimensional Burger's equation (29) is of the form of a system of conservation laws

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) + \frac{\partial}{\partial y} g(u) = 0,$$

where the flux functions are given by

$$f(u) = f_c(u) - \nu\frac{\partial u}{\partial x}, \quad f_c(u) = \frac{1}{2}u^2, \quad g(u) = g_c(u) - \nu\frac{\partial u}{\partial y}, \quad g_c(u) = 0.$$

Before discretizing this equation on a curvilinear grid, we transform it into a coordinate system where the grid is a uniform rectangular grid,

$$\frac{\partial u}{\partial t} + \frac{1}{J}\left(\frac{\partial}{\partial r} F(u) + \frac{\partial}{\partial s} G(u)\right) = 0,$$

where

$$J = x_r y_s - y_r x_s,$$

$$F(u) = y_s f(u) - x_s g(u) = y_s f_c(u) - x_s g_c(u) + \frac{\nu}{J}\left((x_r x_s + y_r y_s)\frac{\partial u}{\partial s} - (x_s^2 + y_s^2)\frac{\partial u}{\partial r}\right),$$

$$G(u) = x_r g(u) - y_r f(u) = x_r g_c(u) - y_r f_c(u) + \frac{\nu}{J}\left((x_r x_s + y_r y_s)\frac{\partial u}{\partial r} - (x_r^2 + y_r^2)\frac{\partial u}{\partial s}\right).$$

The overlapping grid-generation program CMPGRD represents a curvilinear grid as a uniform grid on the unit square and a smooth transformation $\mathbf{d}(r, s) : [0, 1]^2 \to \mathbb{R}^2$, so that the gridpoints of the curvilinear grid are $(x_{ij}, y_{ij}) = \mathbf{d}(i\Delta r, j\Delta s)$, where $\Delta r = 1/(N_r - 1)$ and $\Delta s = 1/(N_s - 1)$. CMPGRD also provides the derivative

$$\begin{bmatrix} x_r & x_s \\ y_r & y_s \end{bmatrix}_{ij} = \frac{\partial \mathbf{d}}{\partial(r, s)}(i\Delta r, j\Delta s)$$

at the gridpoints (cell vertices). The subscript ij is assumed for x_r, x_s, y_r, and y_s wherever they appear below. To aid in the discretization, we define the averaging operators

$$\mu_{-i}u_{ij} = \frac{1}{2}(u_{i-1,j} + u_{ij}), \qquad \mu_{+i}u_{ij} = \frac{1}{2}(u_{ij} + u_{i+1,j}),$$

$$\mu_{-j}u_{ij} = \frac{1}{2}(u_{i,j-1} + u_{ij}), \qquad \mu_{+j}u_{ij} = \frac{1}{2}(u_{ij} + u_{i,j+1}),$$

the undivided difference operators

$$\Delta_{-i}u_{ij} = u_{i,j} - u_{i-1,j}, \quad \Delta_{+i}u_{ij} = u_{i+1,j} - u_{i,j}, \quad \Delta_{0i}u_{ij} = \frac{1}{2}(u_{i+1,j} - u_{i-1,j}),$$

$$\Delta_{-j}u_{ij} = u_{i,j} - u_{i,j-1}, \quad \Delta_{+j}u_{ij} = u_{i,j+1} - u_{i,j}, \quad \Delta_{0j}u_{ij} = \frac{1}{2}(u_{i,j+1} - u_{i,j-1}),$$

and the cell-centred quantities

$$A_{ij} = \mu_{+i}\mu_{+j}\left(\frac{1}{\Delta r \Delta s J_{ij}}\right), \qquad B_{ij} = \mu_{+i}\mu_{+j}\left(\frac{x_r x_s + y_r y_s}{J_{ij}}\right).$$

A_{ij} is an approximation to second-order accuracy of the reciprocal of the area of the cell (i, j). Then we discretize equation (29) in space to second-order accuracy as

$$\frac{du_{ij}}{dt} + A_{ij}(\Delta_{+i}F_{ij} + \Delta_{+j}G_{ij}) = 0, \tag{30}$$

where

$$\begin{aligned} F_{ij} = {} & \Delta s[(\mu_{+j}y_s)(\mu_{-i}f_c(u_{ij})) - (\mu_{+j}x_s)(\mu_{-i}g_c(u_{ij}))] \\ & + \nu\left\{(\mu_{-i}B_{ij})(\mu_{-i}\Delta_{0j}u_{ij}) - \Delta s^2(\mu_{-i}A_{ij})[(\mu_{+j}x_s)^2 + (\mu_{+j}y_s)^2](\Delta_{-i}u_{ij})\right\}, \end{aligned} \tag{31}$$

$$\begin{aligned} G_{ij} = {} & \Delta r[(\mu_{+i}x_r)(\mu_{-j}g_c(u_{ij})) - (\mu_{+i}y_r)(\mu_{-j}f_c(u_{ij}))] \\ & + \nu\left\{(\mu_{-j}B_{ij})(\mu_{-j}\Delta_{0i}u_{ij}) - \Delta r^2(\mu_{-j}A_{ij})[(\mu_{+i}x_r)^2 + (\mu_{+i}y_r)^2](\Delta_{-j}u_{ij})\right\}. \end{aligned} \tag{32}$$

4.2. Discretization on an overlapping grid. To discretize (29) on an overlapping grid, we must supply interpolation boundary conditions in the regions of overlap between grids. The program CMPGRD is designed to generate overlapping grids where the solution u_{ij} is defined either at cell-centres or at cell vertices. In either case it guarantees that all eight neighbours of any "discretization point" are either discretization points themselves or they are "interpolation points" where function values may be interpolated from the corresponding function values at discretization points of another grid. We have implemented conservative interpolation of fluxes using "cell-vertex overlapping grids," as explained in §3. On such a composite grid, CMPGRD labels the vertices as either discretization points, interpolation points, or unused points. We discretize the problem with u_{ij} represented at cell-centres and the fluxes represented on the sides of each cell, as shown in Fig. 12. The computation of F_{ij}

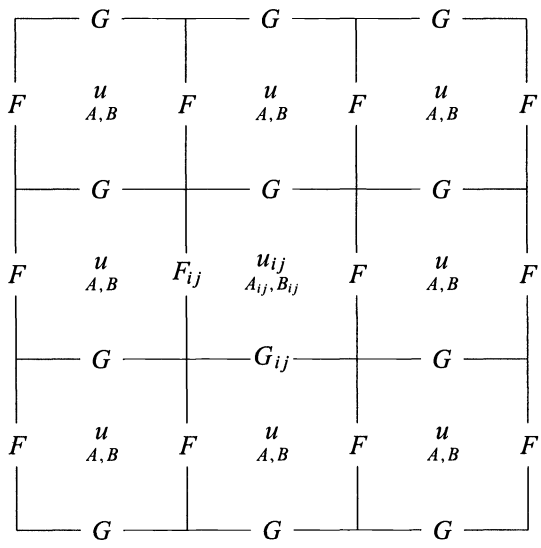


FIG. 12. *Grid functions.*

uses the stencil consisting of $u_{i-1,j}$, $u_{i-1,j\pm1}$, u_{ij} and $u_{i,j\pm1}$. Suppose u is known at every cell-centre that is surrounded by vertices all of which are labelled as either discretization points or interpolation points. CMPGRD guarantees that all eight neighbours of a discretization point are either discretization points or interpolation points. Therefore we can compute F_{ij} on any cell side that connects two discretization points (i, j) and $(i, j+1)$. Similarly, we can compute G_{ij} on any cell side that connects two discretization points (i, j) and $(i+1, j)$. We can update u_{ij} using (30) anywhere that we can compute the expression for du_{ij}/dt; for this, we need values for F_{ij}, $F_{i+1,j}$, G_{ij}, and $G_{i,j+1}$. We know how to compute these fluxes for any cell surrounded by discretization points. To update u in a cell, one or more of whose neighbouring vertices is an interpolation point, we must first interpolate the fluxes that we could not otherwise compute. We must interpolate the flux on each cell side adjoining two vertices, at least one of which is labelled as an interpolation point and the other is labelled as either an interpolation point or a discretization point. We interpolate these fluxes by the procedure explained in §3.1.

We use the classical fourth-order accurate four-stage Runge–Kutta timestepping method. We choose the timestep to be close to the stability limit, which we determined experimentally. In particular, we use

$$\Delta t = \min\{.8h, .3\frac{h^2}{\nu}\}, \quad \text{where } h = \min\{\sqrt{x_r^2 + y_r^2}\Delta r, \sqrt{x_s^2 + y_s^2}\Delta s\}.$$

The minimum here is taken over all gridpoints on all component grids. In the numerical examples we took $\nu = .01$, which resulted in sharp shocks with little or no overshoot.

4.3. Numerical results. Figure 13 shows the numerical solution at time $t = 0, 20, \ldots, 100$ for the case $c = \frac{1}{50}$ of a slowly-moving shock, obtained using an overlapping grid

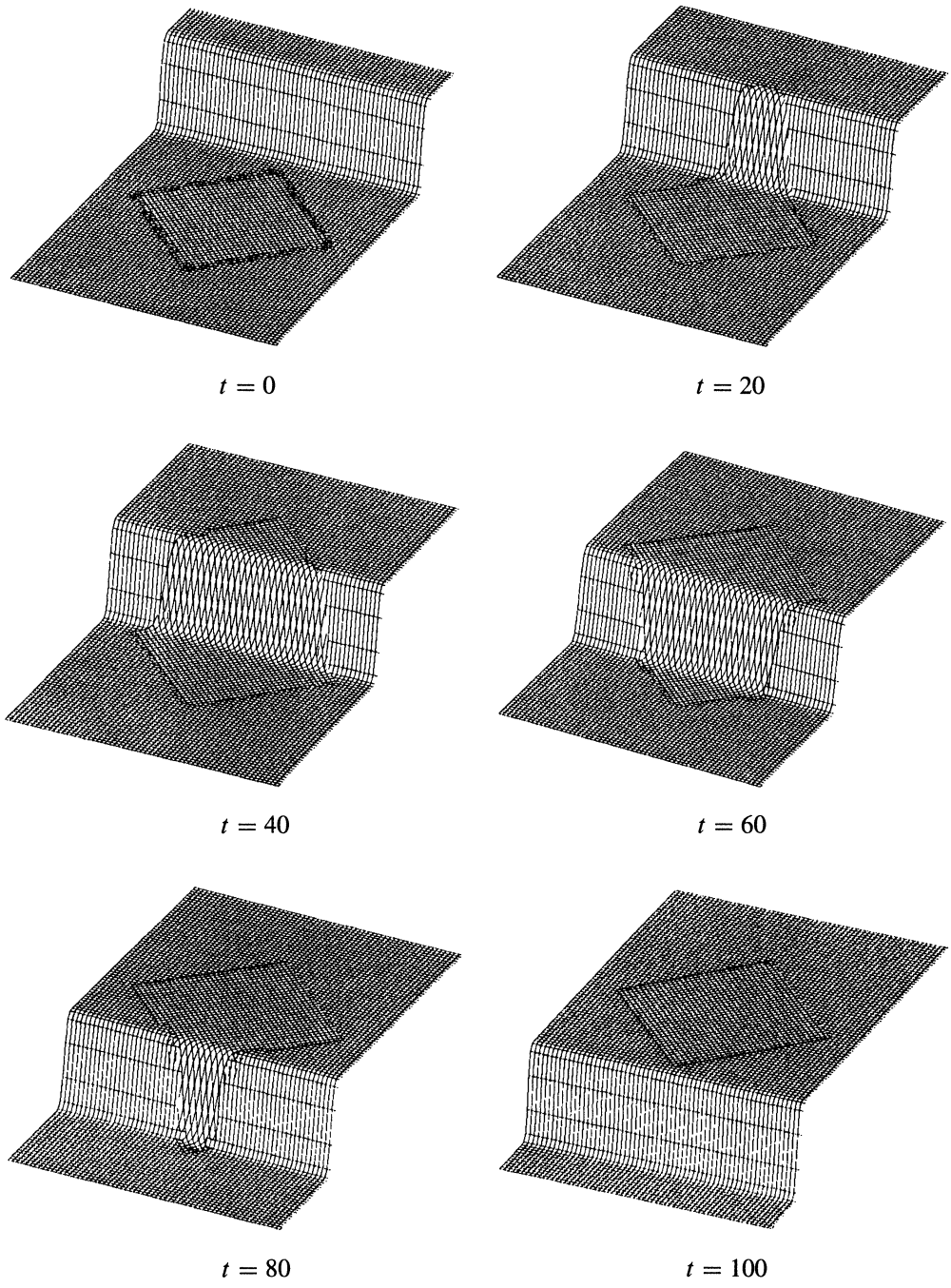


FIG. 13. *Numerical solution at times* $t = 0, 20, \ldots, 100$.

consisting of an overall rectangular grid of 90 by 60 square cells and a square grid in the centre of the region, with sides of length 1.2, rotated 45°, with 36 × 36 cells of the same size as those of the overall grid. The numerical solution shows an undistorted shock whose location is correct to within the width of one grid cell at all times. Figure 14 is a graph of the

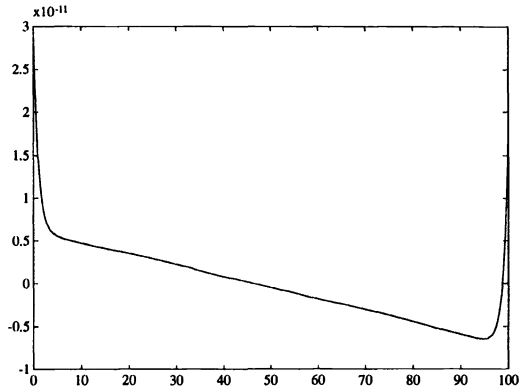


FIG. 14. *Conservation error as a function of time.*

difference between the conserved discrete integral of u and the analytic integral of the exact solution. The mean value of this conservation error, 3.26×10^{-6}, has been subtracted out, as this error is only due to truncation error in the discrete integral of the initial condition. The resulting conservation error is less than 3×10^{-11} at all times. It changes most rapidly at $t = 0$ and $t = 100$, when the shock is nearest to the domain boundaries at $x = \pm\frac{3}{2}$. This effect is due to the truncation error in the discretization of the boundary conditions and not to the interpolation of fluxes between grids.

5. Conclusions. We have described a scheme for interpolating in a conservative manner when solving systems of conservation laws on overlapping grids. The technique is described in one and two space dimensions; the extension to three dimensions is straightforward. We consider the interpolation coefficients and the integration weights in the conserved integral as free parameters. The condition that the scheme be conservative imposes constraints on these parameters. We have shown how to derive these constraints and how to determine accurate interpolation coefficients consistent with the conservation constraints. This requires the solution of a system of equations with number of unknowns essentially equal to the number of interpolation coefficients on one component grid. We have written a code to generate the conservative interpolation coefficients for two-dimensional grids generated by the overlapping grid generation program, CMPGRD.

We have numerically solved a two-dimensional viscous Burger's equation using conservative interpolation and have demonstrated the exact conservative nature of the method. In future work we hope to solve the full Navier–Stokes or Euler equations using the conservative interpolation and to study the problem of slowly moving shocks passing through the interface between grids.

Acknowledgment. The authors are grateful for valuable discussions with Professor Heinz-Otto Kreiss.

REFERENCES

[1] M. J. BERGER, *Stability of interfaces with mesh refinement*, Math. Comp., 45 (1985), pp. 301–318.
[2] ———, *On conservation at grid interfaces*, SIAM J. Numer. Anal., 24 (1987), pp. 967–984.

[3] M. J. BERGER AND J. OLIGER, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys., 53 (1984), pp. 484–512.

[4] D. L. BROWN, *A finite volume method for solving the Navier–Stokes equations on composite overlapping grids*, in Third International Conference on Hyperbolic Problems, B. Engquist and B. Gustafsson, eds., Chartwell-Bratt, Lund. Sweden, 1991, pp. 141–158.

[5] D. L. BROWN, G. CHESSHIRE, AND W. D. HENSHAW, *Getting started with* CMPGRD, *introductory user's guide and reference manual*, report LA-UR-89-1294, Los Alamos National Laboratory, Los Alamos, NM, 1989.

[6] P. G. BUNING, I. T. CHIU, S. OBAYASHI, Y. M. RIZK, AND J. L. STEGER, *Numerical simulation of the integrated space shuttle vehicle in ascent*, paper 88-4359-CP, AIAA, 1988.

[7] G. CHESSHIRE AND W. D. HENSHAW, *Composite overlapping meshes for the solution of partial differential equations*, J. Comput. Phys., 90 (1990), pp. 1–64.

[8] ———, *Conservation on Composite Overlapping Grids*, IBM Research Report RC 16531, IBM Research Division, Yorktown Heights, NY, 1991.

[9] J. J. DONGARRA, C. B. MOLER, J. R. BUNCH, AND G. W. STEWART, *LINPACK Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1978.

[10] W. D. HENSHAW AND G. CHESSHIRE, *Multigrid on composite meshes*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 914–923.

[11] P. LAX AND B. WENDROFF, *Systems of conservation laws*, Comm. Pure and Appl. Math, 13 (1960), pp. 217–237.

[12] E. PÄRT-ENANDER AND B. SJÖGREEN, *Solving the Euler Equations for High Mach Numbers on Overlapping Grids*, Tech. Report 114, Dept. of Scientific Computing, Uppsala University, Sweden, 1988.

[13] ———, *Shock Waves and Overlapping Grids*, Tech. Report 131, Dept. of Scientific Computing, Uppsala University, Sweden, 1991.

[14] M. M. RAI, *A Conservative Treatment of Zonal Boundaries for Euler Equation Calculations*, paper A84-17923, AIAA, 1987.

[15] L. G. M. REYNA, *Part* III. *On Composite Meshes*, Ph.D. thesis, Dept. of Applied Mathematics, California Institute of Technology, Pasadena, 1982.

[16] G. STARIUS, *On composite mesh difference methods for hyperbolic differential equations*, Numer. Math., 35 (1980), pp. 241–255.

[17] J. L. STEGER AND J. A. BENEK, *On the use of composite grid schemes in computational aerodynamics*, Comput. Meth. Appl. Mech. Engrg., 64 (1987), pp. 301–320.

[18] M. THUNÉ, *Stability of a Runge–Kutta method for the Euler equations on a substructured domain*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 154–174.

[19] L. N. TREFETHEN, *Stability of finite-difference models containing two boundaries or interfaces*, Math. Comp., 45 (1985), pp. 279–300.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 846–865, July 1994

005

HIGH ORDER ACCURACY OPTIMIZED METHODS FOR CONSTRAINED NUMERICAL SOLUTIONS OF HYPERBOLIC CONSERVATION LAWS*

C. CORAY† AND J. KOEBBE†

Abstract. A high order accuracy generalization of accuracy optimized methods (AOMs) for the numerical solution of scalar hyperbolic conservation is described. This process includes a presentation of a general framework for the construction of high order accurate base schemes that are linearly stable, consistent, and conservative. The AOM defines and solves a quadratic programming problem at each discrete time level to minimize perturbations from the high order base schemes subject to imposed constraints. The constraints are used to imposed desired behavior on the numerical approximation to the solution of the conservation law. The resulting schemes retain the high order accuracy of the base scheme away from shocks, and minimally perturb the high order base schemes locally where necessary to meet the imposed constraints. The resulting schemes compare favorably with other high resolution schemes for scalar conservation laws. Numerical examples are presented to illustrate convergence rates for the high order methods, stability regions, and AOM results for linear advection of discontinuities and development and transport of shocks in Burgers' equation. The constraints used in this work lead to a systematic method for construction of high order accurate total variation diminishing (TVD) schemes.

Key words. scalar conservation laws, accuracy optimization, total variation diminishing, high resolution

AMS subject classifications. 65M06, 76M20

1. Introduction.

The problems of limiting numerical diffusion, resolution of shock fronts, maintaining high order accuracy, and preventing spurious oscillations that arise in numerically approximating the solution of scalar hyperbolic conservation laws of the form

$$u_t + f(u)_x = 0, \quad t > 0, \quad x \in \Re, \tag{1}$$

$$u(x, 0) = u_0(x)$$

have long been a concern of physicists, engineers, and mathematicians. It is not possible to maintain, for example, monotonicity of a numerical solution near a shock front with an unmodified higher order accurate method. One only needs to examine the second order method of Lax and Wendroff [6] to find a scheme that may introduce spurious oscillations for a variety of initial conditions even in the linear advection problem. Many authors have contributed significantly to this general topic, including Boris and Book [1], Harten [3], Osher [8], Osher and Chakravarthy [9], Van Leer [11], Zalesak [12], Sweby [10], and others. For general reference, a recent comprehensive overview of the subject of numerical methods for conservation laws has been published by LeVeque [7].

In [2] a different approach, via schemes called AOMs, was introduced in which second order base schemes were modified locally through an optimization process to satisfy imposed constraints. The optimization process preserves second order accuracy whenever possible and allows the scheme to reduce locally to first order accuracy if needed. The purpose in this paper is to generalize the AOM approach to include higher order base schemes. The base schemes are constructed in a systematic way so that accuracy, linear stability, flux consistency, and conservation are satisfied. The construction allows efficient local modification of the base scheme within the AOMs.

*Received by the editors March 11, 1992; accepted for publication June 24, 1993.

†Department of Mathematics and Statistics, Utah State University, Logan, Utah 84322 (coray@avatar.math.usu.edu, koebbe@avatar.math.usu.edu).

There are many physical or mathematical reasons for imposing constraints on approximate solutions of scalar conservation laws, e.g., a requirement that the scheme be monotonicity preserving. The AOM minimally perturbs the higher order base scheme where necessary to meet imposed constraints. This process defines at each timestep a constrained optimization problem with a simple sum of squares objective function. The nature of the optimization problem and the conditions placed upon the base scheme guarantee the existence of a global solution. We note here that the methods constructed in this paper are more expensive computationally than other methods, but the results and the ability to control the degree of accuracy of the approximation in regions where the solution is smooth justify the investigation of the AOM framework.

The AOM is built to treat general constraints on the numerical solution. In this paper the constraint addressed will be that the AOM is monotonicity preserving. The sufficient condition prescribed to guarantee that property was shown in [2] to also be sufficient for the AOM to be TVD as well. In fact, the framework developed here allows the systematic construction of arbitrarily high order TVD schemes. By this we mean the scheme is high order accurate where the solution is smooth.

A somewhat surprising and positive result is that although the AOM approach with the monotonicity preserving constraints is entirely motivated by a minimal perturbation of a higher order method where necessary, the resolution of shock fronts with this method compares favorably with the high resolution methods presented in Sweby [10] and others. The AOM approach to modify high order base schemes locally to meet constraints is significantly different from other traditional TVD schemes, e.g., flux limited schemes, in which limited amounts of "antidiffusive" flux are added locally.

The solution of the optimization problem adds to the computational effort necessary to numerically approximate the solution of (1). However, in [2] the optimization problem was shown to be a local problem in the linear advection case. In all numerical experiments performed on nonlinear problems to date the optimization problem again appears to be local in nature. In such cases the smaller disjoint optimization problems can be solved in parallel.

In §2 we review the fundamental properties of the AOMs, including the optimization criteria and methodology used to achieve approximate solutions that satisfy imposed constraints while minimally perturbing higher order accurate base schemes. In §3 an overview is presented of a systematic way to construct higher order base schemes for (1). In §4 a condition on the numerical solution, which will guarantee that the method is both monotonicity preserving and TVD, is introduced as well as the resulting inequality that becomes part of the AOM optimization problem. Section 5 contains numerical results illustrating the AOM with higher order base schemes.

2. Accuracy optimized methods. In this section we describe AOMs in general terms. We begin by considering (1) and we write a numerical scheme to approximate the solution to (1) in conservation form

$$u_j^{n+1} = u_j^n - \lambda\{f(u_{j+\frac{1}{2}}^n) - f(u_{j-\frac{1}{2}}^n)\}, \tag{2}$$

where $u_{j\pm 1/2}^n$ are grid line values to be determined below to impose specified conditions on the discretization. The spatial mesh is defined by nodes x_j and grid lines $x_{j+1/2}$, t_n, and t_{n+1} represent discrete time levels, Δx and Δt are the grid sizes in space and time, $\lambda = \frac{\Delta t}{\Delta x}$, and $u_j^n = u(x_j, t_n)$.

Before application of the AOM a base scheme is constructed. Values $\tilde{u}^n_{j\pm 1/2}$ given by

$$\tilde{u}^n_{j\pm\frac{1}{2}} = m^n_{j\pm\frac{1}{2}}(\alpha^n_{j\pm\frac{1}{2}}) \tag{3}$$

are used in the base scheme

$$\tilde{u}^{n+1}_j = \tilde{u}^n_j - \lambda\{f(\tilde{u}^n_{j+\frac{1}{2}}) - f(\tilde{u}^n_{j-\frac{1}{2}})\}. \tag{4}$$

The $m^n_{j\pm 1/2}$ are functions that depend on the initial profile, $u(x, t_n)$, and $\alpha^n_{j\pm 1/2}$ are evaluation points of the functions used to determine appropriate values of u at the grid lines. The AOM then modifies the grid line values $\tilde{u}^n_{j\pm 1/2}$ locally to meet imposed constraints. That is, the AOM generates the values $u^n_{j\pm 1/2}$ used in the discretization (2) by modifying $\tilde{u}^n_{j\pm 1/2}$. A general method for construction of base schemes is presented in the next section. The form of $m^n_{j\pm 1/2}$ and evaluation points $\alpha^n_{j\pm 1/2}$ are chosen to determine a base scheme that satisfies the four properties:

1. high order space and time accuracy,
2. linear stability,
3. flux consistency, and
4. conservation of u.

In many applications other properties of a numerical method are desirable. Two such examples are

5. achieving high resolution at discontinuities, and/or
6. satisfying some physical or mathematical constraint on the approximate solution, e.g., monotonicity preserving or TVD. Conflicts arise when a numerical method attempts to accomplish all objectives simultaneously. Godunov's Theorem, for example, limits linear monotonicity preserving schemes to first order accuracy. On the other hand unmodified high order accurate methods such as the second order Lax–Wendroff method often produce spurious oscillations near steep solution profiles.

The AOM approach relaxes the first property locally to achieve properties such as the last two. This is accomplished by solving an associated optimization problem. Specifically, the AOM uses the modification

$$u^n_{j\pm\frac{1}{2}} = m^n_{j\pm\frac{1}{2}}(\eta^n_{j\pm\frac{1}{2}}), \tag{5}$$

where $\eta^n_{j\pm\frac{1}{2}} = \alpha^n_{j\pm\frac{1}{2}} + \epsilon^n_{j\pm\frac{1}{2}}$ are modified evaluation points and $\epsilon_{j\pm\frac{1}{2}}$ represent perturbations of the base scheme evaluation points. The AOM minimizes

$$\sum(\epsilon^n_{j+\frac{1}{2}})^2 \tag{6}$$

subject to imposed constraints such as Properties 5 or 6. With the restrictions we will place on the grid line functions and their points of evaluation this gives a well posed constrained quadratic programming problem.

While the optimization problem is solved over the entire domain at each timestep in this paper, it appears the problem can be broken into small, localized optimization problems on those disjoint regions where the unmodified base scheme would violate the imposed constraints. In [2] the localization of the optimization problem was shown in the linear advection problem. On those disjoint regions (generally where steep solution profiles occur) the smaller optimization problems may be solved in parallel.

We conclude this section with the remark that this methodology is different from that of the flux-limiters described by Sweby [10] and others. Here we seek to "optimize" accuracy by varying minimally from a high order method subject to constraints imposed to preserve desirable properties on the approximation of the solution rather than adding a maximal amount of "antidiffusive" flux. We are now ready to address the details of construction of higher order base schemes and examples of imposed constraints within AOM.

3. Construction of higher order base methods. In §2 the AOM introduced requires the construction of a base scheme that will be perturbed locally to satisfy imposed constraints. In this section the higher order accurate base schemes are constructed using polynomial representations at the grid lines as in [5]. After first fixing the evaluation point of the polynomial, the coefficients of the polynomial are chosen so that accuracy, linear stability, flux consistency, and conservation conditions are satisfied for any smooth solution. It should be noted that the polynomial framework discussed below is an alternate way of looking at the construction of numerical methods for scalar conservation laws. The motivation for using this framework is twofold. First, the first four properties of a base scheme may be developed in an efficient algorithmic way, and second, the required perturbations for the AOM are easily imposed on a base scheme within the framework.

The conditions developed here are for the linear conservation law with $f(u) = au$ and $a > 0$. The conservation form of the discretization in this case is

$$u_j^{n+1} = u_j^n - a\lambda(u_{j+\frac{1}{2}}^n - u_{j-\frac{1}{2}}^n), \tag{7}$$

with

$$u_{j\pm\frac{1}{2}}^n = m_{j\pm\frac{1}{2}}^n(\alpha_{j\pm\frac{1}{2}}^n) = \sum_{i=0}^{l} a_{i,j\pm\frac{1}{2}}(\alpha_{j\pm\frac{1}{2}}^n)^i$$

chosen as the representation of the unknown at the grid lines $x_{j\pm1/2}$. The tilde used to denote the base scheme value in the previous section has been dropped here for convenience of notation. The evaluation point $\alpha_{j\pm1/2}^n$ has been scaled so that the distance upwind increases with α. In the linear advection case,

$$\alpha_{j\pm\frac{1}{2}} = \frac{1}{2} + a\lambda.$$

Here it is assumed that the grid is uniform and thus the $\frac{1}{2}$ represents the scaled distance from the downwind node to the grid line where the polynomial is defined. Within this framework it is essential to choose the following representation for the coefficients:

$$a_{i,j-\frac{1}{2}}^n = \sum_{r=1}^{K_i}(b_{i,j-\frac{1}{2},r}u_{j-1+r}^n + b_{i,j-\frac{1}{2},-r}u_{j-r}^n). \tag{8}$$

The values $b_{i,j\pm1/2,\pm r}$ are the nodal multipliers that must be determined to completely specify the method. The upper limit K_i on the sum is the stencil width of the coefficients $a_{i,j\pm\frac{1}{2}}^n$, and thus determines the number of nodes upon which the coefficients $a_{i,j\pm1/2}^n$ depend. Define K as the maximum stencil width over all coefficients. The form in (8) may look cumbersome, but is invaluable in determining appropriate accuracy and stability conditions for the method.

Before proceeding with the development of the conditions on the nodal multipliers $b_{i,j\pm 1/2,\pm r}$ for specifying base schemes, we note that the computations to obtain the conditions for accuracy, linear stability, and flux consistency below are long and tedious and most are not included here. For a thorough development of the conditions on the multipliers see [5].

The desired accuracy for the method is imposed via a standard truncation error analysis on the discretization of the conservation law in (7) with

$$u_{j+r}^n = \sum_{m=0}^{\infty} \left(\frac{\partial u}{\partial x}\right)\Big|_j^n (r\Delta x)^m.$$

Assume that integer spatial accuracy M_x and time accuracy of M_t are desired. Equating coefficients in the expansion yields a system of equations of the form

$$\begin{aligned}\delta_{m,k} = \sum_{i=k}^{l} \beta_{i,k} \sum_{r=1}^{K_i} [&b_{i,j+\frac{1}{2},r} r^m + b_{i,j+\frac{1}{2},-r}(1-r)^m \\ &- b_{i,j-\frac{1}{2},r}(r-1)^m - b_{i,j-\frac{1}{2},-r}(-r)^m],\end{aligned} \tag{9}$$

where $\delta_{m,k}$ is the Kronecker delta function, $\beta_{i,k} = \binom{i}{k}(\frac{1}{2})^{i-k}$, $m = 0, \ldots, M_x$ and $k = 1, \ldots, M_t$. The above expression yields $(M_x + 1)M_t$ linear equations for the multipliers $b_{i,j\pm 1/2,\pm r}$. Note that the maximum degree of accuracy is limited by the degree of the polynomial and the stencil width of the coefficients, $a_{i,j\pm 1/2}^n$. In particular, the maximum degree of accuracy is $l + 1$.

The second condition in the determination of the nodal multipliers is linear stability. By doing a standard von Neumann stability analysis with

$$u_j^n = \xi^n e^{i\omega x_n},$$

the following condition can be obtained:

$$\begin{aligned}\xi = 1 - \sum_{k=0}^{l} (a\lambda)^k \sum_{i=k}^{l} \beta_{i,k} \sum_{r=1}^{K_i} \\ \times (b_{i,j+\frac{1}{2},r} e^{ir\theta} + b_{i,j+\frac{1}{2},-r} e^{i(1-r)\theta} + b_{i,j-\frac{1}{2},r} e^{i(r-1)\theta} + b_{i,j-\frac{1}{2},-r} e^{i(r)\theta}),\end{aligned}$$

where $\theta = \omega\Delta x$. Separating the real and imaginary parts gives

$$\begin{aligned}R(\theta) = 1 - \sum_{k=0}^{l} (a\lambda)^k \sum_{i=k}^{l} \beta_{i,k} \sum_{r=1}^{K_i} \\ \times [(b_{i.j+\frac{1}{2},r} - b_{i,j-\frac{1}{2},-r})\cos(r\theta) + (b_{i.j+\frac{1}{2},-r} - b_{i,j-\frac{1}{2},r})\cos((1-r)\theta)]\end{aligned}$$

and

$$\begin{aligned}I(\theta) = \sum_{k=0}^{l} (a\lambda)^k \sum_{i=k}^{l} \beta_{i,k} \sum_{r=1}^{K_i} \\ \times [(b_{i.j+\frac{1}{2},r} + b_{i,j-\frac{1}{2},-r})\sin(r\theta) + (b_{i.j+\frac{1}{2},-r} + b_{i,j-\frac{1}{2},r})\sin((1-r)\theta)].\end{aligned}$$

Next consider the elementary solution

$$u(x,t) = e^{-i\omega(x-at)}$$

for the linear partial differential equation. The ratio of the solution at successive timesteps is

$$\frac{u(x,t+\Delta t)}{u(x,t)} = e^{i\omega a\Delta t} = e^{ia\lambda\theta}.$$

Ideally ξ should represent $e^{ia\lambda\theta}$. Conditions sufficient for linear stability at the point $a\lambda$ are as follows:

$$\cos(a\lambda\theta) = R(\theta), \qquad \sin(a\lambda\theta) = I(\theta).$$

The important point here is that the expressions for the real and imaginary parts of ξ are truncated Fourier cosine and sine series that may be used to represent the real and imaginary parts of the elementary solution. To determine the coefficients in the Fourier series, the standard formulas for the coefficients may be used. Thus on the interval $[-\pi,\pi]$, the stability conditions become

$$\int_{-\pi}^{\pi} \cos(a\lambda\theta)\cos(m\theta)d\theta = \int_{-\pi}^{\pi} R(\theta)\cos(m\theta)d\theta$$

and

$$\int_{-\pi}^{\pi} \sin(a\lambda\theta)\sin(n\theta)d\theta = \int_{-\pi}^{\pi} I(\theta)\sin(n\theta)d\theta$$

for $m = 0,\ldots,K$ and $n = 1,\ldots,K$, where $K = \max_{0\le i\le l} K_i$.

After the integration is carried out the conditions due to the real part are

$$\frac{2\sin(a\lambda\pi)}{a\lambda} = 2\pi\left[1 - \sum_{i=0}^{l}\sum_{k=0}^{i} \beta_{i,k}(a\lambda)^k (b_{i,j+\frac{1}{2},-1} - b_{i,j-\frac{1}{2},1})\right] \tag{10}$$

for $m = 0$;

$$-\frac{2(a\lambda)\cos(m\pi)\sin(a\lambda\pi)}{m^2-(a\lambda)^2} = -\pi\sum_{i=0}^{l}\sum_{k=0}^{i}\beta_{i,k}(a\lambda)^k \times[(b_{i,j+\frac{1}{2},m} - b_{i,j-\frac{1}{2},-m}) + (b_{i,j+\frac{1}{2},-m-1} - b_{i,j-\frac{1}{2},m+1})] \tag{11}$$

for $0 < m < K$; and

$$-\frac{2(a\lambda)\cos(K\pi)\sin(a\lambda\pi)}{K^2-(a\lambda)^2} = -\pi\sum_{i=0}^{l}\sum_{k=0}^{i}\beta_{i,k}(a\lambda)^k (b_{i,j+\frac{1}{2},K} - b_{i,j-\frac{1}{2},-K}) \tag{12}$$

for $m = K$. For the imaginary part the conditions are

$$\frac{2m\cos(m\pi)\sin(a\lambda\pi)}{m^2-(a\lambda)^2} = -\pi - \sum_{i=0}^{l}\sum_{k=0}^{i}\beta_{i,k}(a\lambda)^k \times[(b_{i,j+\frac{1}{2},m} + b_{i,j-\frac{1}{2},-m}) - (b_{i,j+\frac{1}{2},-m-1} + b_{i,j-\frac{1}{2},m+1})] \tag{13}$$

for $1 \leq m < K$; and

$$\frac{2K\cos(K\pi)\sin(a\lambda\pi)}{K^2-(a\lambda)^2} = -\pi - \sum_{i=0}^{l}\sum_{k=0}^{i}\beta_{i,k}(a\lambda)^k(b_{i,j+\frac{1}{2},K} + b_{i,j-\frac{1}{2},-K}) \tag{14}$$

for $m = K$.

The result of the conditions given in (10)–(14) is a system of $2K+1$ linear equations in the multipliers $b_{i,j\pm 1/2,\pm r}$. These equations allow for the input of a desired stability limit, $a\lambda$, since this parameter appears explicitly in all the equations resulting from the stability conditions. In Theorem 1 of [4], a condition on the relationship between stability and accuracy is stated that is applicable to the framework developed here. If the discretization (7) is written in the form

$$u_j^{n+1} = \sum_{k=-r}^{s} c_j u_{jk}^n,$$

then this theorem implies the maximum order of accuracy p of a stable scheme is given by

$$p = \min\{r+s, 2r+2, 2s\}.$$

In the polynomial framework, if a system of equations is formed that attempts to violate the above condition for the maximum order of accuracy, the system of equations is necessarily inconsistent.

Next, flux consistency is imposed on the numerical scheme. Flux consistency for a discretization of the form

$$u_j^{n+1} = u_j^n - \lambda(h_{j+\frac{1}{2}}^n - h_{j-\frac{1}{2}}^n)$$

implies that the approximate flux function $h_{j\pm 1/2}^n = h_{j\pm 1/2}^n(u_{j-l}^n, \ldots, u_{j+r}^n)$ satisfies

$$h_{j\pm\frac{1}{2}}^n(u, \ldots, u) = f(u).$$

Since in the polynomial framework $h_{j\pm 1/2}^n = f(u_{j\pm 1/2}^n)$ and $m_{j\pm 1/2}^n(\alpha) = m_{j\pm 1/2}^n(u_{j-K}, \ldots, u_{j+K}, \alpha)$, the flux consistency condition reduces to the following condition on the evaluation of the polynomial:

$$u = m_{j\pm\frac{1}{2}}^n(u, \ldots, u, \alpha).$$

If flux consistency is to hold for all possible values of $a\lambda$, and thus all values of α, the condition above becomes

$$1 = \sum_{i=0}^{l}\beta_{i,0}\sum_{r=1}^{K_i}(b_{i,j+\frac{1}{2},r} + b_{i,j+\frac{1}{2},-r}) \tag{15}$$

and

$$0 = \sum_{i=k}^{l}\beta_{i,k}\sum_{r=1}^{K_i}(b_{i,j+\frac{1}{2},r} - b_{i,j+\frac{1}{2},-r}) \tag{16}$$

for $1 \leq k \leq l$. These conditions are given at the grid line located at $x_{j+1/2}$ and there are similar conditions at the grid line located at $x_{j-1/2}$. Thus the flux consistency requirement on the discretization will generate a total of $2(l+1)$ linear conditions on the multipliers, $b_{i,j\pm 1/2,\pm r}$.

If the base scheme is forced to be conservative then the perturbed scheme in the AOM will also be conservative due to the way the perturbation value $\epsilon_{j+1/2}$ is chosen. Conservation of u requires that the polynomial centered at each grid line must be the same. This implies $b_{i,j+1/2,\pm r} = b_{i,j-1/2,\pm r}$, which reduces the number of unknowns by a factor of two.

In summary, the framework presented in this section provides an algorithmic way for determination of higher order accurate base schemes to be used as input for AOM. Thus this step can be thought of as initial overhead in the application of the AOM. The framework lends itself readily to automation. A computer code to generate the multipliers, $b_{i,j\pm 1/2,\pm r}$, has been written and used in this work.

4. AOM example: monotonicity preserving constraints. With a base scheme developed as in the previous section to meet the first four objectives, we are now ready to address Properties 5 and 6 from §2 in the AOM. A monotonicity preserving constraint is introduced to illustrate an example of an imposed constraint that fits neatly under the AOM umbrella. If we assume a decreasing profile, i.e., $u_j^n \leq u_{j-1}^n$, then a sufficient condition for preserving monotonicity would be to require

$$u_j^n \leq u_j^{n+1} \leq u_{j-1}^n. \tag{17}$$

If the initial profile is increasing, the inequalities are reversed. With the basic discretization (2), the monotonicity preserving inequality becomes

$$0 \leq -\lambda\{f(u_{j+\frac{1}{2}}^n) - f(u_{j-\frac{1}{2}}^n)\} \leq u_{j-1}^n - u_j^n. \tag{18}$$

For general f the base scheme uses evaluation points

$$\alpha_{j\pm\frac{1}{2}}^n = \frac{1}{2} + \lambda f'(u_{j\pm\frac{1}{2}}^n). \tag{19}$$

With (3) this implies that the grid line values $u_{j\pm 1/2}^n$ are defined implicitly. In general, this implicit definition requires an iterative loop at each timestep to solve for the $u_{j\pm 1/2}^n$. If one makes reasonable assumptions about the smoothness of f, then a fixed point iteration will converge for small enough λ (Koebbe [5]). We note that this computation lends itself readily to massively parallel computer architecture. As discussed earlier, the AOM scheme redefines the evaluation points to be

$$\eta_{j\pm\frac{1}{2}}^n = \alpha_{j\pm\frac{1}{2}}^n + \epsilon_{j\pm\frac{1}{2}}^n, \tag{20}$$

where the $\epsilon_{j\pm 1/2}^n$ represent perturbations of the base scheme evaluation points. The optimization portion of the AOM minimizes the sum of squares of such perturbations subject to meeting constraints such as (18) imposed at each node x_j.

A consequence of the consistency condition is that the polynomial representation of the approximation interpolates the initial condition at upwind nodes. The upwind interpolation implies that the optimization problem has a feasible solution. This feasibility condition has the additional property that it permits the AOM to become first order accurate in local regions where shocks or steep profiles occur. This reduction to first order must be attainable by the AOM to satisfy Godonov's Theorem [7].

Another important class of methods are those that are TVD. The total variation $\mathrm{TV}(u^n)$ of the numerical solution at time level n is defined by

$$\mathrm{TV}(u^n) = \sum_j |u^n_{j+1} - u^n_j|. \tag{21}$$

A difference scheme is said to be TVD (after Harten [3]) if $\mathrm{TV}(u^{n+1}) \leq \mathrm{TV}(u^n)$. It was shown in [2] that the sufficient condition (17) for a monotonicity preserving scheme is also sufficient for the AOM to be TVD. Thus the nonlinear constraint equation (18) serves equally well to guarantee that the AOM solution that results from application of this inequality is also TVD. Thus the AOMs in this paper form a class of TVD schemes whose order of accuracy in smooth regions can be specified.

5. Numerical results. In this section numerical results are presented to demonstrate properties of the methods constructed in the paper. The numerical results are split into two parts. First, numerical convergence rates are presented for a sample of base methods generated in the polynomial framework introduced in §3. In particular, several methods up through order six are presented. In the second part, results of computations using AOMs on a variety of initial data for both linear advection and Burgers' equations are presented.

5.1. Convergence rates for the higher order schemes. To compute convergence rates for the base methods for the AOM approach, the linear advection equation

$$u_t + u_x = 0, \qquad x \in \Re, t > 0$$

with the initial condition $u(x, 0) = \sin(4\pi x)$ was used. This problem provides the basis for the computation of convergence rates against the analytic solution

$$u(x, t) = \sin(4\pi(x - t)), \qquad x \in \Re, t > 0.$$

All methods were run on meshes with 20, 40, 80, and 160 nodes on the unit interval and errors were computed on these meshes with the analytic solution. Numerical convergence rates were then computed between successive refinements in the mesh.

Three methods were tested in the work; second, third, and fourth order methods generated by linear, quadratic, and cubic polynomial representations, respectively. The base methods were generated using a computer code written to form and solve a linear system of equations as defined in §3. The polynomial generation computer code requires the degree of the polynomial, stencil width of the coefficients, and an imposed stability limit as input. The flux consistency and conservation conditions are implicitly imposed on the nodal multipliers. The code outputs the nodal multipliers for use in another computer code that provides approximations of the solution of (1).

In most cases the solution of the system of linear equations is not unique. There are a variety of ways to specify the extra degrees of freedom. In this work the extra conditions necessary for unique determination of a method were obtained by setting as many downwind nodes as possible to zero.

The second order method generated by a polynomial of degree one, coefficient stencil width one, and stability limit one is specified by the multipliers:

$$b_{0,\cdot,-1} = \tfrac{1}{4}, \quad b_{0,\cdot,1} = \tfrac{3}{4},$$

$$b_{1,\cdot,-1} = \tfrac{1}{2}, \quad b_{1,\cdot,1} = -\tfrac{1}{2}.$$

The third order method that was generated by a polynomial of degree two, coefficient stencil width of two, and stability limit of two is specified by the nodal multipliers:

$$b_{0,\cdot,-2} = -\tfrac{1}{8}, \quad b_{0,\cdot,-1} = \tfrac{1}{2}, \quad b_{0,\cdot,1} = \tfrac{5}{8}, \quad b_{0,\cdot,2} = 0,$$

$$b_{1,\cdot,-2} = -\tfrac{1}{6}, \quad b_{1,\cdot,-1} = \tfrac{5}{6}, \quad b_{1,\cdot,1} = -\tfrac{2}{3}, \quad b_{1,\cdot,2} = 0,$$

$$b_{2,\cdot,-2} = -\tfrac{1}{6}, \quad b_{2,\cdot,-1} = \tfrac{5}{6}, \quad b_{2,\cdot,1} = -\tfrac{2}{3}, \quad b_{2,\cdot,2} = 0.$$

Finally, the fourth order method was generated using a polynomial of degree three, coefficient stencil width of four, and imposed stability limit of three are specified by the nodal multipliers:

$$b_{0,\cdot,-3} = 0.078125, \quad b_{0,\cdot,-2} = -0.359375, \quad b_{0,\cdot,-1} = 0.734375,$$
$$b_{0,\cdot,1} = 0.546875, \quad b_{0,\cdot,2} = 0.0, \quad b_{0,\cdot,3} = 0.0,$$

$$b_{1,\cdot,-3} = 0.078125, \quad b_{1,\cdot,-2} = -0.359375, \quad b_{1,\cdot,-1} = 0.734375,$$
$$b_{1,\cdot,1} = -0.739583, \quad b_{1,\cdot,2} = 0.0, \quad b_{1,\cdot,3} = 0.0,$$

$$b_{2,\cdot,-3} = -0.145833, \quad b_{2,\cdot,-2} = 0.604166, \quad b_{2,\cdot,-1} = -0.770833,$$
$$b_{2,\cdot,1} = 0.3125, \quad b_{2,\cdot,2} = 0.0, \quad b_{2,\cdot,3} = 0.0,$$

$$b_{3,\cdot,-3} = 0.041666, \quad b_{3,\cdot,-2} = -0.125, \quad b_{3,\cdot,-1} = 0.125,$$
$$b_{3,\cdot,1} = -0.041666, \quad b_{3,\cdot,2} = 0.0, \quad b_{3,\cdot,3} = 0.0.$$

As an aside, both the second order methods of Lax–Wendroff and Warming and Beam can be generated in this framework by using a linear polynomial representation, stability limit one, and stencil widths of one and two, respectively. When the maximum number of downwind multipliers is set to zero the two methods are specified.

Tables 1–3 present the convergence rates at several timesteps during the numerical simulation between 20 and 40 nodes, 40 and 80 nodes, and 80 and 160 nodes for the three methods just described. In all three cases the numerical convergence rates are nearly the values predicted in §3. The values of the Courant number λ shown in the tables were chosen to avoid stability problems in the higher order methods. There are no problems with the second order method, but some care must be taken when specifying stability limits larger than one in this framework. To explain the stability problems, we return to the material in §3.

The stability condition specified in (10)–(14) can be used to compute the square of the amplification factor $|\xi|^2$ as a function of the Courant number $a\lambda$. Since the stability conditions are truncated Fourier series approximations to a function, oscillations should be expected in the graphs of these functions. The three methods used in this section were specified so that the stability limit was one, two, and three for the second, third, and fourth order schemes, respectively. The graphs of the square of the amplification factor for the methods are shown in Fig. 1 for the third and fourth order methods. The interval shown in each of the graphs

TABLE 1

Table of computational convergence rates for the second order method generated by a linear polynomial. This is equivalent to the Lax–Wendroff method.

Time level	20 vs. 40 nodes $\lambda = 0.25$	20 vs. 40 nodes $\lambda = 0.75$	40 vs. 80 nodes $\lambda = 0.25$	40 vs. 80 nodes $\lambda = 0.75$	80 vs. 160 nodes $\lambda = 0.25$	80 vs. 160 nodes $\lambda = 0.75$
1	1.98720	1.94955	1.99684	1.98759	1.99926	1.99690
2	1.97268	1.99377	1.99648	1.98739	1.99921	1.99690
3	1.98120	1.94614	1.99610	1.98719	1.99914	1.99687
4	1.97498	1.97838	1.99570	1.98694	1.99910	1.99685
5	1.97456	1.94157	1.99531	1.98670	1.99905	1.99683
6	1.97660	1.96194	1.99490	1.98643	1.99901	1.99679
7	1.96727	1.93586	1.99447	1.98613	1.99895	1.99676
8	1.97758	1.94448	1.99404	1.98583	1.99890	1.99673
9	1.95934	1.92903	1.99360	1.98549	1.99885	1.99670
10	1.97791	1.92600	1.99314	1.98514	1.99881	1.99667

TABLE 2

Table of computational convergence rates for the third order method generated by a quadratic polynomial.

Time level	20 vs. 40 nodes $\lambda = 0.8$	20 vs. 40 nodes $\lambda = 1.8$	40 vs. 80 nodes $\lambda = 0.8$	40 vs. 80 nodes $\lambda = 1.8$	80 vs. 160 nodes $\lambda = 0.8$	80 vs. 160 nodes $\lambda = 1.8$
1	2.94321	2.92056	2.99570	2.99413	2.99831	3.00334
2	2.97921	2.98200	2.99632	2.98629	2.99660	3.00396
3	2.92898	2.98047	2.98524	2.98181	2.99913	3.00398
4	2.97869	2.95941	2.98262	2.99528	2.99653	3.00055
5	2.95773	2.99334	2.99076	2.99663	2.99810	3.00467
6	2.92921	2.93840	2.99375	2.99809	2.99910	3.00150
7	2.96581	3.00823	2.99464	2.99032	2.99589	3.00144
8	2.91623	3.00141	2.98343	2.98542	2.99863	3.00322
9	2.96500	2.98869	2.98090	2.99902	2.99612	2.99867
10	2.94465	3.01680	2.98901	3.00049	2.99765	3.00284

TABLE 3

Table of computational convergence rates for the fourth order method generated by a cubic polynomial.

Time level	20 vs. 40 nodes $\lambda = 1.8$	20 vs. 40 nodes $\lambda = 2.85$	40 vs. 80 nodes $\lambda = 1.8$	40 vs. 80 nodes $\lambda = 2.85$	80 vs. 160 nodes $\lambda = 1.8$	80 vs. 160 nodes $\lambda = 2.85$
1	3.95455	3.92188	3.99371	3.98827	4.00148	3.98775
2	3.92539	3.92257	3.98466	3.98381	3.99255	3.99728
3	3.92393	3.96499	3.98596	3.98020	3.99764	3.99723
4	3.95253	3.94493	3.99199	3.99026	3.99827	3.99695
5	3.98112	3.91417	3.97793	3.97714	3.99443	3.99460
6	3.95306	3.95785	3.99205	3.98850	4.00042	3.99421
7	3.92491	3.96727	3.98488	3.98392	3.99730	3.99626
8	3.91970	3.92521	3.98511	3.98078	3.99821	3.99781
9	3.94922	3.95009	3.99204	3.99059	3.99648	3.99564
10	3.97865	3.98903	3.97802	3.97733	3.99654	3.99399

is slightly larger than the stability limit specified for the method so that the behavior can be seen for the entire region of interest for the method. The graph of the second order method is not included since it is well known that the Lax–Wendroff method, which is equivalent to the second order method presented, is stable for $0 \leq a\lambda \leq 1$. Note that in the methods of order higher than two, there are certain regions where the schemes are unstable. In the tests for convergence rate, values of the Courant number were chosen based on these graphs to stay away from unstable regions.

The breakdown of stability in certain regions is not a desirable behavior for any method. In simple cases such as linear advection the stability problem can be avoided by choosing appropriate timesteps using graphs like those in Figures 1 and 2. However, if the stability

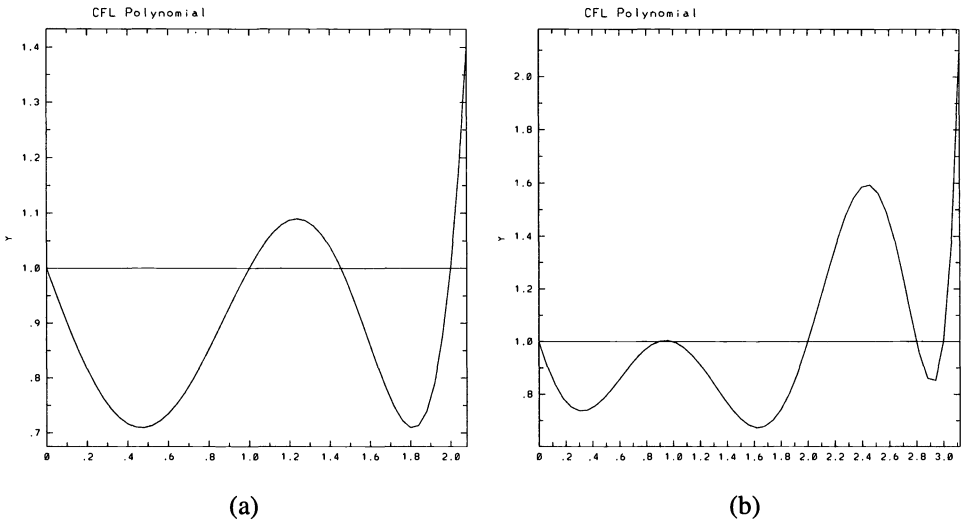


FIG. 1. *Graphs of the square of the amplification factor as a function of the Courant number for* (a) *the third order method generated by a quadratic polynomial and* (b) *a fourth order method generated by a cubic polynomial. The amplification factor is larger than one for* $a\lambda$ *slightly less than one in* (b).

region of the base method is disconnected, then the feasible region in the optimization problem is not convex and standard sequential quadratic programming algorithms (e.g., NAG routine E04UPF) cannot be applied. Thus methods that guarantee that the scheme stays stable over some range of the Courant number must be used. Since it is necessary to allow the AOM to reduce to upwinding, the interval [0, 1] for the Courant number must be included in the stability region. These methods can be generated by reducing the stencil width and the stability limit imposed. The third order method proposed above generates a stable method provided that $0 \leq a\lambda \leq 1$. Thus one could choose Δt such that $0 \leq a\lambda \leq 1$. Figure 2 shows the square of the amplification factor for another fourth order method and a sixth order method where the stencil width has been reduced. Both of these methods will also be stable for $0 \leq a\lambda \leq 1$. Note that for the original fourth order method presented, the amplification factor is larger than one in a narrow region where the Courant number is slightly less than one. This problem does not occur in the fourth order method of Fig. 2(a). The importance of

stability over an entire interval for the AOM is that this guarantees a convex feasible region for the optimization problem and, if the stability interval contains all $a\lambda$ in [0, 1], a solution exists for the optimization problem since this guarantees that the method can locally reduce to upwinding.

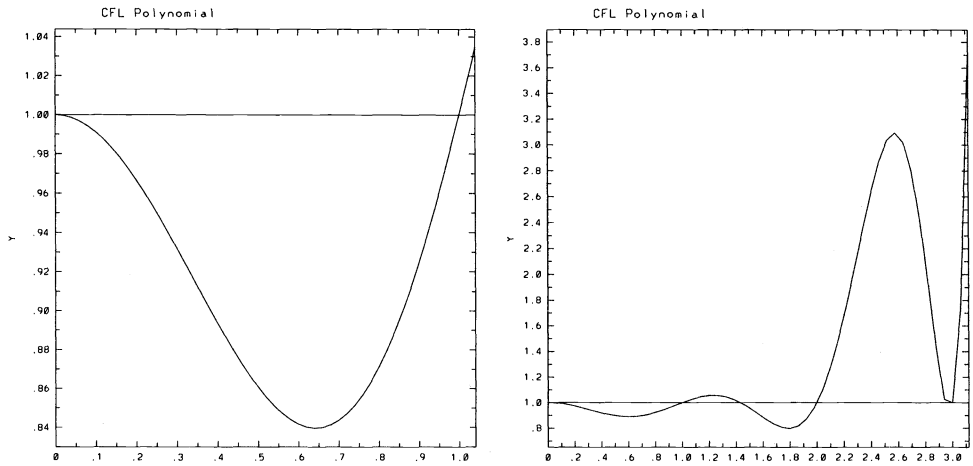


FIG. 2. *Graph of the square of the amplification factor as a function of the Courant number for* (a) *a different fourth order method than the one used to generate Fig.* 1(b) *and* (b) *a sixth order method with stability limit one.*

With these results we can now move on to applying the AOM to higher order methods constructed in the polynomial framework.

5.2. AOM results. In this section results are presented to demonstrate the use of AOM on linear advection problems and Burgers' equation. That is, we treat (1) with $f(u) = u$ or $f(u) = \frac{1}{2}u^2$. In this work the higher order base schemes that were constructed for the linear advection problem will be applied directly in the nonlinear problems presented. One can show that the polynomial form used in the second order base scheme will also produce a second order scheme for a general nonlinear conservation law; see [2]. A more complete discussion of the AOM applied to general nonlinear problems will be the focus of another paper.

Several types of initial data were used to show different behavior in the numerical solution using AOM. Comparisons were done with first order upwinding and an unmodified second order method to show the improvement of the AOM over diffusive first order methods and the elimination of oscillations generated by higher order schemes. Consider the following initial data:

1. $u(x, 0) = 1$ for $x \leq 0.5$ and $u(x, 0) = 0$ for $x > 0.5$;
2. $u(x, 0) = 1$ for $0.25 \leq x \leq 0.5$ and $u(x, 0) = 0$, otherwise;
3. $u(x, 0) = |\sin(4\pi x)|, x \in \Re, t > 0$.

Figure 3 shows results from simulation of Problem 1 in the case of linear advection and where the third order method generated by a quadratic polynomial was used. Figures 3(a) and 3(b) show the approximate unknown values for 80 nodes and 160 nodes after 40 and 80 steps, respectively, with $\lambda = 0.25$. Figures 3(c) and 3(d) show the perturbations necessary at

the given timestep when the AOM is applied. There are two interesting types of behavior in the approximate solutions. First, a steep shock is observed that becomes steeper as the grid is refined. The AOM was not constructed as a high resolution method, but the result is a class of schemes that resolve steep profiles.

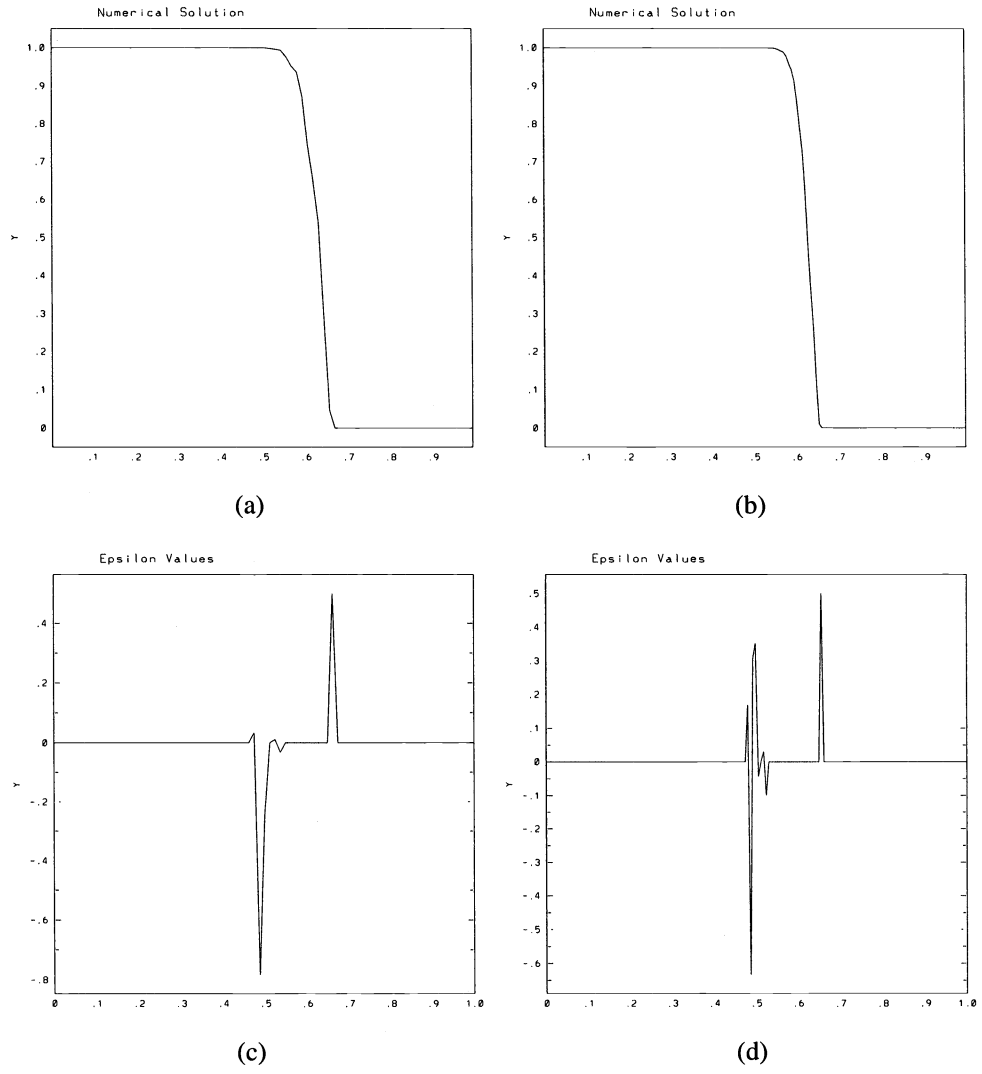


FIG. 3. *Transport of a shock in the case of linear advection using a third order method generated by a quadratic polynomial. The approximations using* (a) 80 *nodes and* (b) 160 *nodes along with the corresponding perturbations* (c) *and* (d) *are shown.*

The second behavior is the difference between the way that the scheme treats the discontinuity upwind and downwind of the shock. Even though a steep profile is observed, the width of the shock region as measured by the interval where the perturbations are nonzero is not

decreased as the grid is refined (Figs. 3(c) and 3(d)). In fact, after 40 and 80 time steps the perturbations are still necessary at the original position of the shock. The problem is that the schemes constructed in this paper are asymmetric in the sense that they are dissipative upwind. This does not mean that all AOMs will have this type of asymmetric behavior. In [2] the same problem was encountered when the second order methods of Lax–Wendroff and Warming and Beam were used as base schemes individually. The use of the Lax–Wendroff method as the base scheme resulted in an AOM that was dissipative downwind of a discontinuity, while use of Warming and Beam as the base scheme produced an AOM that is dissipative upwind of the discontinuity. To take advantage of both of the schemes, an optimized combination of the two schemes was used that produced a symmetric AOM that was less dissipative than either of the individual methods.

Figure 4 shows the results of a simulation of the transport of a square wave in Problem 2 by the AOMs produced by using Lax–Wendroff, Warming and Beam, and a combination of the two schemes as base schemes. The results in Figs. 4(a) and 4(b) show the asymmetric dissipation downwind using Lax–Wendroff and upwind using Warming and Beam, respectively. Figure 4(c) shows the symmetric results of the AOM from the combination of the two schemes. In [2] the symmetric method just described was compared to the flux limited method of Roe as presented in [10]. The AOM shock resolution width was one half that of the Super-Bee flux limited method. The same ideas can be applied to the more general class of AOMs constructed here and will be the topic of a paper addressing the relationships between AOM and essentially nonoscillatory (ENO) schemes.

As an example, computational results using a higher order symmetric AOM are shown in Fig. 5. This AOM was constructed using an optimized combination of two distinct third order methods. Figures 5(a) and 5(b) show the same results as Figs. 3(a) and 3(b) except that half as many nodes were used; 40 nodes and 80 nodes. The number of nodes was divided by two so that the number of variables in the optimization problems was the same in both cases; that is, the optimized combination of methods requires twice as many unknowns that are evenly divided between the modifications of the two base schemes. The improvement in the shock resolution is obvious. The width of the shock region is reduced *and* the support of the perturbations as shown in Figs. 5(c) and 5(d) moves along with the shock instead of increasing in size as in the asymmetric case (see Fig. 3).

As a final test result for this paper, Burgers' equation with the initial condition of Problem 3 was solved using the AOM with $\lambda = 0.25$, 99 nodes and a total of 100 timesteps. Figure 6 shows the results of the simulation at various timesteps. Figure 6(a) shows the initial condition, Fig. 6(b) shows the approximation just before the expected shock develops, Fig. 6(c) shows the approximation as the shock is forming, and Fig. 6(d) shows the approximation after the shock has fully formed and has begun to move. The "lip" in Fig. 6(c) shows that the shock is forming in a nonlinear fashion, which is expected due to the form of the initial condition.

As a comparison, the same simulation was performed using first order upwinding and an unmodified second order method generated by the linear polynomial. The results are shown in Fig. 7 for the same time level as the results shown in Fig. 6(d). The results show that the first order method does not resolve the shock very well and dissipates the cusp at the bottom more than the AOM results in Fig. 6. The unmodified higher order method fails near the shock as expected; however, the second order method that fails in Fig. 7(b) was used as the base method to obtain the better results in Fig. 6.

The solution of an optimization problem at each step over the entire spatial domain at each timestep is of concern in the computational expense of the AOMs. However, as noticed in [2], the optimization problem can be localized to those regions where the imposed constraints are

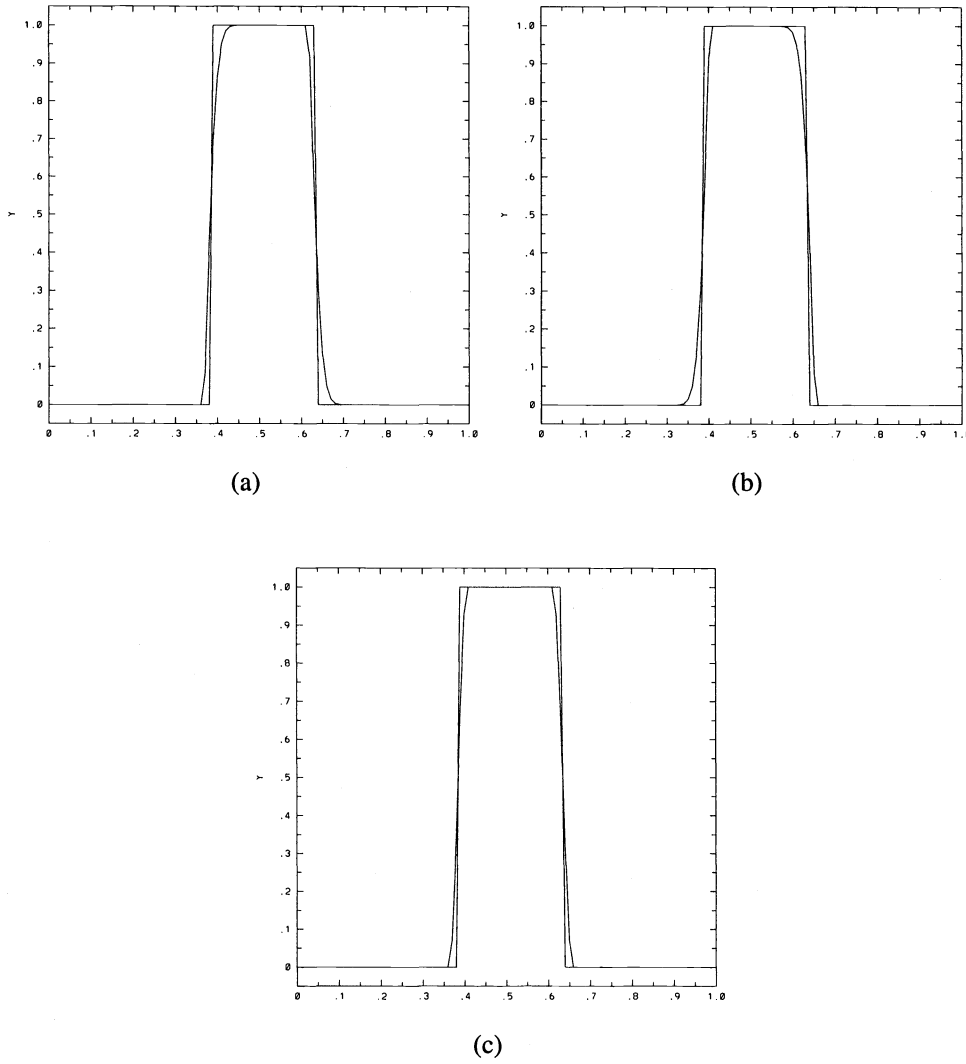


FIG. 4. *Transport of the square wave in the case of linear advection using* (a) AOM *based on Lax–Wendroff*, (b) AOM *based on Warming and Beam, and* (c) AOM *based on a combination of Lax–Wendroff and Warming and Beam.*

not satisfied by the base scheme for linear advection and Burgers' equation. The numerical results presented here also demonstrate this fact. In Figs. 3(c) and 3(d), Figs. 5(c) and 5(d), and Fig. 8(b) the regions where the perturbation ϵ is nonzero are small bands at the beginning and end of the shock region. Fig. 8(a) can be used to see that the optimization is only needed where the gradients become steep. The optimization problem can be split into two smaller optimization problems that could be solved in parallel. Thus it appears that the optimization problem need not consider the entire domain and the number of variables in the optimization problem is reduced significantly. In the last test problem the same localization occurs. Figure 6(d) shows the graph of the perturbations over the entire domain. Again the nonzero values

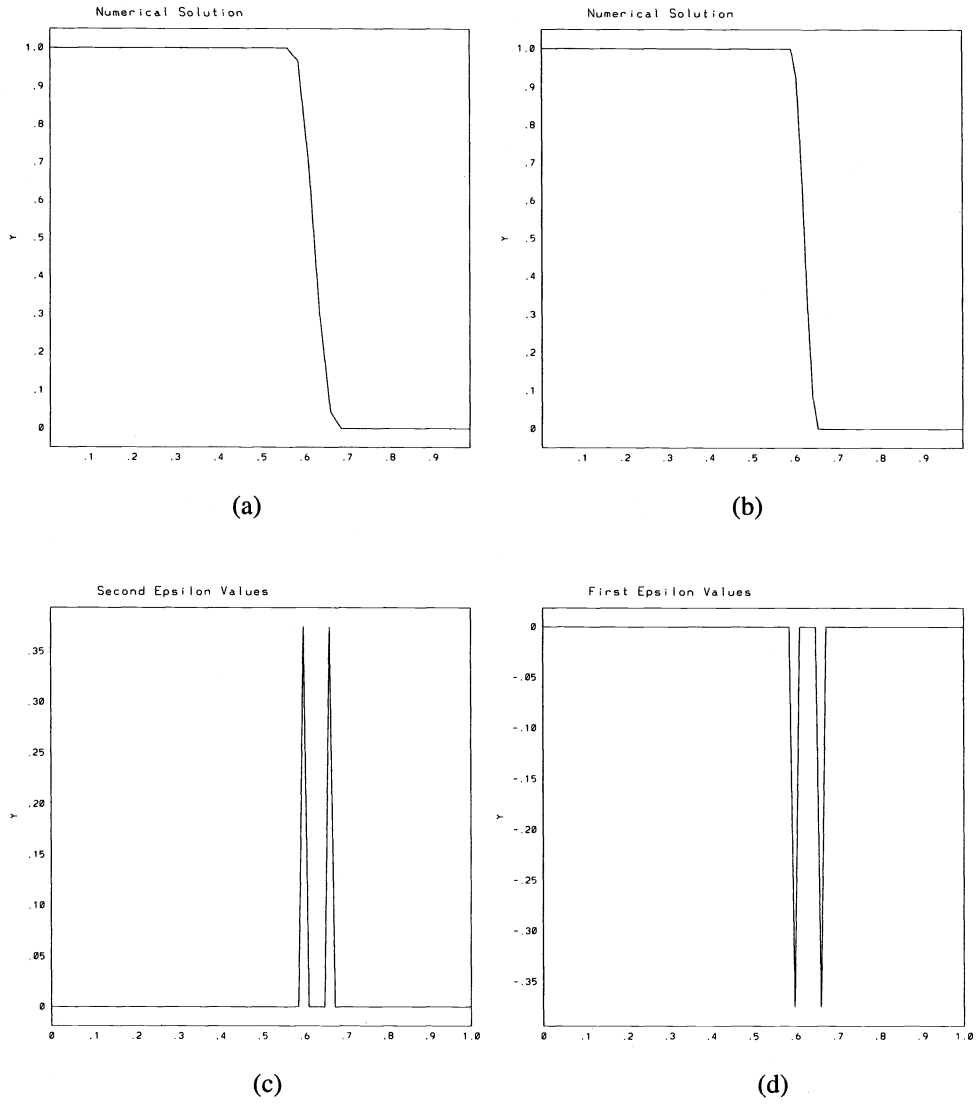


FIG. 5. *Transport of a shock in the case of linear advection using a symmetric third order method generated by a quadratic polynomial. The approximations using* 40 *nodes and* 80 *nodes ares shown in* (a) *and* (b), *respectively. The computed perturbations in* (b) *for the two polynomial evaluations are shown in* (c) *and* (d).

of the perturbation are localized to small bands. The localization of the optimization will be addressed in a paper on the AOM applied to nonlinear problems.

6. Summary. In this paper we have presented a framework for the construction of numerical schemes that provide constrained approximations of the solutions of scalar conservation laws. The construction includes the ability to specify higher order accuracy in regions where the solution is smooth. In regions where the solution has large gradients, the method perturbs

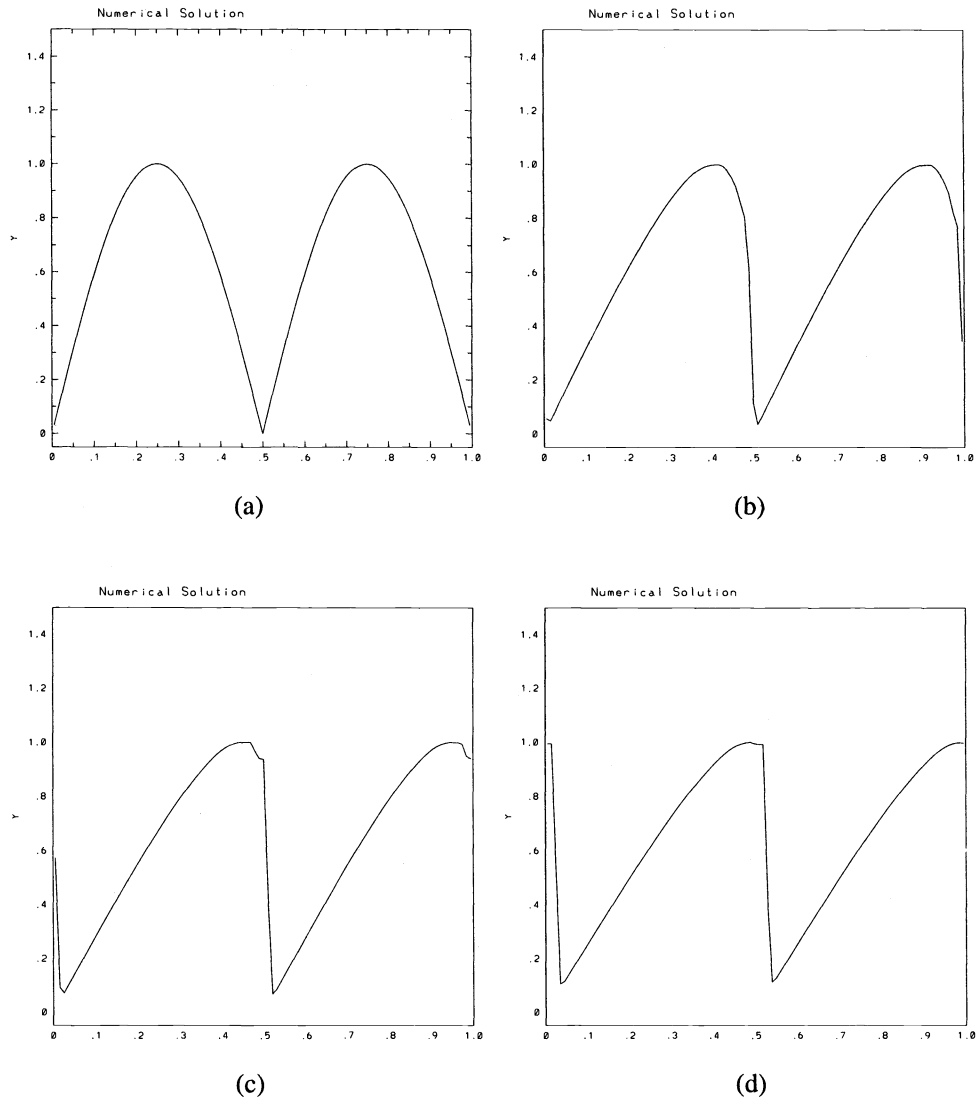


FIG. 6. *Transport of* $|\sin(2\pi x)|$ *in Burgers' equation using an* AOM *with a second order base scheme. The four figures include* (a) *the initial condition,* (b) *the approximation just before shock formation,* (c) *the approximation as the shock is forming, and* (d) *the approximation after the shock is fully formed and being transported.*

the higher order base scheme locally to meet imposed constraints. The AOM examples presented here use monotonicity preserving constraints that require that the approximation reduce to first order in regions where the solution is not smooth. The conditions also imply that the approximation is TVD. Thus the framework provides a method for the construction of higher order TVD schemes in a systematic way.

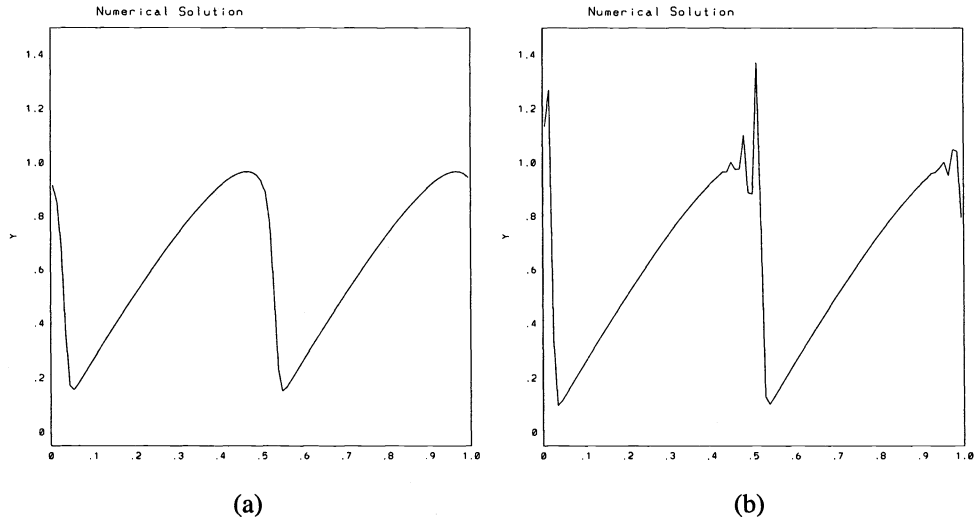


FIG. 7. *Transport of* $|\sin(2\pi x)|$ *in Burgers' equation using* (a) *first order upwinding and* (b) *unconstrained second order method at the same time level as in Fig.* 4(d).

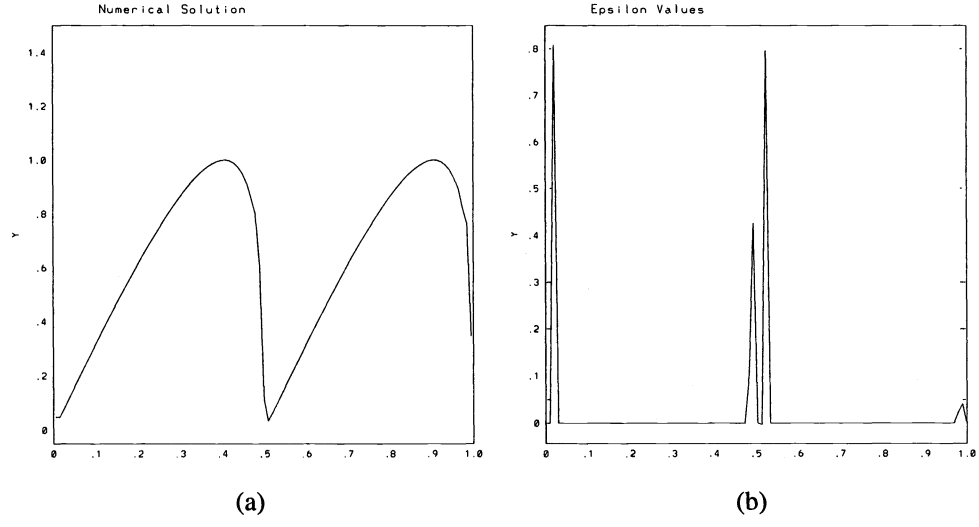


FIG. 8. *Transport of* $|\sin(2\pi x)|$ *in Burgers' equation using* AOM *at the same timestep as in Fig.* 4(b) *with perturbation values over the entire domain.*

REFERENCES

[1] J. BORIS AND D. BOOK, *Flux-corrected transport* I: *SHASTA, a fluid transport that works*, J. Comput. Phys., 11 (1973), pp. 38–69.

[2] C. CORAY AND J. KOEBBE, *Accuracy optimized methods for constrained numerical solutions of hyperbolic conservation laws*, J. Comput. Phys., 109, (1993), pp. 115–132.

[3] A. HARTEN, *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.

[4] A. ISERLES AND G. STRANG, *The optimal accuracy of difference schemes*, Trans. AMS, 277 (1983), pp. 779–803.

[5] J. KOEBBE, *Numerical Schemes for the Immiscible Displacement Equations Using a General Polynomial Framework for the Saturation Equation and Mixed Finite Element Methods for the Pressure*, Ph.D. thesis, University of Wyoming, Laramie, Aug. 1988.

[6] P. LAX AND B. WENDROFF, *Systems of conservation laws*, Comm. Appl. Math., 13 (1960), pp. 217–237.

[7] R. J. LEVEQUE, *Numerical Methods for Conservation Laws*, Birkhauser, Basel, 1990.

[8] S. OSHER, *Riemann solvers, the entropy condition, and difference approximations*, SIAM J. Numer. Anal., 21 (1984), pp. 217–235.

[9] S. OSHER AND S. CHAKRAVARTHY, *High resolution schemes and the entropy condition*, SIAM J. Numer. Anal., 21 (1984), pp. 955–984.

[10] P. SWEBY, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[11] B. VAN LEER, *Towards the ultimate conservative difference scheme*, II *monotonicity and conservation in a second order scheme*, J. Comput. Phys., 14 (1974), pp. 361–380.

[12] S. ZALESAK, *Fully multidimensional flux corrected transport algorithms for fluids*, J. Comput. Phys., 31 (1979), pp. 335–362.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 866–891, July 1994

006

DOMAIN DECOMPOSITION TO SOLVE TRANSITION LAYERS AND ASYMPTOTICS*

MARC GARBEY†

Abstract. The author considers the numerical computation of stiff nonlinear partial differential equations (PDEs) that can be studied by the methods of singular perturbation. Two domain decomposition methods that solve numerically the layers of the singular perturbation problem are presented. These numerical methods use at different stages the information given by the asymptotic analysis. The simplified model of reacting flow of Majda [*SIAM J. Sci. Statist. Comput.*, 7 (1986), pp. 1059–1080], a singular perturbation problem with a turning point [L. Abrahamsson and S. Osher, *SIAM J. Numer. Anal.*, 19 (1982), pp. 979–992], and a combustion problem [J. Pelaez, *SIAM J. Appl. Math.*, 47 (1987), pp. 781–799] that models a sequence of two chemical reactions will be considered as test problems.

Key words. domain decomposition, singular perturbation

AMS subject classifications. 65M55, 65M70, 65M35

1. Introduction.

Our aim concerns the numerical computation of stiff nonlinear PDEs that can be studied by the methods of singular perturbation analysis. We present some domain decomposition (DD) methods that solve numerically the layers of singular perturbation problems of the following type:

$$-\epsilon\frac{\partial}{\partial x}\left(P(U)\frac{\partial U}{\partial x}\right)+\frac{\partial U}{\partial t}+\frac{\partial}{\partial x}F(U)+D(U,v)=0. \tag{1}$$

We show with two different DD methods that the combination of asymptotic and numerical analysis provides improved accuracy and/or improved efficiency for such multiple scale problems.

We focus our study on singular perturbation problems with *strong* reaction term, i.e., $D(U, v) >> 1$ in some transition layers. It is well known that strong reaction terms give rise to numerical difficulties: for example, a splitting method with finite differences to solve a spiked strong detonation profile, with an error less than one percent in the maximum norm, can require 560 discretization points when ϵ is not even small (cf. [20, p. 1066]).

Many physical problems have multiple scales [15], [22]; a typical situation occurs when physics on the fastest scale induces narrow regions where the variation in the solution is large. Such regions are called *boundary layers* (BL) or *transition layers* (TL) depending on whether they are near a boundary or inside the interior of the domain. Examples of such situations are laminar flow of a slightly viscous fluid or combustion with high activation energy. Classical schemes applied to these types of situations generally fail to describe correctly the behavior inside the layers and sometimes the speed of propagation of the interfaces. Also, most of the previous problems can be seen as singular perturbation problems. It seems a natural idea to implement some of the results of the singular perturbation analysis in the numerical computation. Interest in this field of research has been increasing in the last few years, see [2], [10], [11], and their references.

In this paper, we present first an asymptotic-induced numerical method based on a hyperbolic scheme. The idea is that the matched asymptotic technique is typically a DD method; in such asymptotic analysis, we split the domain into subdomains where different processes occur; we look for the right scaling in each subdomain and derive the appropriate subproblems.

*Received by the editors February 12, 1992; accepted for publication (in revised form) May 4, 1993. This work was supported by Direction de la Recherche et des Etudes Techniques/SR/ 901968.

†University Claude Bernard Lyon1, LAN, Bat101, 69622 Villeurbanne cedex, France (garbey@lan1.univ-lyon1.fr).

We use the matching relations to connect the subproblems and so on. So the asymptotic-induced numerical method that we present is a numerical algorithm that is in some sense the image of the matched asymptotic analysis. Also, the matched asymptotic relations that are difficult to check are validated through the computation of the residual. This method has been applied to singular perturbation problems driven by conservation laws, for example the isentropic gas dynamic equation with a physical viscosity, in [8], [17]. In this paper, we extend the method to equations involving a source term $\epsilon^{-1} D(U)$ as in reacting flows.

We present second an adaptive DD method in the context of pseudospectral approximation with Chebyshev polynomials. The idea is to use as part of the criterion for the adaptivity what one knows from the analysis. Stiff fronts are a major difficulty in the use of spectral methods [3]. Very efficient adaptive methods have been developed in [6], [4], [25], [13] that solve stiff problems. This method has been extended in [5] to the case of multiple fronts via an adaptive domain decomposition. We show how one can enhance this adaptive method for singular perturbation problems: the position of the interfaces, the choice of the mapping, and the strategy of the DD can be effectively related to the asymptotic analysis.

The two DD methods presented here are complementary approaches. The first method starts from scratch and builds a new numerical algorithm based on the matched asymptotic method. The second method starts from an existing adaptive method and adds some improvement using asymptotic analysis. We will also see them as complementary methods because they are best on different types of singularities, i.e., formation or interaction of shocks as opposed to shocks waves.

We will consider mainly as a test problem the simplified model of reacting flow of Majda. It has been shown in [20] that this model contains some of the difficulties of the numerics for reacting flows. This model plays the role of the Burgers equation for the Navier–Stokes equation. In particular, in the computation of reacting shock waves, one can see in the simplified model that the viscosity balances the source terms in the layer and influences the speed of propagation. So one cannot expect to compute such phenomena with a classical hyperbolic scheme. In addition, as shown in [20], a splitting method applied to the operator required an order of magnitude with more discretization points needed than to compute, for example, a Buckley and Leverett equation in the one-dimensional case. So we demonstrate the efficiency of our DD methods on the difficult case of reacting shock layer. In the context of pseudospectral method, we will give some results on a singular perturbation problem with a turning point [1], [21], and on a combustion problem [24] that models a sequence of two chemical reactions.

2. Asymptotic-induced numerical methods based on an hyperbolic scheme. Let us first review briefly our asymptotic-induced numerical method in the case of a singular perturbation problem driven by a conservation law [16], i.e.,

$$\frac{\partial U}{\partial t} + \frac{\partial}{\partial x} F(U) = \epsilon \frac{\partial}{\partial x} \left(P(U) \frac{\partial U}{\partial x} \right).$$

The solution of the inviscid problem, $\frac{\partial U}{\partial t} + \frac{\partial}{\partial x} F(U) = 0$, can exhibit some singularities such as shocks, weak discontinuities that propagate along the characteristics, and interaction of singularities. To each singularity corresponds a thin layer, where the viscous perturbation cannot be neglected. One can identify using a matched asymptotic technique the order of magnitude of the residual and the scaling of the layer for each type of singularities in the scalar case cf. Table 1.

In particular, this result holds for a shock layer in the case of a system of conservation laws. When we use a hyperbolic scheme as, for example, a Godunov scheme to compute a conservation law with or without viscosity, we can use this information to identify the zone of a shock, based on the computation of the residual. This may require the use of two grids or

TABLE 1
Asymptotic order of residual.

Type of zone	Order of residual	Local coordinates	
	$u_t + f(u)_x$	ξ	τ
Regular zone	$O(\epsilon)$	x	t
Shock layer	$O(\epsilon^{-1})$	$(x - S(t))/\epsilon$	t
Weak singularity	$O(\epsilon^{1/2})$	$(x - S(t))/\epsilon^{1/2}$	t
Shock interaction with other singularities	$O(\epsilon^{-1})$	$(x - S_0 - S_1 t)/\epsilon$	$(t - t_o)/\epsilon$
Discontinuity with f locally linear	$O(1)$	$(x - S_0 - S_1 t)/\epsilon^{1/2}$	$(t - t_o)$
Formation of shock	$O(\epsilon^{-1/4})$	$(x - S_0 - S_1 t)/\epsilon^{3/4}$	$(t - t_o)/\epsilon^{3/4}$

more; however, we can identify three categories of points depending on whether it is a regular zone, a shock, or something else. Layers that correspond to weak singularities can be solved using a regular correction technique as in [17] (with eventually some adaptivity). Interaction of singularities needs a stretching in time and space in the subdomain according to Table 1. We will emphasize here the treatment of the *shock layer*. We first review briefly the analysis of a shock layer based on the asymptotic technique of matching [14] for a singular perturbation problem driven by a system of conservation laws.

2.1. Shock layer analysis and system of conservation laws. Let us consider the Cauchy problem:

$$\text{(2)} \qquad \begin{cases} \frac{\partial U}{\partial t} + \frac{\partial}{\partial x} F(U) = \epsilon \frac{\partial}{\partial x} \left(P(U) \frac{\partial U}{\partial x} \right) & \text{for } (x,t) \in \Omega, \\ \\ U(x,0) = V(x) & \text{for } x \in \mathbb{R}. \end{cases}$$

Here the solution $U : \Omega \to \mathbb{R}^n$ is a vector-valued function, the domain is $\Omega = \mathbb{R} \times]0, T[$, and $\epsilon << 1$ is a small parameter.

We assume that V is piecewise smooth. We also assume that F and P are smooth functions of U. We suppose that P is a *suitable viscosity matrix* [12] for the shocks of the following associated inviscid problem:

$$\text{(3)} \qquad \begin{cases} \frac{\partial U}{\partial t} + \frac{\partial}{\partial x} F(U) = 0 & \text{for } (x,t) \in \Omega, \\ \\ U(x,0) = V(x) & \text{for } x \in \mathbb{R}. \end{cases}$$

Namely, a shock wave solution to (3) can be obtained as a limit of progressive wave solutions of (2). Problem (2) is a parabolic-hyperbolic singular perturbation problem driven by (3).

One obtains easily the regular expansion

$$\text{(4)} \qquad U_{\text{as}}^{\text{outer}} = U^0 + \epsilon U^1 + \epsilon^2 U^2 + \cdots$$

that is a priori valid outside the neighborhood of the singularities of the solution to (3). We substitute $U_{\text{as}}^{\text{outer}}$ in the differential equation of (2) and use identification in ϵ to find that U^0

must be a solution of (3). We also find that U^1 must be a solution of the following linear hyperbolic problem:

$$\text{(5)} \qquad \begin{cases} \frac{\partial U^1}{\partial t} + \frac{\partial}{\partial x}\left(DF(U^0)U^1\right) = \frac{\partial}{\partial x}\left(P(U^0)\frac{\partial U^0}{\partial x}\right) & \text{for } (x,t) \in \Omega, \\ \\ U^1(x,0) = 0 \quad \text{for } x \in \mathbb{R}, \end{cases}$$

where DF denotes the Jacobian matrix of F. The inviscid problem (3) has many weak solutions; we shall uniquely define U^0 as the analysis progresses.

We assume that solutions U^0 of (3) are smooth except on piecewise regular curves $S_k(t)$. It is assumed for $t \in [t_0, t_1]$ that the S_k are isolated from each other. Without loss of generality, we may drop the subscript k. S is assumed to be smooth for $t \in [t_0, t_1]$. In particular, we assume no focusing of characteristics. To make this more precise, we suppose that there is an interval of time $[t_0, t_1]$ such that for each $t \in [t_0, t_1]$ and for each S the following limits exist:

H0. $\boxed{U_l^0 = \lim_{x \to S(t)^-} U^0(x,t), \quad U_r^0 = \lim_{x \to S(t)^+} U^0(x,t).}$

Let us define the change of variable: $\tilde{x} = x - S'(t)\,t$ and $\tau = t$. We denote $\tilde{U}(\tilde{x}, \tau) = U(x,t)$. We further assume that:

H1. $\boxed{U_{\tau,l}^0 = \lim_{\tilde{x} \to 0^-} \frac{\partial U^0}{\partial \tau}, \quad U_{\tau,r}^0 = \lim_{\tilde{x} \to 0^+} \frac{\partial U^0}{\partial \tau},}$

H2. $\boxed{U_l^{j,i} = \lim_{\tilde{x} \to 0^-} \frac{\partial^i U^j}{\partial \tilde{x}^i}, \quad U_r^{j,i} = \lim_{\tilde{x} \to 0^+} \frac{\partial^i U^j}{\partial \tilde{x}^i}, \text{ for } (i,j) = (1,0),\ (0,1),}$

H3. $\boxed{U_{\tau,l}^0 = O(1), \quad U_{\tau,r}^0 = O(1).}$

The shock layer profile will not have rapid temporal variation, so it is appropriate to scale and translate only the spatial variable (and not the temporal variable). Such a transformation is defined by

$$\text{(6)} \qquad \xi = \frac{x - S(t)}{\epsilon} \quad \text{and} \quad \tau = t,$$

where we denote $\hat{U}(\xi, \tau) = U(x,t)$. Under this transformation the differential equation of problem (2) becomes

$$\text{(7)} \qquad \frac{\partial \hat{U}}{\partial \tau} + \epsilon^{-1}\frac{\partial}{\partial \xi}\left(F(\hat{U}) - S(\tau)\hat{U}\right) = \epsilon^{-1}\frac{\partial}{\partial \xi}\left(P(\hat{U})\frac{\partial \hat{U}}{\partial \xi}\right).$$

This suggests an expansion in the TL of the form

$$\text{(8)} \qquad \hat{U}_{\text{as}}^{\text{inner}} = \hat{U}^0 + \epsilon\hat{U}^1 + \cdots,$$

where $\hat{U}^0$ and $\hat{U}^1$ are functions of ξ and τ. Using this expansion in (7) and imposing the matching relations [14], [16] with the outer expansion $U_{\text{as}}^{\text{outer}}$, we derive ordinary differential equation (ODE) problems for $\hat{U}^0$ and $\hat{U}^1$. The equation for the first term is

$$\text{(9)} \qquad \begin{cases} -\frac{\partial}{\partial \xi}\left(P(\hat{U}^0)\frac{\partial \hat{U}^0}{\partial \xi}\right) + \frac{\partial}{\partial \xi}\left(F(\hat{U}^0) - S'(t)\hat{U}^0\right) = 0, \\ \\ \hat{U}^0 \to U_l^0 \quad \text{as } \xi \to -\infty, \\ \hat{U}^0 \to U_r^0 \quad \text{as } \xi \to +\infty, \end{cases}$$

and for the second term we have

$$
(10)\qquad \begin{cases} -\frac{\partial^2}{\partial \xi^2}\left(P(\hat{U}^0)\cdot \hat{U}^1\right) + \frac{\partial}{\partial \xi}\left(DF(\hat{U}^0)\cdot \hat{U}^1 - S'(t)\hat{U}^1\right) = -\frac{\partial \hat{U}^0}{\partial \tau}, \\ \|\hat{U}^1 - (U^1_{0,l}\xi + U^0_{1,l})\|_o \to 0 \quad \text{as } \xi \to -\infty, \\ \|\hat{U}^1 - (U^1_{0,r}\xi + U^0_{1,r})\|_o \to 0 \quad \text{as } \xi \to +\infty, \end{cases}
$$

where $\|U\|_o = \max_{i=1,\dots,n}(|u_i|)$ and $U = (u_i)_{i=1,\dots,n}$. The temporal variable τ can be considered as a parameter in the transition layer because the above problems require only the solution of ODEs. Also, the problems for $\hat{U}^1$ is linear.

Next we discuss the existence of $\hat{U}_0$ and the uniqueness of U_0 as well as the uniqueness of the curve S. Using a matching relation on the first spatial derivates on the terms in the expansion $U^{\text{inner}}_{\text{as}}$, one looks for $\hat{U}^0$ such that

$$
\frac{\partial \hat{U}^0}{\partial \xi} \to 0 \quad \text{as } \xi \to \pm\infty.
$$

We integrate (9) from $-\infty$ to ξ to obtain:

$$
(11)\qquad P(\hat{U}^0)\frac{\partial \hat{U}^0}{\partial \xi} = H(\hat{U}^0) - H(U^0_l),
$$

where $H(U) = F(U) - S'(t)U$. Thus, the existence of a solution to (9) implies the Rankine–Hugoniot (RH) condition

$$
H(U^0_l) = H(U^0_r).
$$

This means that U^0_l and U^0_r are critical points of the dynamical system (11).

In addition to the RH condition, we must be able to construct the layer; thus, we assume that:

H4. There exists a unique trajectory for the dynamical system (11) from U^0_l to U^0_r.

For scalar conservation laws, H4 is equivalent to the classical geometric entropy condition (GEC); however, this equivalence is not true in general. The existence of a viscous profile (i.e., H4) is not of the same nature as the GEC. However, we restrict our problems to cases where the RH condition and H4 are enough to uniquely define U^0 and S (and, consequently, $U^{\text{outer}}_{\text{as}}$).

Now we have determined $\hat{U}^0$ up to a translation in the spatial variable ξ.

A solvability condition for the problem that $\hat{U}^1$ satisfies and a construction of $\hat{U}^1$ up to the kernel function $\frac{\partial}{\partial \xi}\hat{U}^0$ may be derived. We leave the details of the computation to the interested reader.

We can prove that $\hat{U}^1$ is a smooth function if and only if $\hat{U}^0$ satisfies the relation

$$
\frac{\partial}{\partial t}\left\{\int_{-\infty}^{0}\left(\hat{U}^0(\xi,t) - U^0_l(t)\right)d\xi + \int_0^{\infty}\left(\hat{U}^0(\xi,t) - U^0_r(t)\right)d\xi\right\}
$$

$$
= \left[\!\left[S'(t)U^1 - DF(U^0)\cdot U^1 + P(U^0)\frac{\partial U^0}{\partial x}\right]\!\right],
$$

where $[\![\cdot]\!]$ denotes the jump across the shock. This is simply an area relation that determines the shift in ξ for $\hat{U^0}$. Let us assume for simplicity that U and its space derivatives vanish at infinity. One can show that this relation is a consequence of the conservation relation

$$\int_{-\infty}^{\infty} \frac{\partial U}{\partial t}\, dx = 0,$$

satisfied by solutions to (2) and (3). This relation is also satisfied by our uniform approximation of the solution to (2), where

$$U_{\text{as}} = (1 - K(\xi_\nu))U_{\text{as}}^{\text{outer}} + K(\xi_\nu)(\hat{U^0} + \epsilon\hat{U^1}) + O(\epsilon^2).$$

Here $\xi_\nu = (x - S(t))/\epsilon^\nu$ is the intermediate variable and K is a smooth cutoff function such that:

$$K(y) = \begin{cases} 0 & \text{if } |y| > 2, \\ \\ 1 & \text{if } |y| < 1. \end{cases}$$

More precisely, we have

$$\int_{-\infty}^{\infty} \frac{\partial U_{\text{as}}}{\partial t}\, d\xi = O(\epsilon). \tag{12}$$

2.2. Numerical treatment of a shock layer. We first review briefly the ideas of an asymptotic-induced numerical treatment of a shock layer. We refer to [8] and [17] for more details about the method and some numerical experiments. We know that in the neighborhood of strong singularities, the residual $\frac{\partial}{\partial t}U^\epsilon + \frac{\partial}{\partial x}F(U^\epsilon)$ and the viscosity $\epsilon\frac{\partial}{\partial x}(P(U^\epsilon)\frac{\partial}{\partial x}U^\epsilon)$ are of order ϵ^{-1}. Thus such zone of singularity is easily detected based on the numerical approximation of the viscous problem (2) or the inviscid problem (3). For example, we can use a Godunov scheme to compute (3) and enhance the accuracy on the localization of the singularities using two or more levels of grids. Let $\Omega_0(t)$ be a zone of singularity for a given time t; let U^h be the numerical approximation of (2) outside Ω_0. For a given curve of discontinuity $S(t)$ inside $\Omega_0(t)$ one can construct W_S^h that extrapolates U^h inside $\Omega_0(t)$ on each side of $S(t)$. Let W_{S-}^h, W_{S+}^h be the left and right values of W_S^h on $S(t)$; the conservation of mass, i.e.:

$$\int_{\Omega_0(t)} U^h - W^h = O \tag{13}$$

gives us an approximation of $S(t)$. Then we can check numerically the assumptions H_0 to H_4 and conclude if it is a shock layer or not.

Then we implement numerically the asymptotic analysis of the shock layer: that is, a DD method; we solve the layer by computing the solution of the ODE (11) that matches with the interface condition on $\partial\Omega_0$. The correction for the slope on $\partial\Omega_0$ is satisfied by the next order term in the layer. Let us mention that we can also use this DD method to minimize the viscosity of the hyperbolic scheme in the layer: we take W^h as the solution of the inviscid problem in the layer except on a grid point that neighbors $S(t)$ in order to keep the scheme conservative. We refer to [8] and [17] for more details.

2.3. Asymptotic-induced numerics for reacting shock layer. Now we are going to describe more precisely the method extended to the simplified model of reacting flow of Majda, that is:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}[F(u) - q_o Z] = \epsilon \frac{\partial^2 u}{\partial x^2}, \tag{14}$$

$$Z_x = \varepsilon^{-1}\phi(u)Z, \tag{15}$$

where $\phi(u) = 1$ if $u > 0$ and 0 elsewhere. We refer to [20] for the derivation of this model and its precise statement.

This model can be compared to the following class of diffusion-convection-reaction problem:

$$-\varepsilon \frac{\partial}{\partial x}\left(P(u)\frac{\partial u}{\partial x}\right) + \frac{\partial u}{\partial t} + \frac{\partial}{\partial x}F(u) + D(u,\varepsilon) = 0, \tag{16}$$

when $D(u, \varepsilon)$ is of order 1 except possibly in some transition layer; if $D(u, \varepsilon)$ is uniformly bounded, $D(u, \varepsilon)$ occurs as a regular perturbation in a shock layer, i.e., $D(\hat{u}_0, \varepsilon)$ appears only in the ODE problem satisfied by $\hat{u}_1$. Therefore, the previous asymptotic analysis of a shock layer extended to (16) has only some minor modification.

We will emphasize the case when $D(u, \varepsilon)$ balances the viscosity in a transition layer, that is $D(u, \varepsilon) \sim \varepsilon^{-1} D_1(\hat{u})$ with $D_1(\hat{u}) = O_s(1)$. We will refer in this case to a *reacting shock layer.*

Now, the first order term in this layer satisfies the ODE problem:

$$\begin{cases} -\frac{\partial}{\partial \xi}\left(P(\hat{U}^0)\frac{\partial \hat{U}^0}{\partial \xi}\right) + \frac{\partial}{\partial \xi}\left(F(\hat{U}^0) - S'(t)\hat{U}^0\right) + D_1(\hat{U}_0) = 0, \\ \\ \hat{U}^0 \to U_l^0 \quad \text{as } \xi \to -\infty, \\ \hat{U}^0 \to U_r^0 \quad \text{as } \xi \to +\infty, \end{cases} \tag{17}$$

Thus, the RH relation becomes the following jump condition:

$$\left[\left[F(\hat{U}_0) - S'(t)\hat{U}_0\right]\right] + \int_{-J\infty}^{+\infty} D_1(\hat{U})d\xi = 0.$$

In particular, a hyperbolic scheme applied to (16) will not, in general, give the right speed of propagation for a reacting shock layer, since the numerical viscosity of the scheme will interact with the source term.

The asymptotic analysis of a reacting shock layer for the Majda problem shows that the first order term in the layer satisfies:

$$\begin{cases} \hat{U}_{0,\xi} = H(\hat{U}_0, \hat{Z}_0) - H(U_{l/r}, Z_{l/r}), \\ \hat{Z}_{0,\xi} = \phi(\hat{U}_0)\hat{Z}_0, \\ \\ \hat{U}_0 \to U_{l/r} \text{ for } \xi \to \mp\infty, \\ \phi(U_l) = 1; \phi(U_r) = 0, \\ \hat{Z}_0 \to 0 \text{ for } \xi \to -\infty, J, \\ \hat{Z}_0 \to 1 \text{ for } \xi \to +\infty, \end{cases} \tag{18}$$

where $H(u, z) = \frac{u^2}{2} - S'(t)u - q_0 Z$.

To ensure the ability to construct the layer, more precisely the existence of a solution of (16), we suppose the jump condition:

$$\frac{1}{2}U_l^2 - S'(t)U_l = \frac{1}{2}U_r^2 - S'(t)U_r - q_0$$

and assume that there exists a trajectory of (18) from $(U_l, 0)$ to $(U_r, 1)$. An extensive study of (18) is given in [20].

Also, we obtain the shift on $\hat{U}_0$ using a conservation relation analogous to (12), i.e.:

$$\frac{\partial}{\partial t}\int_{\mathbb{R}} u_{\text{as}} = q_0 - [\![F(U)]\!]_{U_l}^{U_r} + 0(\varepsilon). \tag{19}$$

Now, we present the numerical method based on these results. We start from the following elementary finite difference scheme:

$$\begin{cases} \frac{U_i^{n+1}-U_i^n}{\Delta t} = \varepsilon \frac{U_{i-1}^n - 2U_i^n + U_{i+1}^n}{\Delta x^2} - \frac{F(U_{i+1}^n) - F(U_{i-1}^n)}{2\Delta x} + q_0 \frac{Z_{i+1}^n - Z_{i-1}^n}{2\Delta x}, \\ \frac{Z_i^{n+1}-Z_{i+1}^{n+1}}{\Delta x} = K\,\phi(U_{i+1}^{n+1})\, Z_{i+1}^{n+1}. \end{cases}$$

This explicit scheme must be improved by using a Riemann solver to compute the flux. However, for our purpose, we do not need a more sophisticated scheme, as long as the scheme is conservative. We are interested in the reacting shock layer: so, our numerical test identifies the zone of strong singularities where *both* the viscosity and the reaction terms are of order ε^{-1}.

As in [8] we obtain U_l, U_r, and some approximation of the position of the interface. Let us notice that in our numerical experiment we solve a Riemann problem and U_l, U_r are independent of the time. Therefore, $\hat{U}_0$ is the only nonzero term of the reacting shock layer.

A solution (U_0, Z_0) of (16) is then computed with any ODE scheme. Then, one substitutes to (U_i^n, Z_i^n) in the reacting shock layer, the only travelling wave $(\hat{U}_0, \hat{Z}_0)$ that satisfies the conservation relation (19).

We have experimented first with this DD method in the spiked strong detonation case with $U_l = 1.0$, with $U_r = -1.5$, and $q_0 = 2.375$. We use 40 points on the regular grid to solve the problem with $\varepsilon = 0.05$ (see Fig. 1). Numerically, the thickness of the layer is about 12ε. One can see the ε jump at the interface between the inner domain and the outer domains. We obtain some analogous results with 400 points on a regular grid and $\varepsilon = 0.005$ (see Fig. 2). Because U_l and U_r are so easy to obtain numerically, there is no significant error in the speed of propagation of the layer. In the future, we will solve the more difficult case where singularities interact.

The problem of the initial formation of this reacting shock layer will be solved with the DD method presented in the next section.

We can also apply the minimum viscosity method of [8] to this problem in a straightforward way. We keep only one artificial point in the layer to let the scheme be conservative. Figure 3 shows the result with 40. We have done this computation with various numbers of discretization points: in any case, the composite scheme tracks the interface with an error on the location of the interface that is less than the meshsize.

We have solved a weak detonation case with $U_l = 1.0$, $U_r = -0.4$, $q_0 = 0.568$. This case requires a high order ODE scheme to compute the layer. However, the minimum viscosity method works as well as for the case of a strong detonation.

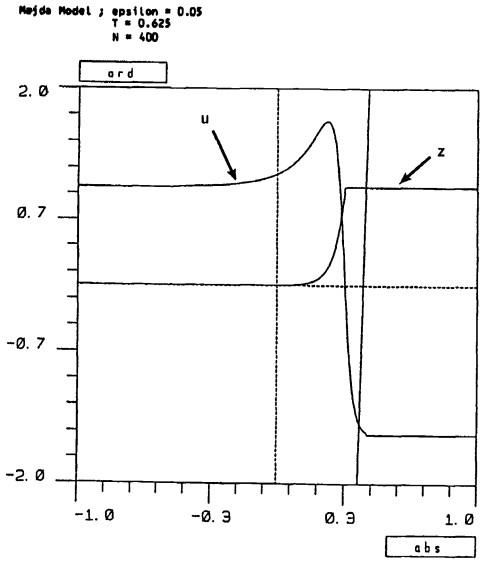


FIG. 1. *Majda problem with* $\varepsilon = 0.05$.

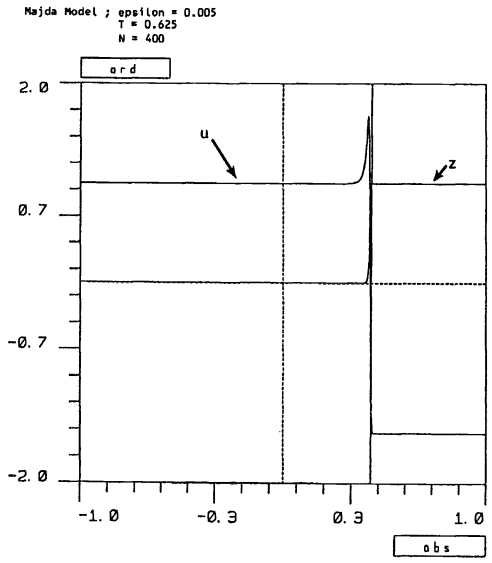


FIG. 2. *Majda problem with* $\varepsilon = 0.005$.

3. Adaptive domain decomposition with pseudospectral. When solving a regular problem with DD and a fixed total number of collocation points, the accuracy decreases as the number of domains increases. This is characteristic of pseudospectral accuracy. However, for stiff problems, numerical experiments demonstrate that one can improve drastically the numerical accuracy of the pseudospectral approximation by adapting the DD. This adaptivity is based on the a priori estimates of [6]. However, for PDEs with moving layers, the cost of adaptivity obviously increases as a function of the speed of propagation of the layers and more generally the complexity of its dynamics. This cost can be reduced by using the information

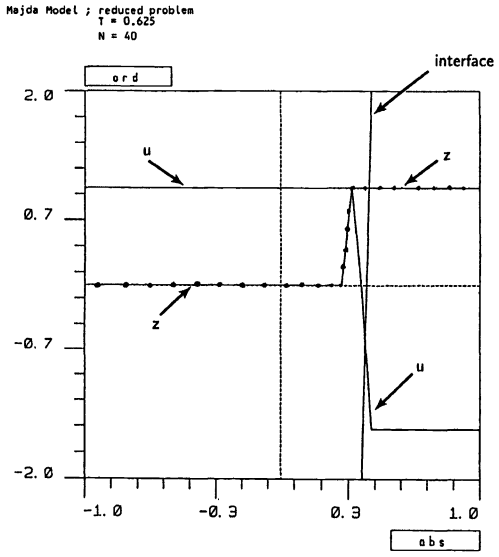


FIG. 3. *Majda problem with the minimum viscosity method.*

given by the asymptotic analysis to localize the layers and obtain the magnitude of the mapping parameters for each type of layer.

We show how one can improve the efficiency of the adaptive DD method for singular perturbation problems.

Let us consider the typical singular perturbation problem studied in [1] (see also [19] and [21]). We have an example of a TL in the neighborhood of $x_o \approx 0.1$ and a BL at the right end of the interval. These layers are the major difficulties in the application of the Chebyshev pseudospectral method (see Fig. 7). In the singular perturbation theory [14], one introduces subdomains and stretching variables to solve these layers and obtain a uniform approximation of the solution. We will use some analogous tools in the numerical method that follows.

3.1. Mappings and subdomains. Let us introduce two one-parameter families of mappings [4]:

$$\begin{array}{ccc} [-1,1] & \longrightarrow & [-1,1], \\ s & & y = f_i(s,\alpha), \end{array} \tag{20}$$

with: $f_1(s,\alpha) = \pm\left[\frac{4}{\pi}\tan^{-1}(\alpha \quad \tan(\frac{\pi}{4}(\pm s - 1))) + 1\right]$ and $f_2(s,\alpha) = \alpha \ \tan(s \ \tan^{-1}(\alpha^{-1}))$. The small free parameter, α, describes how one concentrates the collation points in the physical space.

We will call f_1 a mapping of BL type and f_2 a mapping of TL type. We use in f_1 a plus sign (respectively, a minus sign) for a BL on the right (respectively, on the left) of the interval. A number of other mappings are possible of course; we will restrict ourselves to the previous mappings.

In singular perturbation, one uses stretching variables of the form $\xi = (x - x_k)/\epsilon$, where x_k is the location of the layer and ϵ is a measure of the stretching. It is easy to see that the parameter α in the nonlinear mappings f_1 and f_2 plays an analogous role as ϵ in the numerical method. One needs also to focus the stretching on a subdomain $[x_k - L_k, x_k + L_k]$ of Ω. So we introduce a second one-parameter family of affine mappings; to solve a TL, we use:

$$\begin{array}{ccc} [-1,1] & \longrightarrow & [x_k - L_k, x_k + L_k], \\ y & & x = g_k(y, L_k), \end{array} \tag{21}$$

to solve a BL on the right, for example, we use:

$$\begin{array}{ccc} [-1,1] & \longrightarrow & [x_k - L_k, x_k], \\ y & & x = g_k(y, L_k). \end{array} \tag{22}$$

A difficult problem is to compute the free parameter, α, which is characteristic of the stretching and to localize the layer, i.e., to compute x_k. We have two tools at our disposal. The first tool is the asymptotic analysis of the PDE involving a critical parameter. This tool is strongly related to the PDE, so we will emphasize this aspect in the last section where we study specific examples. The second tool is approximation theory and requires a purely automatic treatment cf. [4]–[6]; we refer to these papers in the following for the exact definition of the *a priori estimates*. We will extend these ideas in the context of an arbitrary number of subdomains to solve singular perturbation problems.

3.2. How to adapt? Let us present the pseudospectral technique from the simplest case to the general situation: let us first consider the *single domain* case with no adaptivity. Let u be any smooth real function on Ω, a compact subset of $\mathbb{R}$, and h an affine function from $[-1, 1]$ into Ω. In the Chebyshev pseudospectral method, $u(x)$ is approximated as a finite sum of Chebyshev polynomials: $P_N u = \sum_{j=0}^{N} a_j T_j(h^{-1}(x))$, where $T_j(y) = \cos(j \cos^{-1}(y))$ and the coefficients a_j are obtained from collocating the solution at the points $(x_j)_{j=0,\ldots,N}$ such that $\{h^{-1}(x_j) = \cos(\frac{\pi j}{N});\ j = 0, \ldots, N\}$.

Let e_N be the error $e_N = u - P_N u$; let us denote $|\cdot|$ the maximum norm. Let $\tilde{u}(h^{-1}(x))$ be $u(x)$. One has the a priori estimate: $|e_N| \leq C^t \frac{\|\tilde{u}\|}{N^p}$, where p is an integer and C^t is a real number that depends specifically on the choice of the Sobolev norm $\| \ \|$.

Let us suppose now that u exhibits one BL or TL in Ω. Now we consider the *single domain* case with *adaptivity*. We choose the appropriate BL mapping f_1 or TL mapping f_2 to solve the layer. We obtain a one parameter approximation of u: $P_{N,\alpha} u = \sum_{j=0}^{N} a_j T_j(f_i^{-1}(h^{-1}(x), \alpha))$.

Let $\tilde{u}(f_i^{-1}(h^{-1}(x), \alpha))$ be $u(x)$; we have the a priori estimate: $| e_{N,\alpha} | \leq C^t \frac{\|\tilde{u}\|_\alpha}{N^p}$, where C^t is a constant *independent* of α.

Now the norm $\| \ \|_\alpha$ is depending on α. One observes that as in singular perturbation analysis, sharp estimates require norms parametrized by the singular parameter. The optimum choice of α can be obtained by solving the minimization problem:

$$\min_{\alpha \in [0,1]} (\| \tilde{u} \|_\alpha).$$

Practically, we will look rather in the numerical computation for the minimum of $\| P_{N,\alpha} \tilde{u} \|_\alpha$. Therefore, N must be large enough (according to the stiffness of u) such that the discretization does not miss the layer.

Now, let us consider a DD with M subdomains $[Q_0, Q_1]$, $[Q_1, Q_2], \ldots,$ $[Q_{M-1}, Q_M]$, and adaptivity within each subdomain; the unknown function u on the interval Ω is approximated by a piecewise polynomial $P_{N,I}$ of $C^1(\Omega)$, that is C^∞ inside each subdomain, depending on a set of parameters $I = \{\alpha_k, L_k\}_{k=1,\ldots,nd}$.

Let us denote by

$$\tilde{u}_k(f_i^{-1} o\, h_k^{-1}(x)) \equiv u(x),$$

the unknown function in the Chebyshev space restricted to the subdomain Ω_k.

One extends easily the previous a priori estimate to the multiple domain case with the norm:

$$\| (\tilde{u}_1, \ldots, \tilde{u}_{nd}) \|_I = \max_{k=1,\ldots,nd} \| \tilde{u}_k \|_{\Omega_k,\alpha_k};$$

that is,

$$|u - p_{N,I}| \leq C^t \frac{\| (\tilde{u}_1, \ldots, \tilde{u}_{nd}) \|_I}{N^p},$$

where p and C^t depend only on the choice of the Sobolev norm in each subdomain. We will have to compute nd free parameters α_i and $nd - 1$ interfaces Q_k. Thus we need to adopt the choice of a strategy to minimize $\| (\tilde{u}_1, \ldots, \tilde{u}_{nd} \|_I$ in case of multiple layers: in other words, we have to fix some constraints on the set of parameters I in order to keep affordable the problem of minimization. This strategy will be studied in §4.

3.3. Multiple domains technique and parallel computing. Let us briefly recall the multiple domain technique introduced in [18] and described here for an arbitrary number of domains. We consider the example of a nonlinear PDE:

$$\frac{\partial \Theta}{\partial t} = \frac{\partial^2 \Theta}{\partial x^2} + F(\Theta); \qquad (x, t) \in \Omega \times [0, T], \tag{23}$$

with some boundary and initial conditions. For simplicity, we restrict ourselves to use an implicit predictor corrector scheme in the following.

Moreover, we are using both vectorization and parallelization facilities to run our codes. Since most of our applications required few collocations points per subdomain, less than 40 usually, it is more convenient and efficient in our applications to use a matrix-multiply approach in the implementation of the pseudospectral method. Let us first consider the single domain case; the algorithm can be written for the predictor step:

$$\frac{\Theta^{n+1/2} - \Theta^n}{0.5\Delta t} = D^2\Theta^{n+1/2} + F^n, \tag{24}$$

and for the corrector step:

$$\frac{\Theta^{n+1} - \Theta^n}{\Delta t} = \frac{1}{2}(D^2\Theta^{n+1} + D^2\Theta^n) + F^{n+1/2}, \tag{25}$$

where D is the operator of differentiation. Each step of the time differencing scheme is equivalent to the solution of a linear system for the predictor step:

$$\tilde{D}\Theta^{n+1/2} = G_p(\Theta^n, \Delta t), \tag{26}$$

and for the corrector step:

$$\tilde{D}\Theta^{n+1} = G_c(\Theta^n, \Theta^{n+1/2}, \Delta t), \tag{27}$$

with $\tilde{D} = I - 0.5\ \Delta t\ D^2$.

Now, we look for the general construction of $\tilde{D}$ in the case of a DD computation of (23) when we impose $\Theta(., t) \in C^1(\Omega)$. We suppose Ω split into nd subdomains (without overlap). We assume, for simplicity, that we are using nd processors or clusters of processors of a parallel computer. Therefore, because of the load balancing of these M units, we assume the same number of collocation points N inside each subdomains. However, depending

on the parallel computer, some other choices might be more appropriate. We denote by $X_{i,k}$, $i = 1, \ldots, N$, $k = 1, \ldots, M$, the discretization points in physical space.

Let $\Theta^n_{i,k}$ be $\Theta(X_{i,k}, t_n)$, and let D_k be the differentiation operator for each subdomain Ω_k, $k = 1, \ldots, M$. D_k is parametrized by α_k and L_k. We impose the continuity conditions

$$\begin{aligned} \Theta^n_{N,k} &= \Theta^n_{0,k+1}, \\ D_k\Theta^n_{N,k} &= D_{k+1}\Theta^n_{0,k+1} \end{aligned} \tag{28}$$

for $k = 1, \ldots, M-1$.

$\tilde{D}$ has the following characteristic structure:

$$\tilde{D} = \begin{pmatrix} A_1 & \hat{\beta}(1) & & & & & & & & \\ t\tilde{\alpha}(1) & \gamma(1) & t\tilde{\beta}(1) & \delta(1) & & & & & & \\ & \hat{\alpha}(2) & A_2 & \hat{\beta}(2) & & & & & & \\ & & & & \cdots & & & & & \\ & & & & & \hat{\alpha}(i) & A_i & \hat{\beta}(i) & & \\ & & & & & \eta(i) & t\tilde{\alpha}(i) & \gamma(i) & t\tilde{\beta}(i) & \delta(i) \\ & & & & & & & \hat{\alpha}(i+1) & A_{i+1} & \hat{\beta}(i+1) \\ & & & & & & & & \cdots & \cdots \\ & & & & & & & & \hat{\alpha}(M) & A_M \end{pmatrix},$$

where

- $\gamma(i), \delta(i), \eta(i) \in \mathbb{R}$;
- A_i is an $(N-2) \times (N-2)$ matrix except for $i \neq 1$ and $i \neq nd$; A_1 and A_M are $(N-1) \times (N-1)$ matrices; and
- $\hat{\alpha}(i), \hat{\beta}(i), \tilde{\alpha}(i), \tilde{\beta}(i)$ are some vectors.

The rows that do not contain A_i correspond to the continuity of the first derivative at the interface given by (28).

The time marching scheme requires the solution of the linear system:

$$\begin{aligned} &\tilde{D}\,{}^t[\Theta^{(1)}(t_n), W^n_1, \Theta^{(2)}(t_n), W^n_2, \ldots, W^{n,(*)}_i, \ldots, \Theta^{(M-1)}(t_n), W^n_{M-1}, \Theta^{(M)}(t_n)] \\ &\quad = {}^t[F^{(1)}(t_n), 0, F^{(2)}(t_n), 0, \ldots, 0^{(*)}, \ldots, F^{(M-1)}(t_n), 0, F^{(M)}(t_n)], \end{aligned} \tag{29}$$

where

- $\Theta^{(k)}$ (respectively, $F^{(k)}$) is the restriction of Θ (respectively, F) inside the subdomains Ω_k;
- W^n_k, $k = 1, \ldots, M-1$, is the value of $\Theta(., t_n)$ at the interface P_k, i.e., $W^n_k = \Theta^n_{N,k} = \Theta^n_{0,k+1}$.

We solve the linear system (29) in the following way: first the W_j are solutions of the tridiagonal system of dimension $M-1$:

$$\begin{cases} \vdots \\ (\eta(j) - t\alpha(j)g(j))W_{j-1} + (\gamma(j) - t\alpha(j)h(j) - t\beta(j)g(j+1))W_j \\ +(\delta(j) - t\beta(j)h(j+1))W_{j+1} = -(tg(j)F^{(j)} + th(j)F^{(j+1)}), \\ \vdots \end{cases} \tag{30}$$

where

- $g(j) = A_j^{-1}\alpha(j)$, $h(j) = A_j^{-1}\beta(j)$;

- $tg(j) = tJ\alpha(j)A_j^{-1}$, $th(j) = tJ\beta(j)A_{j+1}^{-1}$.

Next, we obtain the $\Theta^{(j)}$ by solving the M decoupled linear systems:

$$\text{(31)} \qquad \begin{cases} A_1\Theta^{(1)} = F^{(1)} - W_1\hat{\beta}(1) \\ \vdots \\ A_j\Theta^{(j)} = F^{(j)} - (W_{j-1}\hat{\alpha}(j) + W_j\hat{\beta}(j)) \\ \vdots \\ A_M\Theta^{(M)} = F^{(M)} - W_{M-1}\hat{\alpha}(M). \end{cases}$$

Since most of the time in the time marching scheme is spent in building and solving these M previous linear systems, this DD technique is valuable for parallel computation.

However, (31) is not the only source of parallelism in the *adaptive* DD. We remark that because of the DD, most of the adaptivity process can be parallelized. Moving from an old physical grid parametrized by I_{old} to a new grid parametrized by I_{new} requires three steps:

- a minimization process with respect to I such that $\| (\tilde{u}_1, \ldots, \tilde{u}_M) \|_{I_{\text{old}}} > \| (\tilde{u}_1, \ldots, \tilde{u}_M) \|_{I_{\text{new}}}$;
- an interpolation from the old grid to the new grid; and
- a construction of the new operator $\tilde{D}$.

In most cases, the computations are either local to a single domain or depend on two adjacent subdomains. Therefore, M or $(M-1)$ parallel processes are involved in each step.

4. Asymptotics and adaptivity with pseudospectral approximations methods. We will examine several issues in adaptive DD with pseudospectral. We first compare and analyze different strategies to solve a TL.

Then we apply the method to singular perturbation problems including the spiked strong detonation case for the reacting flow problem of Majda.

4.1. What kind of mapping for a transition layer? It is quite clear that the Chebyshev method more easily solves a BL than a TL because of the $O(N^2)$ clustering of the collocation points at both ends of the interval $[-1, 1]$. Also, a mapping of BL type will improve the numerical accuracy of the approximation when boundary layers are solved. However, it is unclear for a TL what will be the best mapping and subdomain technique.

As a matter of fact, let us suppose that u exhibits a single TL at $x_0 \in \Omega$. One can use either:

- two subdomains with their mapping of BL type and their interface in x_0 or
- three subdomains with an inner domain centered in x_0 and its mapping of TL type.

We report here some error results based on the numerical approximation of a typical profile. By typical profile we mean the kind of behavior that we observe in the numerical computation of a reacting shock wave or a combustion problem.

Table 2 (respectively, Table 3) gives the best values of the parameter mapping and interface positions with 30 (respectively, 50) collocation points per subdomain. We approximate in this table the function:

$$h(x) = (1 - \tanh(\xi))/2 + 0.5\exp\left(-\frac{\xi^2}{2}\right); \qquad \xi = \frac{x}{\epsilon}.$$

- When using two subdomains we find the best accuracy in the symmetric case, i.e., an interface at $x = 0$ and the same parameter value for the mapping of BL type inside each subdomain. We observe that the value of α that minimizes the error is of order $\epsilon^{1/2}$.

- When using three subdomains we find the best accuracy in the symmetric case, i.e., the inner domain centered at $x = 0$. We observe that the value of α that minimizes the error is of order ϵ.

We also notice that as a greater fraction of the points enter the layer, the conditioning of the matrix of differentiation deteriorates: for a given arithmetic and a fixed ϵ, we reach easily the limit of accuracy of the method as we increase the number of collocation points; when we increase further the number of collocation points, the conditioning of the matrix of differentiation is so bad that even in double precision, the accuracy of the pseudospectral approximation starts to decrease.

TABLE 2

Best mapping for 2 *and* 3 *subdomains. In this table norm* ϕ (*respectively, norm* 1, *respectively, norm* 2) *is the error in the maximum norm for the function h* (*respectively, the first derivative* h', *respectively, the second derivative* h'').

	epsilon = 0.1	0.01	0.001	
2 domains	5.10^{-9}	$1.6\ 10^{-6}$	2.10^{-5}	norm ∅
	0.4	0.08	0.015	alfa
	4.10^{-7}	5.10^{-4}	2.10^{-2}	norm 1
	0.4	0.05	0.009	alfa
	6.10^{-5}	1	481	norm 2
	0.4	0.07	0.009	alfa
3 domains	1.10^{-6}	$4.7\ 10^{-4}$	$2.7\ 10^{-3}$	norm ∅
	0.2	0.04	0.005	alfa
	$9.6\ 10^{-5}$	0.14	3	norm 1
	0.2	0.04	0.003	alfa
	3	40	6485	norm 2
	0.2	0.03	0.003	alfa

30 collation points per domain

either
2 domains : [-2, 0] [0, 2]
3 domains : [-2, -1] ; [-1, 1] ; [1, 2]

TABLE 3

Best mapping for 2 and 3 subdomains. In this table norm ϕ (respectively, norm 1, respectively, norm 2) is the error in the maximum norm for the function h (respectively, the first derivative h', respectively, the second derivative h'').

	epsilon = 0.1	0.01	0.001	
2 domains	1.10^{-9}	1.10^{-9}	8.10^{-8}	norm ∅
	0.4	0.09	0.016	alfa
	4.10^{-8}	5.10^{-7}	2.10^{-4}	norm 1
	0.4	0.08	0.012	alfa
	2.10^{-6}	1	483	norm 2
	0.4	0.03	0.006	alfa
3 domains	$2.8\ 10^{-10}$	6.10^{-6}	$2.6\ 10^{-4}$	norm ∅
	0.3	0.04	0.005	alfa
	$4.\ 10^{-8}$	4.10^{-3}	0.5	norm 1
	0.2	0.04	0.004	alfa
	$1.8\ 10^{-6}$	1.2	1000	norm 2
	0.2	0.04	0.004	alfa

50 collation points per domain

either

2 domains : [-2, 0] [0, 2]

3 domains : [-2, -1] ; [-1, 1] ; [1, 2]

Tables 2 and 3 show clearly that with the type of layer profile chosen the BL mapping is better than a TL mapping. Let us notice that the same experiment on the hyperbolic tangent profile gives comparable results for both decompositions. In Figs. 4 and 5 we can see the density of discretization points in the physical space that corresponds, respectively, to the results of Tables 2 and 3. These figures show that the width of the layer and the density of collocation points in the layer are bigger in the 2 subdomain case than in the 3 subdomain case.

Let us analyze asymptotically the action of the mapping of BL type when $\alpha = O(\sqrt{\varepsilon})$. It is easy to show that in physical space, the image of the Chebyshev points in the neighborhood of one end $+1$ (respectively, -1) collapses in the BL when the Chebyshev points in the

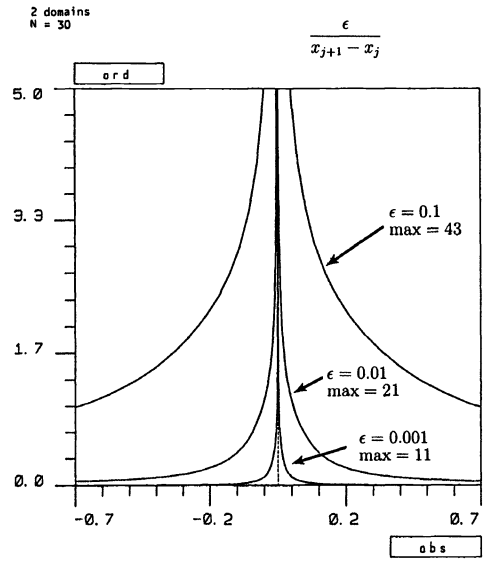


FIG. 4. *Density of points for the best mappings with* 2 *subdomains.*

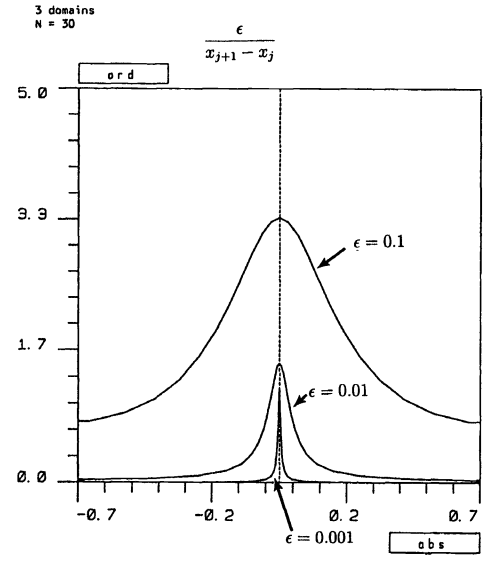


FIG. 5. *Density of points for the best mappings with* 3 *subdomains.*

neighborhood of the other end -1 (respectively, $+1$) spray out in the outer domain. Let Δx^{inner} (respectively, Δx^{outer}) be the order of magnitude of the distance between two points of the discretization in the layer (respectively, in the outer domain). One has:

$$\alpha = O(\sqrt{\varepsilon}) \Rightarrow \Delta x^{\text{inner}} \simeq \frac{\sqrt{\varepsilon}}{N^2}, \qquad \Delta x^{\text{outer}} \simeq \frac{1}{N^2\sqrt{\varepsilon}}.$$

Therefore, this mapping meets the scaling required in the asymptotic expansion of h, i.e.,

$$\Delta x^{\text{inner}} = \varepsilon \Delta x^{\text{outer}}.$$

On the contrary, in the 3 subdomain case, we have:

$$\alpha = O(\varepsilon) \Rightarrow \Delta x^{\text{inner}} \simeq \frac{\varepsilon}{N} \qquad \Delta x^{\text{outer}} \simeq \frac{1}{\varepsilon N^2};$$

therefore, the 3 subdomain case does not satisfy the stretching required by the asymptotic analysis.

In the previous case, the size of each subdomain is of order 1, and we only use a mapping inside a subdomain to solve the layer. But we can also use some subdomain of ϵ order thickness to solve the same layer. We have obtained some good results using the 3 subdomain technique when the size of the inner domain is of order ε. However, in this later case, the accuracy is very sensitive to the localization of the layer.

Remark. The adaptivity scheme relies on the fact that the minimization of a Sobolev type norm gives the best value of the mapping parameter; a large number of numerical experiments on the previous profiles in the one-dimensional case suggest the fact that the following Sobolev norm denoted I_2:

$$I_2(f) = \left[\int_{-1}^{1} \frac{(L^2 f)^2}{(1-x^2)^{1/2}} dx\right]^{1/2},$$

where $L = (1-x^2)^{1/2}(d/dx)$, may underestimate the optimum value of α. We found better approximation of α with the next order Sobolev type norm I_4 [6]:

$$I_4(f) = \left[\int_{-1}^{1} \frac{(L^4 f)^2}{(1-x^2)^{1/2}} dx\right]^{1/2}.$$

For example, with a 2 subdomain strategy to solve a TL with $\varepsilon = 0.1$, when the optimum α is 0.4, one can get $\alpha = 0.3$ with the a priori estimate based on I_4, and $\alpha = 0.2$ with the a priori estimate based on I_2. These examples show that an asymptotic estimation of the best α might be from the practical point of view as efficient as the use of a rigorous a priori estimate. We found the same conclusion in the numerical experiments of the following section.

4.2. Strategy to adapt and split the domain. We illustrate our strategy to adapt and split the domain on three singular perturbation problems.

4.2.1. A two point boundary value problem. Following a suggestion of R. J. O'Malley [23], we apply the adaptive DD technique with pseudospectral approximations on the two point boundary value problem:

$$\begin{cases} \varepsilon y'' + (1-4y^2)y' - 4y = 0, \\ y(0) = 0, \qquad y(1) = -1. \end{cases} \tag{32}$$

This singular perturbation problem has a shock layer of order ε thickness around $x \simeq 0.1$; in addition, the turning point at $y = \frac{1}{2}$ makes the structure of the BL at $x = 1$ rather complex: the mean layer thickness of this multiple layer is of order $\varepsilon^{1/2}$ [23], but the approximation is dominated by a boundary layer analogous to a shock layer of order ε thickness.

A Godunov scheme applied to the corresponding time dependent problem:

$$\begin{cases} -u_t + \varepsilon u_{xx} + (1-4u^2)u_x - 4u = 0, \\ u(0,t) = 0, \qquad u(1,t) = -1, \\ u(x,0) = u_o(x), \end{cases} \tag{33}$$

converges to the solution of (32) *when* one includes in the numerical scheme the viscosity, i.e.:

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \varepsilon \frac{u_{i-1}^n - 2u_i^n + u_{i+1}^n}{\Delta x^2} + \frac{F_{i+1/2}^n - F_{i-1/2}^n}{\Delta x} - 4u_i^n, \tag{34}$$

where $F_{i\pm 1/2}^n$ is the flux of the Godunov scheme. We refer to [1] for the numerical analysis of finite difference schemes that solve (32).

In our numerical experiment, we use as an initial condition $u_o(x)$, a numerical approximation of the steady solution of (33) given by this finite difference scheme with $\frac{\varepsilon}{\Delta x}$ of order 1; Fig. 6 shows this initial condition interpolated to the pseudospectral grid; the finite difference scheme is used with one hundred points and one thousand timesteps. Figure 6 looks good, however, since the numerical viscosity of the Godunov scheme interacts with the given viscosity εu_{xx}, the numerical approximation does not really solve for the layers with the given ε.

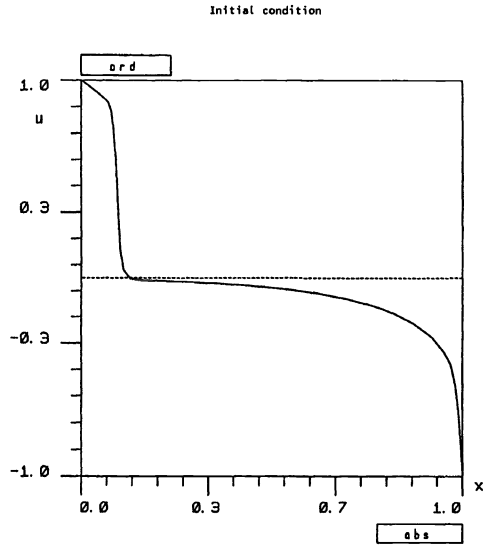


FIG. 6. *L. Abrahamsson and S. Osher* [1] *problem with* $\varepsilon = 0.01$ *and* $\varepsilon = 0.001$: *initial condition obtain with a modified Godunov scheme.*

Figures 7 and 8 show the result of the adaptive DD method with $\varepsilon = 0.01$ and $\varepsilon = 0.001$. Our numerical criteria for terminating the time marching scheme of the adaptive DD scheme is:

$$\max_i \left\| \frac{u_i^{n+1} - u_i^n}{\Delta t} \right\| = o(\varepsilon).$$

We use either the BL mapping (situation A) or the TL mapping (situation B) strategy to solve the shock layer. We use a BL mapping to solve the BL at the right end of the interval. For the initial condition $u_o(x)$, the shock layer can be identified by computing the viscosity and/or the residual. Based on this criteria, we find an approximate location of the layer at $x_o = 0.09$. In situation A (respectively, B) we start with the DD: $Q(0) = 0$, $Q(1) = x_o$, $Q(2) = 0.7$, $Q(3) = 1$ (respectively, $Q(0) = 0$, $Q(1) = x_o$, $Q(2) = 2x_o$, $Q(3) = 0.7$, $Q(4) = 1$). Then the best interface positions are computed adaptively.

With $\varepsilon = 0.01$, we obtain for the steady solution:

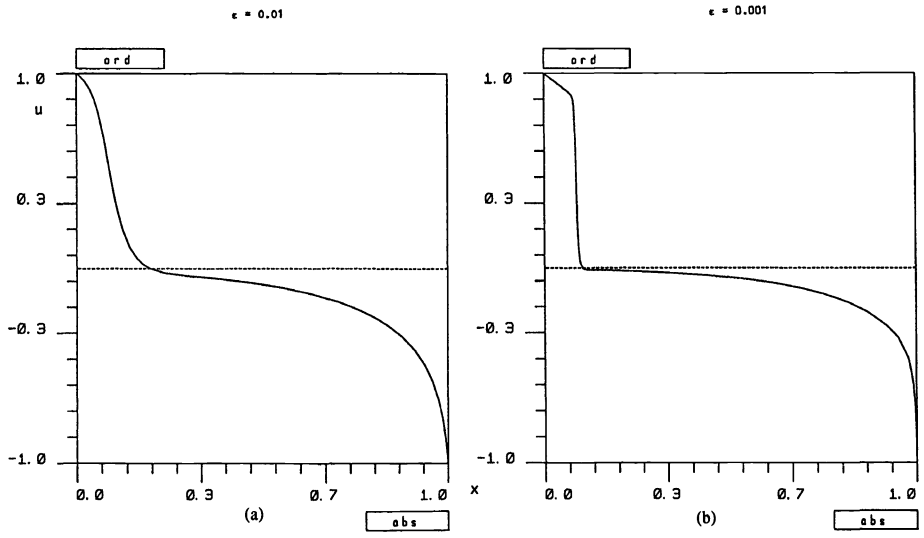


FIG. 7. *L. Abrahamsson and S. Osher* [1] *problem with* $\varepsilon = 0.01$ *and* $\varepsilon = 0.001$: *steady solution.*

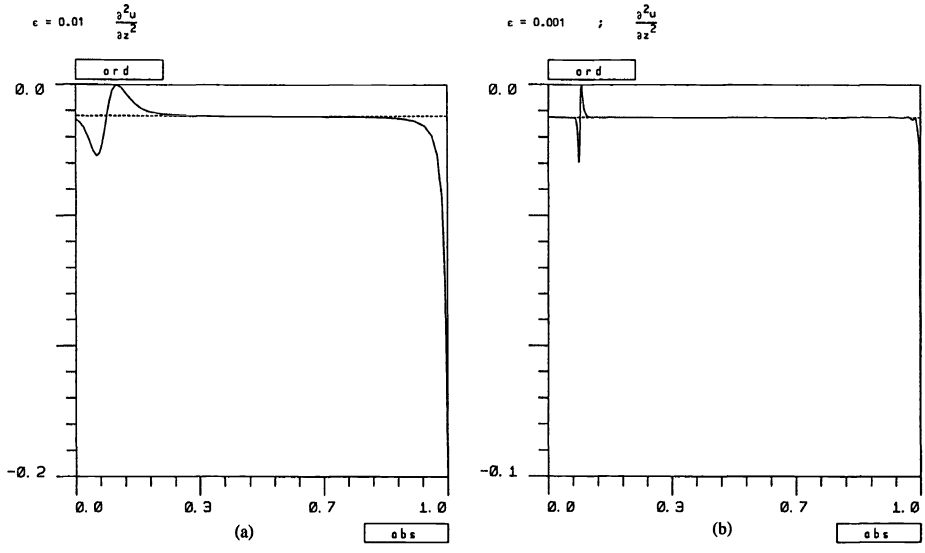


FIG. 8. *L. Abrahamsson and S. Osher* [1] *problem with* $\varepsilon = 0.01$ *and* $\varepsilon = 0.001$: *second derivative of the steady solution.*

- in situation A, $Q(1) = 0.170, \alpha_1^* = 0.324$ with 12 collocation points per subdomain;
- in situation B, $Q(1) = 0.0787, \alpha_1^{**} = 0.58$, $Q(2) = 0.1787, \alpha_2 = 0.42$ with 12 collocation points per subdomains.

We notice that the adaptive choice of the interfaces are in agreement within ϵ of the choice of the interface position based on the asymptotic criterion. The ratio of α_1^*/α_1^{**} is also in agreement with the asymptotic analysis of §4.1.

We have solved (32) with $\varepsilon = 10^{-3}$; we obtain for the steady solution:

- in situation A, $Q(1) = 0.170, \alpha_1^* = 0.324$ with 15 collocation points per subdomain;
- in situation B, $Q(1) = 0.0787, \alpha_1^{**} = 0.58$, $Q(2) = 0.1787, \alpha_2 = 0.42$ with 12 collocation points per subdomains.

We have checked that both situations A and B give comparable results, i.e., we have a very good agreement on the maximum value of u_x and u_{xx}. Also, Fig. 8 shows that the interface between subdomains do not induce significant numerical problems.

We notice that the adaptive choice of the interface is in agreement within ϵ of the choice of the interface position based on the asymptotic criterion. We notice that the subdomain technique stretches ϵ to roughly 10ϵ for the first layer in the subdomain $[Q(0), Q(1)]$: that explains why we found in our experiment α_1^*, α_2 (respectively, α_1^{**}, α_2) roughly in agreement with the column $\epsilon = 0.1$ (respectively, $\epsilon = 0.01$) of Tables 2 and 3.

Finally, we have remarked that the adaptive choice of α based on the a priori estimates ([6]) has the tendency to underestimate α. In our numerical experiment it is especially clear for the last subdomain that solves the BL at the right end of the interval. In the adaptive scheme, we had to impose a lower bound to the last α. During the convergence process to the stationary solution, we have seen that most of the time evolution occurs in the layers. It suggests that we should use different timesteps for different subdomains.

4.2.2. Majda's simplified model of reacting flow. We have applied the DD technique with pseudospectral approximation to the initial boundary value problem (16) on the domain $[a, b]$ with the initial condition:

$$u(x,0) = (U_l - U_r)\frac{1-\tanh(\xi)}{2} + U_r; \qquad \xi = \frac{x}{\varepsilon},$$

and the boundary conditions

$$U(a) = U_l; \qquad U(b) = U_r.$$

We report here some numerical experiments of the spiked strong detonation case discussed in §2.3 with $U_l = 1$, $U_r = -1.5$, $q_0 = 2.375$, and $a = -6$, $b = 8$. Our numerical scheme solves explicitly the nonlinear convection term $(u^2/2)_x$; therefore, the timestep is limited by the Courant–Friedrish–Levy (CFL) condition. We found it more accurate, robust, and easy to adopt the two subdomain strategy to solve the reactive shock layer. Also, it is very convenient and efficient to move the interface by tracking the maximum value of the Arrhenius term $q_o\varepsilon^{-1}\Phi(u)Z$. The numerical method solves the initial formation of the reactive shock layer and then as t increases, the solution converges to the travelling wave (17).

We have obtained a numerical error of order 1% on the maximum value of u and on the speed of propagation of the travelling wave for $\varepsilon = 0.1$ (respectively, $\varepsilon = 0.01$) with 20 (respectively, 42) collocation points per subdomain (see Fig. 9, respectively, Fig. 10). We observe that the method is conceptually very well adapted to solve the initial formation of the layers as well as interaction of singularities. However, because of the CFL constraint and because the minimum distance between two discretization points has to be of order less than ε in the layer, this method is computationally expensive when solving for a traveling wave.

Figure 11 shows the initial formation of a strong detonation.

We have also solved a weak detonation case (see Fig. 12) with $U_l = 1$, $U_r = -0.4$, and $q_0 = 0.568$.

A further step in our research is to implement a DD method that is a mixture of both methods presented in this paper. Strong singularities as simple shock or reactive shock that are easily identified can be solved by the first DD technique presented in this paper. Interaction of singularities or initial layers may be solved by the adaptive DD based on pseudospectral approximation methods. In addition, at each stage of the computation, we use asymptotic analysis to select the best method.

Let us mention that to extend the method to solve problems in two space dimensions, we may use DD to solve layers as well as the difficult problem of geometry due to the curvature of

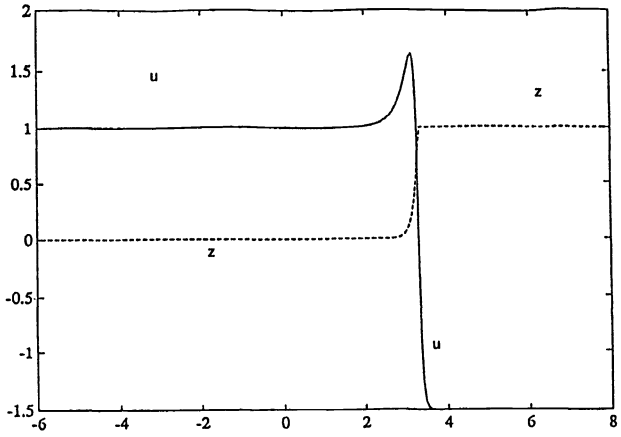


FIG. 9. *Strong detonation with* $\varepsilon = 0.1$.

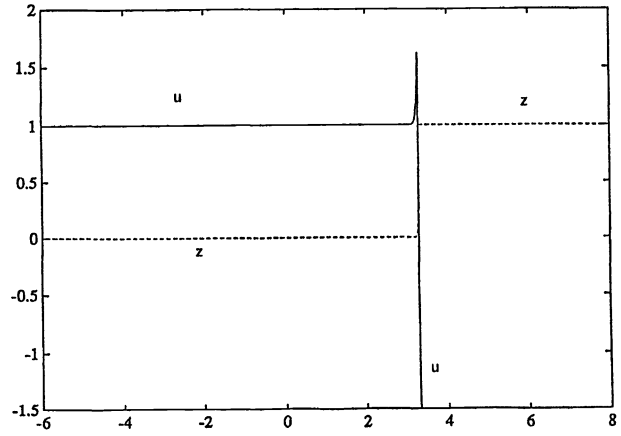


FIG. 10. *Strong detonation with* $\varepsilon = 0.01$.

the fronts. We expect both DD methods described here to be useful tools for parallel computing of singular perturbed problem in two space dimensions. Finally, we shall illustrate an easy use of asymptotics in the numerical computation of a nontrivial combustion problem.

4.2.3. A combustion problem with two thin flame fronts. We look for the long time behavior of a thermal diffusive model of combustion with a two step reaction: $Y \to X \to P$. We use as a test problem for our DD the following model see [24] with *separate reaction* layers:

$$\begin{cases} X_t = X_{rr}/L_1 - (K - 1/L_1)X_r - Da_1 X R_1(\Theta), \\ Y_t = Y_{rr}/L_2 - (K - 1/L_2)Y_r + Da_1 X R_1(\Theta) - Da_2 Y R_2(\Theta), \\ \Theta_t = \Theta_{rr} - (K - 1)\Theta_r + Da_1/2 X R_1(\Theta) + Da_2/2 Y R_2(\Theta). \end{cases} \tag{35}$$

This model is a system of reaction diffusion equation of dimension 3. Two equations give the evolution of the concentrations for each component Y and X. The third equation gives the evolution of the temperature Θ. K is a given constant. Typically, for this class of problem the small singular perturbation parameter(s) is (are) the inverse of the activation energy of the

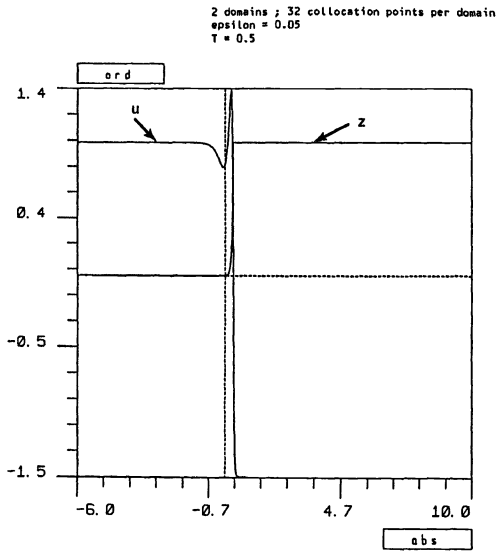


FIG. 11. *Initial formation of a strong detonation with* $\varepsilon = 0.05$.

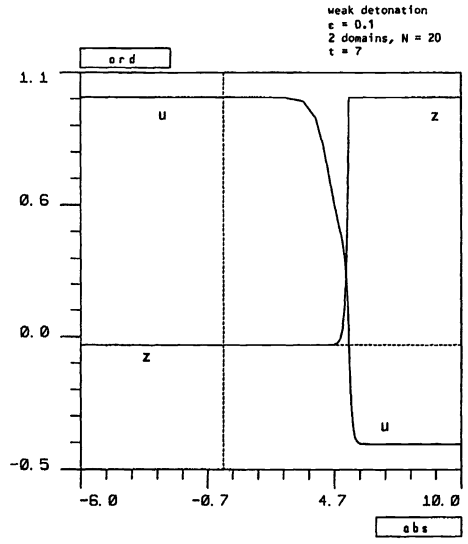


FIG. 12. *Weak detonation with* $\varepsilon = 0.1$.

reaction(s): consequently, the coefficients $Da(1)$ and $Da(2)$ are large numbers. The source terms R_1 and R_2 are exponential nonlinearities given by the Arrhenius law. We refer to [24] and [7] for the precise statement of the problem. This model belongs to the general model problem (1) modulo eventually a rescaling for long time behavior.

Under the assumption of large activation energy, the asymptotic analysis of this model indicates that the transition layers coincide with the sharp zones where the Arrhenius terms are of the order one strictly. We base our choice to move the interfaces on this general criteria.

In our numerical experiment, we take the Lewis number and activation energy to be the same for each reaction. We have chosen the Lewis number to be one, and we have chosen as a large activation number N=22. We compute the radial symmetric solution with a radius bounded by 1 and 41. We know that the thickness of the two layers is of order $(\frac{1}{N})$. Figure 13 gives an approximation of the steady solution.

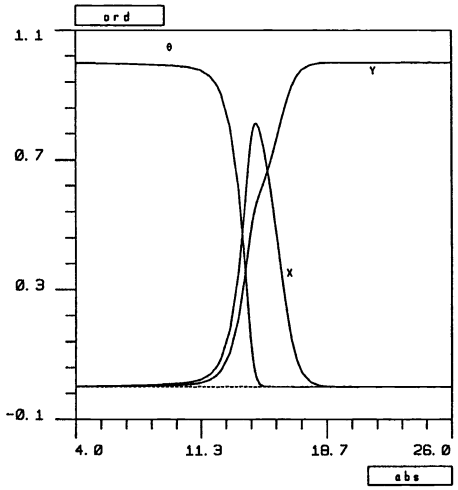


FIG. 13. *Pelaez problem: steady radial symmetric solution.*

The strategy used to solve this problem is to use six subdomains: two of them are outer domains and we use 2 subdomains to solve each layer with a boundary layer mapping. We need 12 collocation points per domain. Figure 14 shows the Arrhenius source terms $Da_i * R_i$ for the first and second reaction. The distance between the maximum of each Arrhenius term is in agreement with the distance predicted by the asymptotic analysis of Pelaez. Figure 15 shows that most of the dynamics happen in the combustion layers, as it should.

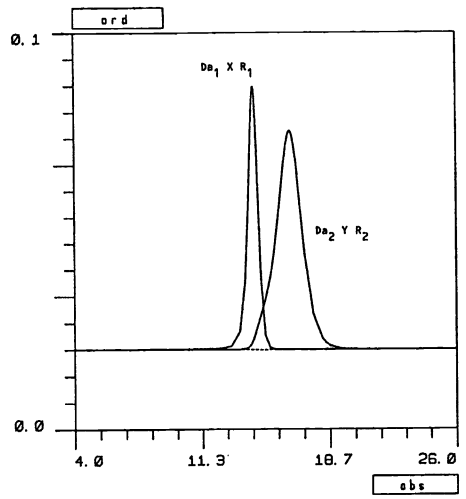


FIG. 14. *Pelaez problem: Arrhenius terms.*

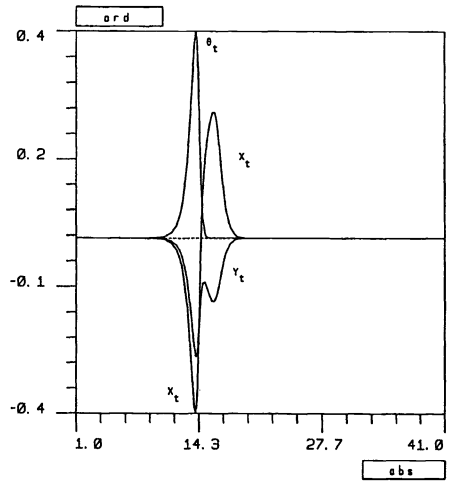


FIG. 15. *Pelaez problem: graph of* $10^4(u^{n+1} - u^n)/\Delta t$.

The very important point is the fact that we have chosen to fix the interfaces between the subdomains by tracking the maximum value of the Arrhenius terms for each reaction: more precisely, we impose $Q(2)=Q(3)$-1, $Q(6)=Q(5)$+1, $Q(3)$ to be at the maximum of the first reaction, $Q(5)$ to be at the maximum of the second layer and $Q(4)=(Q(3)+Q(5))/2$. We update the position of the interfaces depending on their speed. Our experiments suggest that for this case, it is not necessary and, in fact, more difficult to adapt the position of the interfaces based on the a priori estimates of §3.2. This experiment shows again on a nontrivial problem that an asymptotic understanding of the nature of the problem can decrease the cost of adaptivity. The numerical computation of this combustion problem can be extended in two space dimensions. This joint work with A. Bayliss and B. J. Matkowsky is reported in [7].

Acknowledgments. The author is very grateful to A. Bayliss and B. J. Matkowsky who introduced me in the field of adaptive domain decompositions with pseudospectral approximation. Thanks also to A. Harten and R. J. O'Malley who suggested that I study, respectively, the Majda model and the turning point problem mentioned in this paper.

REFERENCES

[1] L. ABRAHAMSSON AND S. OSHER, *Monotone difference schemes for singular perturbation problems*, SIAM J. Numer. Anal., 19 (1982), pp. 979–992.

[2] J. BARANGER AND EL AMRI, *Numerical solution of a spectral problem for an* ODE *with a small parameter using an asymptotic expansion and a finite element method*, Numer. Funct. Anal. Optim., 11 (1990), pp. 621–642.

[3] C. BASDEVANT, M. DEVILLE, P. HALDENWANG, J. M. LACROIX, J. OUAZZANI, R. PEYRET, P. ORLANDI, AND A. T. PATERA, *Spectral and finite difference solutions of the Burgers equation*, Comput. & Fluids, 14 (1986), pp. 23–41.

[4] A. BAYLISS AND E. TURKEL, *Mappings and accuracy for Chebyshev pseudospectral approximations*, J. Comput. Phys., to appear.

[5] A. BAYLISS, J. BELYTSCHKO, D. HANSEN, AND E. TURKEL, *Adaptative Multidomain Spectral Methods*, in Proc. 5th International Symposium on Domain Decomposition Methods for Partial Differential Equations, Norfolk, VA, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 195–203.

[6] A. BAYLISS, D. GOTTLIEB, B. J. MATKOWSKY, AND M. MINKOFF, *An adaptive pseudo-spectral method for reaction diffusion problems*, J. Comput. Phys., 81 (1989), pp. 421–443.

[7] F. DESPREZ AND M. GARBEY, *Parallel Computation of a Combustion Problem*, Equipe d'analyse Numerique, Lyon, St. Etienne, November, 1993, preprint 160; Parallel Comput., submitted.

[8] A. BOURGEAT AND M. GARBEY, *Computation of viscous (or nonviscous) conservation law by domain decomposition based on asymptotic analysis*, Numer. Methods Partial Differential Equations, 8 (1992), pp. 127–142.

[9] P. BJORSTAD AND O. B. WIDLUND, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal., 23 (1986), pp. 1097–1120.

[10] R. C. CHIN, G. W. HEDSTROM, J. R. MCGRAW, 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, 1986, pp. 136–153.

[11] R. C. CHIN AND R. KRASNY, *A hybrid asymptotic-finite element method for stiff two points boundary value problems*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 229–243.

[12] C. CONLEY AND J. SMOLLER, *Topological methods in the theory of shock waves*, in Proc. Symposium on Pure Mathematics, Providence, RI, 1973, pp. 233–302.

[13] U. EHRENSTEIN, H. GUILLARD, AND R. PEYRET, *Flame computations by a Chebyshev multidomain method*, Internat. J. Numer. Methods Fluids, 9 (1989), pp. 499–515.

[14] W. ECKHAUS, *Asymptotic Analysis of Singular Perturations*, North-Holland, Amsterdam, 1979.

[15] P. C. FIFE, *Dynamics of internal layers and diffusive interfaces*, in CBMS-NF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 1988.

[16] M. GARBEY, *Asymptotic analysis of singular perturbation governed by a conservation law*, MCS-P107-1089, MCS, Argonne Nat. Lab., 1989, preprint.

[17] M. GARBEY AND J. S. SCROGGS, *Asymptotic-induced numerical methods*, in Asymptotic Analysis and the Numerical Method of PDE, Lecture Notes in Pure and Applied Mathematics 130, Dekker, New York, Basel, Switzerland, 1991, pp. 75–96.

[18] D. GOTTLIEB AND R. S. HIRSH, *Parallel Pseudospectral Domain Decomposition Techniques*, Tech. report 83, ICASE, Hampton, VA.

[19] J. LORENZ, *Nonlinear boundary value problems with turning points and properties of difference schemes*, Lecture Notes in Mathematics, 942, Springer-Verlag, New York, 1982, pp. 150–169.

[20] A. J. MAJDA, V. ROYTBURD, AND P. COLLELA, *Theoretical and numerical structure for reacting shock waves*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1059–1080.

[21] R. E. O'MALLEY, *Singular perturbation, asymptotic evaluation of integrals and computational challenges*, in Asymptotic Analysis and the Numerical Method of PDE, Lecture Notes in Pure and Applied Mathematics 130, Marcel Dekker, New York, Basel, Switzerland, 1991, pp. 3–17.

[22] R. E. O'MALLEY, JR., *Shock and transition layers for singularly perturbed second order vector systems*, SIAM J. Appl. Math., 43 (1983), pp. 935–943.

[23] ———, *private communication.*

[24] J. PELAEZ, *Stability of premixed flames with two thin reactions layers*, SIAM J. Appl. Math., 47 (1987), pp. 781–799.

[25] R. PEYRET, *The Chebyshev multidomain approach to stiff problems in fluid dynamics*, Comput. Methods Appl. Mech. Engrg., 80 (1990), pp. 129–145.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 892–915, July 1994

007

LOCAL PIECEWISE HYPERBOLIC RECONSTRUCTION OF NUMERICAL FLUXES FOR NONLINEAR SCALAR CONSERVATION LAWS*

ANTONIO MARQUINA†

Abstract. This paper constructs a local third order accurate shock capturing method for hyperbolic scalar conservation laws, based on numerical fluxes with a total variation diminishing (TVD) Runge–Kutta evolution in time, using the idea recently introduced by C. W. Shu and S. J. Osher for essentially nonoscillatory (ENO) methods. The constructed method is an upwind conservative scheme that is local in the sense that numerical fluxes are reconstructed without using extrapolation from the data of the smoothest neighboring cell. To design the method, a new concept of local smoothing is introduced to prevent the increasing of total variation of the solution near discontinuities and to achieve third order accuracy. The method becomes third order accurate in smooth regions of the solution, except at local extrema where it may degenerate to $O(h^{3/2})$, thus giving better accuracy than TVD methods at local extrema. The main advantage of this method lies on the property that is more local than that of ENO and TVD upwind schemes of the same order, (and, thus, giving better resolution of corners, i.e., jumps in derivative), because numerical fluxes depend only on four variables. Numerical experiments for scalar conservation laws one-dimensional (1D) and two-dimensional (2D) are presented, and they show that the author's method is stable and behaves as an entropy-satisfying method for nonlinear fluxes. This method becomes efficient since it is low cost and it is not sensitive to the Courant–Friedrichs-Lewy (CFL) number.

Key words. nonlinear conservation laws, Runge–Kutta, total variation diminishing, essentially nonoscillatory

AMS subject classifications. primary 65M05; secondary 65M10

1. Introduction.

In this paper, we consider numerical approximations to weak solutions of the scalar initial value problem

$$u_t + f(u)_x = 0, \tag{1}$$

$$u(x, 0) = u_0(x). \tag{2}$$

The initial data $u_0(x)$ are supposed to be piecewise-smooth functions that are either periodic or of compact support.

Let $u_j^n = u_h(x_j, t_n)$ denote a numerical approximation to the exact solution $u(x_j, t_n)$ of (1) and (2) defined on a computational grid $x_j = jh$, $t_n = n\Delta t$ in conservation form:

$$u_j^{n+1} = u_j^n - \lambda(\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}}), \tag{3}$$

where $\lambda = \frac{\Delta t}{h}$, the numerical flux is a function of $2k$ variables

$$\hat{f}_{j+\frac{1}{2}} = \hat{f}(u_{j-k+1}^n, \ldots, u_{j+k}^n), \tag{4}$$

which is consistent with (1), i.e.,

$$\hat{f}(u, \ldots, u) = f(u). \tag{5}$$

*Received by the editors September 18, 1989; accepted for publication (in revised form) September 17, 1993. This research was supported by a grant from Conselleria de Cultura, Educació i Ciéncia de la Generalitat Valenciana and in part by Dirección General de Investigación Cientifica y Tecnológica grant PS90-0265. Computer time was supported by Office Of Naval Research Grant N00014-86-K-0691.

†Departamento de Matemática Aplicada Universitat de Valencia 46100-Burjassot (Valencia), Spain and Depart-break ment of Mathematics, University of California, Los Angeles, California 90024-1555 (marquina@godella.matapl.uv.es).

The *total variation* of a discrete solution is usually defined by

$$TV(u^n) = \sum_j |u^n_{j+1} - u^n_j|. \tag{6}$$

This is the total variation with respect to x of the numerical approximation $u_h(x,t)$ in (3) considered as a piecewise-constant function defined by

$$u_h(x,t) = u^n_j \tag{7}$$

for $x_{j-1/2} < x < x_{j+1/2}$ and $n\Delta t < t < (n+1)\Delta t$, where $x_{j+1/2} = (j + \frac{1}{2})h$.

We say scheme (3) is TVB (total variation bounded), if there exists a constant $M > 0$, independent of h for $0 \le t \le T$ (T fixed), such that

$$TV[u_h(\cdot\,,t)] \le M \cdot TV(u_0). \tag{8}$$

If $M = 1$, the scheme is called TVD.

The importance of TVB and TVD methods relies on the fact that any refinement sequence $h \to 0$, u_h has a subsequence h_n, such that u_{h_n} L^1-converges to a weak solution of (1) and (2). The notion of the TVD scheme was introduced by Harten in [3], where he constructed TVD schemes that in the sense of truncation error are high order accurate everywhere except at local extrema, where they necessarily degenerate into first order accuracy, (see [2], [3], and [7]).

To get high order accuracy in smooth regions, Harten, Osher, Engquist, and Chakravarthy constructed ENO schemes that use adaptive stencils obtaining information from regions of smoothness if discontinuities are present. The analysis and numerical experiments of these methods can be found in [4]–[6]. However, the most efficient implementation of ENO methods has been investigated by C.W. Shu and S.J. Osher in the remarkable papers [9] and [10], where they reconstructed *numerical fluxes* from *point values* by applying the adaptive idea of the ENO interpolation and using a TVD Runge–Kutta-type time discretization. Originally, ENO schemes were based on the reconstruction of the *solution* from *cell averages*. The following lemma (taken from [10]) establishes a useful relation between the numerical flux and the flux of a solution to (1).

LEMMA 1.1 (Shu and Osher). *If a function $g(x)$ satisfies*

$$f(u(x)) = \frac{1}{h}\int_{x-\frac{h}{2}}^{x+\frac{h}{2}} g(\xi)\,d\xi, \tag{9}$$

then

$$f(u(x))_x = \frac{g(x+\frac{h}{2}) - g(x-\frac{h}{2})}{h}.$$

Lemma 1.1 implies that to approximate the numerical flux $\hat{f}_{j+1/2}$ to a high order accuracy, it is enough to reconstruct $g(x_{j+1/2})$ up to the same order. The reconstruction "via primitive function" (see [5]) seems to be the most efficient one and it is used in [10] in an ENO fashion.

In this paper we construct a local third order accurate method by using a piecewise hyperbolic reconstruction of the function g in (9) instead of the polynomial ENO interpolation of the primitive function of g as in [10]. The evolution in time is performed by means of the Shu–Osher third order TVD Runge–Kutta method (see [10]) computed by the recurrence formula

$$u^{(i)}_j = \sum_{k=0}^{i-1}\left[\alpha_{ik}\cdot u^{(k)}_j + \beta_{ik}\cdot(-\lambda)\cdot(\hat{f}^{(k)}_{j+\frac{1}{2}} - \hat{f}^{(k)}_{j-\frac{1}{2}})\right], \qquad i = 1,2,3, \tag{10}$$

where

$$u_j^{(0)} = u_j^n, \qquad u_j^{(3)} = u_j^{n+1} \tag{11}$$

with coefficients of Table 1. The one time-stepping procedure described in this paper is *assumed* to be total variation stable for scalar 1D nonlinear problems under a suitable CFL restriction

$$\lambda = \frac{\Delta t}{h} \leq \lambda_0 \tag{12}$$

and λ_0 is inversely proportional to $\max |f'(u)|$; as usual, (i.e., $\lambda_0 \max |f'(u)|$ is less than one for this scheme). This stability cannot be proven for third order ENO or for our method, but there is some theoretical and numerical evidence to indicate that the methods are indeed stable as we shall see later (see [4]–[6], [9], and [10] for details about ENO methods). The obtained method is *local* in the sense that the numerical flux depends only on four-point values in contrast with the corresponding third order accurate ENO that uses six-point values. When discontinuities are present, some smoothing becomes necessary to prevent the increase of the total variation of the solution. This is achieved in our method by means of a "preprocessing of derivatives" carried out in every computational cell as explained in §2. The method is constructed using the Roe entropy-fix framework as presented in [10]. It becomes third order accurate in smooth regions except at local extrema where it may degenerate to $O(h^{3/2})$ due to the shifting of the fluxes and the monotonic character of the reconstruction, which may degenerate into second order accuracy along the cell of transition at local extrema (see Tables 2 and 3). We present two piecewise hyperbolic reconstructions: the first one is satisfactory for contact discontinuities and is unstable for nonlinear fluxes and consists of natural hyperbolas without "preprocessing" of derivatives; the second one is satisfactory for all fluxes and consists of hyperbolas with "preprocessed derivatives." Concerning the method built from the second reconstruction, the following features were found through numerical experiments (see §4):

(a) third order accuracy in smooth regions of the solution except at local extrema where it may degenerate to $O(h^{3/2})$;

(b) correct position and speed of discontinuities;

(c) good resolution of linear discontinuities and jumps of derivatives, where the smearing appears to be more local than in ENO methods and upwind TVD schemes of the same order with a satisfactory behavior for high CFL (see Figs. 1(a), 3(c), and 6(c));

(d) The artificial compression method (see [10] and [11]) for linear discontinuities works efficiently in 1D and 2D linear problems with low CFL (as in ENO methods) (see Figs. 1(c), 2(b), and 3(d)). The artificial compression method is a procedure used to sharpen linear discontinuities in a way that preserves monotonicity.

TABLE 1
Third order TVD *Runge–Kutta scheme* (10).

α_{1k}	α_{2k}	α_{3k}	β_{1k}	β_{2k}	β_{3k}
1	0	0	1	0	0
$\frac{3}{4}$	$\frac{1}{4}$	0	0	$\frac{1}{4}$	0
$\frac{1}{3}$	0	$\frac{2}{3}$	0	0	$\frac{2}{3}$

Our method behaves as a TVB method, as the third order ENO method (as it appears in [10] with r=2), but we have not yet been able to prove this property. However, we have found some theoretical evidence about that problem from a weaker property, which we called "local

TVB property", satisfied by our method (and also by the above-mentioned third order ENO method) and very easy to check. A maximum principle appears to be necessary to prove the TVB property of the scheme.

The paper is organized as follows: §2 contains the reconstruction step, the complete algorithm is presented in §3, and §4 contains numerical experiments including 1D nonconvex Riemann problems, 1D and 2D contact (linear) discontinuities, and a Riemann problem for 2D Burgers' equation.

Before going further we must pay attention to the following two issues.

(i) The upwind scheme presented in this paper (as well as the Shu–Osher ENO scheme) is based on flux point values and is implemented for a uniform fixed grid. We alert the readers who try to implement our scheme for adaptive grids by pointing out that we do not know if our approach is still valid in that case. Some experiments for nonuniform fixed grids are still under investigation.

(ii) We compare our scheme with the third order ENO scheme based on point values introduced by Shu and Osher in [10] through numerical experiments. The reader must take into account that the features encountered for our scheme are always relative to the order of the method and the number of grid points chosen as well as to the spatial stepsize and the CFL number. We point out that to be concise in our study, we omitted the comparison with other remarkable shock-capturing methods such as the Colella and Woodward piecewise parabolic method (PPM) [1].

2. Piecewise hyperbolic reconstruction. The most important step in our method, as well as in ENO methods, is the reconstruction step. Since we look for the reconstruction of the function g in (9) up to third order accuracy in the sense of truncation error, we define the grid data and the reconstruction procedures.

Let $g(x)$ be a piecewise smooth function that is either periodic or of compact support. We have defined a computational grid $x_j = jh$, j integer, $h > 0$, where the cells are

$$C_j = \{x : x_{j-\frac{1}{2}} \le x \le x_{j+\frac{1}{2}}\}, \tag{13}$$

where $x_{j+1/2} = x_j + \frac{h}{2}$. Our grid data are (i) for every j the mean value of $g(x)$ in C_j, v_j is given, i.e.,

$$v_j = \frac{1}{h}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} g(\xi)\,d\xi. \tag{14}$$

(ii) For every j, $d_{j+1/2}$ is given, which is either $g'(x_{j+1/2})$ or

$$d_{j+\frac{1}{2}} = \frac{v_{j+1} - v_j}{h} \tag{15}$$

($d_{j+1/2} = g'(x_{j+1/2}) + O(h^2)$). For our purposes we suppose that (15) is satisfied all the time.

Let $\mathfrak{R}$ be a class of elementary functions. We are only concerned with third order accurate reconstructions, i.e., for every j we must look for r_j in $\mathfrak{R}$, defined on C_j, such that $r_j(x)$ reconstructs $g(x)$ on C_j up to third order accuracy. When we say third order accurate reconstruction, we mean that every time $g(x)$ is smooth enough at x in C_j, then

$$g(x) - r_j(x) = O(h^3). \tag{16}$$

To get consistent third order accurate reconstruction procedures from the grid data with different classes of functions $\mathfrak{R}$, we require that the following two conditions be satisfied for every j:

$$v_j = \frac{1}{h}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} r_j(\xi)\,d\xi, \tag{17}$$

$$d_{j+\frac{1}{2}} = r_j'(x_{j+\frac{1}{2}}). \tag{18}$$

Taylor series expansion shows that (17) and (18) imply third order accuracy.

Our discussion includes the following two classes of elementary functions:

(i) The class of parabolas of the form

$$p_j(x) = a_j + b_j \cdot (x - x_j) + \left(\frac{c_j}{2}\right) \cdot (x - x_j)^2 \tag{19}$$

defined on C_j. We denote this class by $\mathfrak{R}_p$.

(ii) The class of hyperbolas of the form

$$r_j(x) = a_j + \frac{\lambda_j}{(x - x_j) + c_j} \tag{20}$$

defined on C_j. We denote this class by $\mathfrak{R}_h$.

We are interested in reconstructions with the following property.

DEFINITION 2.1. *We say a method of reconstruction* $\{r_j\}$ *of* $g(x)$ *is* local *if for every* j *the function* r_j *depends only on* v_j, $d_{j-1/2}$ *and* $d_{j+1/2}$.

The simplest example of a *local* third order method of reconstruction is the local parabolic reconstruction (LPR). Indeed, for every j there is a unique parabola of the form (17) defined on C_j, and determined from v_j, $d_{j-1/2}$, and $d_{j+1/2}$ by

$$c_j = \frac{d_{j+\frac{1}{2}} - d_{j-\frac{1}{2}}}{h}, \tag{21}$$

$$b_j = \frac{d_{j+\frac{1}{2}} + d_{j-\frac{1}{2}}}{2}, \tag{22}$$

$$a_j = v_j - c_j \cdot \frac{h^2}{24}. \tag{23}$$

We use the following notations:

$$D_j = d_{j+\frac{1}{2}} - d_{j-\frac{1}{2}}, \tag{24}$$

$$d_j = \frac{d_{j+\frac{1}{2}} + d_{j-\frac{1}{2}}}{2}. \tag{25}$$

If d_j is nonzero, then we define the adimensional parameter

$$\alpha_j = \frac{D_j}{2 \cdot d_j}. \tag{26}$$

Obviously, we can also determine the parabola from v_j, d_j, and D_j through (21)–(23).

The reconstruction procedure is repeated at every timestep, and, therefore, the change in total variation of the reconstruction must be controlled. The *local total variation* of the reconstruction $\{r_j\}$ is defined by

$$LTV_j = TV(r_j), \tag{27}$$

where $TV(r_j)$ means the total variation of the function $r_j(x)$ in the cell C_j. The size of LTV_j determines locally the increasing of the total variation of the reconstruction. Then, we introduce the following definition.

DEFINITION 2.2. *A method of reconstruction is* local total variation bounded (LTVB) *if there exists a constant $M > 0$, independent of h (depending only on the function $g(x)$ to be reconstructed), such that*

$$LTV_j \leq M \cdot h \quad \text{for all } j. \tag{28}$$

The LTV for LPR is

$$TV(p_j) = \frac{h}{2} \cdot \left(|d_{j-1/2}| + |d_{j+1/2}| - e(j) \cdot \frac{2 \cdot |d_{j-\frac{1}{2}}| \cdot |d_{j+\frac{1}{2}}|}{|d_{j+\frac{1}{2}}| + |d_{j-\frac{1}{2}}|} \right), \tag{29}$$

where $e(j) = 0$ if $d_{j-1/2} \cdot d_{j+1/2} \geq 0$ and $e(j) = 1$ otherwise. If discontinuities are present in $g(x)$, then for some j, $d_{j+1/2} = O(h^{-1})$, and therefore (28) is not satisfied for that j. Thus, the LPR method is not LTVB. We have seen, through our numerical experiments, that when using the LPR method for Burgers' equation with smooth data with the algorithm of §3, the total variation of the numerical solution blows up near discontinuities (when they appear). The LTVB condition appears to be necessary for a method to own enough "smoothing" to deal with discontinuities and to behave as a TVB method.

Thus we must look for methods of reconstruction satisfying Definition 2.2 to prevent the increase of total variation. Since the total variation of the function $g(x)$ on a cell C_j where it is smooth depends essentially (up to $O(h^3)$) on the size of $d_{j+1/2}$ and $d_{j-1/2}$ (this follows from the trapezoidal rule), then to get local total variation bounded methods of reconstruction, it is necessary to correct the values of $d_{j+1/2}$ and $d_{j-1/2}$, on every cell, preserving the third order accuracy of the reconstruction. For this purpose we introduce the following definition.

DEFINITION 2.3. *A* numerical left derivative *is a function of $2k + 2$ variables ($k \geq 0$)*

$$dl_j = dl(d_{j-k-\frac{1}{2}}, \ldots, d_{j+k+\frac{1}{2}}), \tag{30}$$

which is "consistent" in the following sense:

$$dl_j - d_{j-\frac{1}{2}} = O(h^2). \tag{31}$$

The concept of numerical right derivative dr_j is defined analogously and satisfies

$$dr_j - d_{j+\frac{1}{2}} = O(h^2) \tag{32}$$

instead of (31). The numerical central derivative d_j^* is also defined in the same way by using

$$d_j^* - d_j = O(h^2), \tag{33}$$

where d_j is defined by (25). The numerical difference, D_j^*, is defined also as a function of the same variables and the following consistency property:

$$D_j^* - D_j = O(h), \tag{34}$$

where D_j is defined by (24).

A *consistent preprocessing of derivatives* is a set of pairs of lateral numerical derivatives $\{dl_j, dr_j\}$ defined as above associated to the grid data. A preprocessing is called *local* if the numerical derivatives are functions of only two variables.

A *preprocessed reconstruction procedure* is a third order method of reconstruction consisting of functions chosen from $\Re$ such that for every j, (17),

$$dl_j = r_j'(x_{j-\frac{1}{2}}) \tag{35}$$

and

$$dr_j = r_j'(x_{j+\frac{1}{2}}) \tag{36}$$

are satisfied. We need consistency properties (31)–(34) since they keep preprocessed solutions third order accurate.

Then we have the following theorem.

THEOREM 2.1. *The polynomial* ENO *third order method of reconstruction "via primitive function" of the function* $g(x)$ *is a nonlocal preprocessed parabolic reconstruction procedure that is* LTVB.

Proof. The following algorithm determines the lateral numerical derivatives in the ENO third order method on a cell C_j.

if $|d_{j-\frac{1}{2}}| \le |d_{j+\frac{1}{2}}|$ **then**
  $dl_j = d_{j-\frac{1}{2}}$
  if $|D_j| \le |D_{j-1}|$ **then**
    $dr_j = d_{j+\frac{1}{2}}$
  else
    $dr_j = dl_j + D_{j-1}$
else
  $dr_j = d_{j+\frac{1}{2}}$
  if $|D_j| \le |D_{j+1}|$ **then**
    $dl_j = d_{j-\frac{1}{2}}$
  else
    $dl_j = dr_j - D_{j+1}$

Thus, we determine the parabola (19) by substituting in (21) and (22), $d_{j-1/2}$ and d_{j+12} by dl_j and dr_j, respectively. We denote that parabola by p_j^*. Since the numerical derivatives are functions of four variables, then this preprocessing is not local. On the other hand, let us choose a number $h > 0$, such that there is at least two cells between two jumps of $g(x)$. Since $g(x)$ is a piecewise smooth function, it is easy to see that there exists a constant $M > 0$ depending only on derivatives of g in smooth regions, such that for all j, except for a finite number of "isolated" $j's$ (for which $d_{j+1/2} = O(h^{-1})$),

$$|d_{j+\frac{1}{2}}| \le M.$$

Thus, by the definition of the preprocessing, we have that for all j,

$$|dl_j| + |dr_j| \le 4 \cdot M. \tag{37}$$

Therefore, according to (29) applied to p_j^*, we obtain

$$LTV_j = TV(p_j^*) \le 4 \cdot M \cdot h \tag{38}$$

and the ENO reconstruction will be LTVB. □

Next, we look at local piecewise hyperbolic reconstructions. For the class $\Re_h$, we have no natural reconstruction procedure if local extrema are present in $g(x)$ because hyperbolas are monotonic. However, we can find a unique hyperbola r_j of the form (20) in every cell C_j, such that $d_{j-1/2} \cdot d_{j+1/2} > 0$ satisfying (17) and (18). We say a cell C_j is a *transition cell* if $d_{j-1/2} \cdot d_{j+1/2} \leq 0$. Since we look for a hyperbola at every cell including transition cells, we study more general conditions for the existence. For our purpose we need the following simple lemma.

LEMMA 2.2. *We consider the generic cell*

$$C_0 = \left\{ x : |x - x_0| \leq \frac{h}{2} \right\}.$$

Let θ be a real number between -1 and 1. We set $x(\theta) = x_0 + \theta \cdot \frac{h}{2}$. Let v_0 be the mean value of $g(x)$ in C_0. Let d be a nonzero real number. Then the following statements are true.

1. *The hyperbola*

$$r_0(x) = v_0 + d \cdot h \cdot \frac{1}{\alpha^2} \cdot \left(\log\left(\frac{2 - \alpha(1-\theta)}{2 + \alpha(1+\theta)}\right) - \frac{h}{(x - x_0) - \frac{h}{2} \cdot (\theta + \frac{2}{\alpha})} \right) \tag{39}$$

with derivative

$$r_0'(x) = d \cdot \frac{h^2}{\alpha^2 \cdot \left((x - x_0) - \frac{h}{2} \cdot (\theta + \frac{2}{\alpha})\right)^2} \tag{40}$$

is well defined according to the following restrictions on the range of values of the parameter α:

(a) *if $-1 < \theta < 1$ then $\frac{-2}{1+\theta} < \alpha < \frac{2}{1-\theta}$,*
(b) *if $\theta = 0$ then $-2 < \alpha < 2$,*
(c) *if $\theta = -1$ then $\alpha < 1$,*
(d) *if $\theta = 1$ then $\alpha > -1$.*

2. *The mean value of $r_0(x)$ in C_0 is v_0, i.e.,*

$$v_0 = \frac{1}{h} \int_{x_0 - h/2}^{x_0 + h/2} r_0(\xi)\, d\xi. \tag{41}$$

3. *The derivative of the hyperbola at $x(\theta)$ is d:*

$$r_0'(x(\theta)) = d \tag{42}$$

4. *If $d1$ is a nonzero real number, such that $d \cdot d1 > 0$, and θ_1 is a real number, such that $-1 < \theta_1 < 1$ and $\theta_1 \neq \theta$, then we can determine the value of α by the formula*

$$\alpha = \left(\frac{2}{\theta_1 - \theta}\right) \cdot \left(1 - \sqrt{\frac{d}{d1}}\right), \tag{43}$$

such that

$$r_0'(x(\theta_1)) = d1. \tag{44}$$

Thus the hyperbola is completely determined from v_0, d, and $d1$, when $d \cdot d1 > 0$, supposing that parameters θ and θ_1 are given. Moreover, the derivative of $r_0(x)$ at the midpoint $x(\frac{\theta+\theta_1}{2})$ is

$$r_0'\left(x\left(\frac{\theta + \theta_1}{2}\right)\right) = \operatorname{sgn}(d) \cdot \left(\frac{2 \cdot \sqrt{|d|} \cdot \sqrt{|d1|}}{\sqrt{|d|} + \sqrt{|d1|}}\right)^2. \tag{45}$$

5. *We set* $dl_0 = r_0'(x_0 - \frac{h}{2})$ *and* $dr_0 = r_0'(x_0 + \frac{h}{2})$. *Then the total variation of the hyperbola on* C_0 *is*

$$TV(r_0) = h \cdot \sqrt{dl_0 \cdot dr_0}. \tag{46}$$

6. *If* $\theta = 0$ *then the lateral values of* r_0 *are well defined for* $-2 < \alpha < 2$ *by the formulas*

$$r_0\left(x_0 + \frac{h}{2}\right) = v_0 + d \cdot h \cdot \eta(\alpha), \tag{47}$$

$$r_0\left(x_0 - \frac{h}{2}\right) = v_0 - d \cdot h \cdot \eta(-\alpha), \tag{48}$$

where the function η *is defined as*

$$\eta(\alpha) = \frac{1}{\alpha^2} \cdot \left(\log\left(\frac{2-\alpha}{2+\alpha}\right) + \frac{2 \cdot \alpha}{2-\alpha} \right) \tag{49}$$

(*the function* η *is positive in* $-2 < \alpha < 2$ *and has a removable discontinuity at* $\alpha = 0$ *by defining* $\eta(0) = \frac{1}{2}$).

7. *Let* D_0^* *be a nonzero preprocessed difference satisfying* (34). *If* α *defined by*

$$\alpha = \frac{D_0^*}{2 \cdot d}$$

satisfies the restrictions in (1), *the hyperbola* r_0 *satisfies* (42) *and* $r_0''(x(\theta)) = \frac{D_0^*}{h}$.

Proof. The proof is straightforward. □

First, we study the "most natural" local hyperbolic reconstruction (LHR) procedure. Since our reconstruction is local, we restrict our discussion to the cell C_0 and the grid data of the cell: v_0, $d_{-1/2}$, and $d_{1/2}$. To fit the hyperbola, we establish formulas to obtain d and α with $\theta = 0$. The LHR algorithm computes for the generic cell C_0 the derivative at the point x_0 and the adimensional parameter α from $d_{-1/2}$ and $d_{1/2}$. If C_0 is a nontransition cell, then the value assigned to d is the average (45) and α is computed by means of (43); therefore, the hyperbola obtained has as derivatives at the endpoints of the cell $d_{-1/2}$ and $d_{1/2}$. If C_0 is a transition cell, we change the derivative with largest absolute value by the other one multiplied by h^2, thus, the reconstruction on this cell may degenerate to second order.

Let tol be a constant such that $\text{tol} = O(h^2)$, e.g., define $\text{tol} = h^2$.

LOCAL HYPERBOLIC RECONSTRUCTION (LHR)

if $(|d_{-\frac{1}{2}}| \le \text{tol})$ **and** $(|d_{\frac{1}{2}}| \le \text{tol})$ **then**

$d = 0$ **and** $\alpha = 0$

else

if $(|d_{-\frac{1}{2}}| \le \text{tol})$ **or** (C_0 is a transition cell with $|d_{\frac{1}{2}}| \le |d_{-\frac{1}{2}}|$) **then**

$d = 4 \cdot d_{\frac{1}{2}} \cdot \left(\frac{h}{1+h}\right)^2$

$\alpha = 2\frac{1-h}{1+h}$

else

if $(|d_{\frac{1}{2}}| \le \text{tol})$ **or** (C_0 is a transition cell with $|d_{-\frac{1}{2}}| \le |d_{\frac{1}{2}}|$) **then**

$$d = 4 \cdot d_{-\frac{1}{2}} \cdot \left(\frac{h}{1+h}\right)^2$$

$$\alpha = -2\frac{1-h}{1+h}$$

else

$$d = \operatorname{sgn}(d_{-\frac{1}{2}}) \cdot \frac{4 \cdot |d_{-\frac{1}{2}}| \cdot |d_{\frac{1}{2}}|}{|d_{-\frac{1}{2}}| + |d_{\frac{1}{2}}| + 2 \cdot \sqrt{d_{-\frac{1}{2}} \cdot d_{\frac{1}{2}}}}$$

if $|d_{-\frac{1}{2}}| \leq |d_{\frac{1}{2}}|$ **then**

$$\alpha = 2 \cdot \left(\sqrt{\frac{d}{d_{-\frac{1}{2}}}} - 1\right)$$

else

$$\alpha = 2 \cdot \left(1 - \sqrt{\frac{d}{d_{\frac{1}{2}}}}\right)$$

In Table 2, we present the numerical order at each point for the lateral values of the hyperbola for the LHR method applied to the function $\frac{1}{2} \cdot (\frac{1}{2} + \sin(2\pi x))$ for $0 \leq x \leq 1$ using 40 grid points. (In fact, we only show the numerical orders for the grid points between 0 and 0.5 because we have the same values for the other half interval.) We computed numerical orders by applying extrapolation to the errors for 40 and 80 grid points. We have second order accuracy on transition cells near local extrema with abscissa $x = .25$. In Table 2 we also included the pointwise errors of the right-side approximations.

TABLE 2
Numerical orders and pointwise errors of the LHR *method.*

Abscissa	40(right err.)	80(right err.)	r. ord.	l. ord.
.025	$.85 \cdot 10^{-4}$	$.10 \cdot 10^{-4}$	3.02	2.99
.050	$.94 \cdot 10^{-4}$	$.11 \cdot 10^{-4}$	3.04	2.97
.075	$.11 \cdot 10^{-3}$	$.13 \cdot 10^{-4}$	3.06	2.95
.100	$.13 \cdot 10^{-3}$	$.15 \cdot 10^{-4}$	3.08	2.94
.125	$.16 \cdot 10^{-3}$	$.19 \cdot 10^{-4}$	3.10	2.92
.150	$.21 \cdot 10^{-3}$	$.25 \cdot 10^{-4}$	3.12	2.90
.175	$.31 \cdot 10^{-3}$	$.34 \cdot 10^{-4}$	3.17	2.88
.200	$.55 \cdot 10^{-3}$	$.56 \cdot 10^{-4}$	3.30	2.84
.225	$.38 \cdot 10^{-2}$	$.14 \cdot 10^{-3}$	4.79	2.73
.250	$-.20 \cdot 10^{-2}$	$-.51 \cdot 10^{-3}$	1.99	1.99
.275	$-.50 \cdot 10^{-3}$	$-.75 \cdot 10^{-4}$	2.73	4.79
.300	$-.30 \cdot 10^{-3}$	$-.42 \cdot 10^{-4}$	2.84	3.30
.325	$-.21 \cdot 10^{-3}$	$-.28 \cdot 10^{-4}$	2.88	3.17
.350	$-.16 \cdot 10^{-3}$	$-.21 \cdot 10^{-4}$	2.90	3.12
.375	$-.13 \cdot 10^{-3}$	$-.17 \cdot 10^{-4}$	2.92	3.10
.400	$-.11 \cdot 10^{-3}$	$-.14 \cdot 10^{-4}$	2.94	3.08
.425	$-.94 \cdot 10^{-4}$	$-.12 \cdot 10^{-4}$	2.95	3.06
.450	$-.85 \cdot 10^{-4}$	$-.11 \cdot 10^{-4}$	2.97	3.04
.475	$-.81 \cdot 10^{-4}$	$-.10 \cdot 10^{-4}$	2.99	3.02
.500	$-.81 \cdot 10^{-4}$	$-.10 \cdot 10^{-4}$	3.01	3.01

In spite of the smoothing made on transition cells in LHR, the method of reconstruction is not LTVB, because for nontransition cells, C_j, such that $d_{j+1/2} = O(h^{-1})$, we have that $TV(r_j) = O(h^{1/2})$ according to (46). However, this method of reconstruction works satisfactorily for contact discontinuities when it is used with the Shu–Osher TVD Runge–Kutta time evolution of §3. For nonlinear fluxes, the method becomes "unstable." According to our numerical experiments, the LHR method shows that for nonlinear fluxes, more smoothing

appears to be necessary and that the LTVB condition provides enough "smoothing" to prevent excessive increasing of the total variation of the solution.

Next, we introduce a piecewise hyperbolic reconstruction with a *local smoothing* such that the method of reconstruction becomes LTVB. We use the same structure as the LHR method, but we assign a different value to the central derivative d (this is the preprocessing we do). If C_0 is a nontransition cell, then we define d by

$$d = \frac{2 \cdot d_{-\frac{1}{2}} \cdot d_{\frac{1}{2}}}{d_{-\frac{1}{2}} + d_{\frac{1}{2}}}, \tag{50}$$

which is the harmonic mean of $d_{-1/2}$ and $d_{1/2}$. Then the algorithm defines the hyperbola such that its derivative interpolates d at x_0 and the lateral grid derivative with smallest absolute value. Since the harmonic mean is smaller in absolute value than (45), it is easy to see that the hyperbola has a lateral derivative at the nonsmooth side smaller in absolute value than the original one. Taylor series expansion arguments show that the preprocessing defined in this way is consistent, since the harmonic mean provides an $O(h^2)$ approximation of the derivative at x_0. Thus, we have the complete algorithm.

LOCAL HYPERBOLIC HARMONIC RECONSTRUCTION (LHHR)
* tol $= O(h^2)$

if $(|d_{-\frac{1}{2}}| \le \text{tol})$ **and** $(|d_{\frac{1}{2}}| \le \text{tol})$ **then**

$d = 0$ **and** $\alpha = 0$

else

if $(|d_{-\frac{1}{2}}| \le \text{tol})$ **or** (C_0 is a transition cell with $|d_{\frac{1}{2}}| \le |d_{-\frac{1}{2}}|$) **then**

$$d = 2 \cdot d_{\frac{1}{2}} \cdot \left(\frac{h^2}{1+h^2}\right)$$

$$\alpha = 2 \cdot \left(\sqrt{\frac{2}{1+h^2}} - 1\right)$$

else

if $(|d_{\frac{1}{2}}| \le \text{tol})$ **or** (C_0 is a transition cell with $|d_{-\frac{1}{2}}| \le |d_{\frac{1}{2}}|$) **then**

$$d = 2 \cdot d_{-\frac{1}{2}} \cdot \left(\frac{h^2}{1+h^2}\right)$$

$$\alpha = -2 \cdot \left(\sqrt{\frac{2}{1+h^2}} - 1\right)$$

else

$$d = \frac{2 \cdot d_{-\frac{1}{2}} \cdot d_{\frac{1}{2}}}{d_{-\frac{1}{2}} + d_{\frac{1}{2}}}$$

if $|d_{-\frac{1}{2}}| \le |d_{\frac{1}{2}}|$ **then**

$$\alpha = 2 \cdot \left(\sqrt{\frac{d}{d_{-\frac{1}{2}}}} - 1\right)$$

else

$$\alpha = 2 \cdot \left(1 - \sqrt{\frac{d}{d_{\frac{1}{2}}}}\right)$$

Transition cells are treated analogously to the LHR method, but use the harmonic mean instead of (45). In Table 3, we show the pointwise numerical orders and pointwise right error

for the LHHR method computed for the function used in Table 2 in the same way. We also show the half interval because the obtained values are the same.

TABLE 3
Numerical orders and pointwise errors of the LHHR *Method.*

Abscissa	40(right err.)	80(right err.)	r. ord.	l. ord.
.025	$.90 \cdot 10^{-4}$	$.11 \cdot 10^{-4}$	3.05	2.98
.050	$.11 \cdot 10^{-3}$	$.13 \cdot 10^{-4}$	3.07	2.96
.075	$.13 \cdot 10^{-3}$	$.16 \cdot 10^{-4}$	3.10	2.94
.100	$.18 \cdot 10^{-3}$	$.20 \cdot 10^{-4}$	3.11	2.92
.125	$.24 \cdot 10^{-3}$	$.27 \cdot 10^{-4}$	3.13	2.91
.150	$.34 \cdot 10^{-3}$	$.38 \cdot 10^{-4}$	3.16	2.89
.175	$.52 \cdot 10^{-3}$	$.56 \cdot 10^{-4}$	3.22	2.87
.200	$.95 \cdot 10^{-3}$	$.94 \cdot 10^{-4}$	3.35	2.83
.225	$.41 \cdot 10^{-2}$	$.24 \cdot 10^{-3}$	4.07	2.72
.250	$-.20 \cdot 10^{-2}$	$-.51 \cdot 10^{-3}$	2.00	2.00
.275	$-.66 \cdot 10^{-3}$	$-.10 \cdot 10^{-3}$	2.72	4.07
.300	$-.39 \cdot 10^{-3}$	$-.54 \cdot 10^{-4}$	2.83	3.35
.325	$-.26 \cdot 10^{-3}$	$-.36 \cdot 10^{-4}$	2.87	3.22
.350	$-.19 \cdot 10^{-3}$	$-.26 \cdot 10^{-4}$	2.89	3.16
.375	$-.15 \cdot 10^{-3}$	$-.20 \cdot 10^{-4}$	2.91	3.13
.400	$-.12 \cdot 10^{-3}$	$-.16 \cdot 10^{-4}$	2.92	3.11
.425	$-.10 \cdot 10^{-3}$	$-.13 \cdot 10^{-4}$	2.94	3.10
.450	$-.87 \cdot 10^{-4}$	$-.11 \cdot 10^{-4}$	2.96	3.07
.475	$-.81 \cdot 10^{-4}$	$-.10 \cdot 10^{-4}$	2.98	3.05
.500	$-.82 \cdot 10^{-4}$	$-.10 \cdot 10^{-4}$	3.01	3.01

Now, we show that the LHHR method is LTVB. We need the following lemma.

LEMMA 2.3. *The range of values of the adimensional parameter* α *for the* LHHR *method is*

$$-2(\sqrt{2}-1) \le \alpha \le 2(\sqrt{2}-1). \tag{51}$$

Proof. For transition cells the proof is trivial. If C_0 is a nontransition cell, then we will suppose that $|d_{-\frac{1}{2}}| \le |d_{\frac{1}{2}}|$ (the other case is symmetric). Then the algorithm defines

$$\alpha = 2 \cdot \left(\sqrt{\frac{d}{d_{-\frac{1}{2}}}} - 1\right).$$

Thus it is enough to prove that $\frac{d}{d_{-1/2}} \le 2$ and this follows from $d_{-1/2} \cdot d_{1/2} > 0$. □

THEOREM 2.4. *The* LHHR *method of reconstruction of the function* $g(x)$ *is a local preprocessed hyperbolic reconstruction procedure that is* LTVB.

Proof. Following the same argument used in Theorem 2.1, we can find a constant $M > 0$ such that for all j except for a finite number of "isolated" j's (for which $d_{j+1/2} = O(h^{-1})$), $|d_{j+1/2}| \le M$. From (40) if C_j is a nontransition cell and $|d_{j-1/2}| \le |d_{j+1/2}|$, then the preprocessed derivatives at the endpoint of the cells are the following:

$$dl_j = d_{j-\frac{1}{2}}, \tag{52}$$

$$dr_j = d_{j-\frac{1}{2}} \cdot \left(\frac{2}{2-\alpha}\right)^2. \tag{53}$$

From (46) and Lemma 2.3, it follows that

$$LTV_j = TV(r_j) \le 2 \cdot M \cdot h. \tag{54}$$

The argument is similar for transition cells. □

In theory, it is possible to choose other means (averages) between the harmonic mean and (45) by giving methods of reconstruction that are LTVB, but we have not found any others that are as computationally convenient as LHHR.

3. Piecewise hyperbolic methods (PHMs). In this section, we describe the algorithm that is based on the first order Roe scheme [8], with the entropy-fix correction due to Shu and Osher [10], for local piecewise hyperbolic reconstructions. This gives us what we call PHMs. Since the evolution in time is performed by means of the third order TVD Runge–Kutta method (10) with positive coefficients (see Table 1), we have that the numerical solution at every timestep is a convex combination of Euler forward time substeps (3). Thus, we restrict our description of the algorithm to the computation of numerical fluxes $\hat{f}_{j+1/2}$ in (3).

Roughly speaking, the reconstruction procedure is integrated in (3), taking into account the dynamics of the differential equation (1). Indeed, the numerical fluxes are reconstructed from the *upwind side* (i.e., according to the direction of the *wind*), except that if the *wind* changes direction at the cell (i.e., there is a "sonic point" at the cell), then a local Lax–Friedrichs flux decomposition is performed. Thus, we have two alternative phases: upwindness or flux decomposition. Flux decomposition is only used in cells containing "sonic points," and since these points are isolated, the cost of the algorithm depends on the "upwindness phase." The "upwind side" is determined according to the *local* sign of $f'(u)$ at $x_{j+1/2}$. In our case we use the "Roe" speed

$$\overline{a}_{j+\frac{1}{2}} = \frac{f(u^n_{j+1}) - f(u^n_j)}{u^n_{j+1} - u^n_j} \tag{55}$$

to determine the sign of $f'(u_{j+1/2})$. For a detailed explanation on the local Lax–Friedrichs flux decomposition, we refer to [10]. We then have the following algorithm with the LHHR method.

ALGORITHM PHM-REF

Step 1: *Computation of Grid Data*

From u^n_j we compute the grid data by means of:

$$v_j = f(u^n_j), \tag{56}$$

$$d_{j+\frac{1}{2}} = \frac{v_{j+1} - v_j}{h}, \tag{57}$$

for all j.

Step 2: *Local Preprocessing of Derivatives*

Computation of $d(j)$ and α_j using LHHR for all j.

Step 3: **for every** j **do**

begin

if $f'(u)$ does not change sign between u^n_j and u^n_{j+1} **then**

Upwindness Phase

(UP1) $\overline{a}_{j+\frac{1}{2}} = \frac{v_{j+1}-v_j}{u^n_{j+1}-u^n_j}$ (Roe speed)

if $\overline{a}_{j+\frac{1}{2}} \geq 0$ **then**

(UP21) $\hat{f}_{j+\frac{1}{2}} = v_j + d(j) \cdot h \cdot \eta(\alpha_j)$ (using (47))

else

(UP22) $\hat{f}_{j+\frac{1}{2}} = v_{j+1} - d(j+1) \cdot h \cdot \eta(-\alpha_{j+1})$ (using (48))

else

Flux Decomposition Phase

(FD1) $M_{j+\frac{1}{2}} = \max_{u_j^n < u < u_{j+1}^n} |f'(u)|$

(FD2) $v_k^+ = \frac{1}{2} \cdot (v_k + M_{j+\frac{1}{2}} \cdot u_k^n), \qquad k = j-1, j, j+1$

(FD3) $d_{k-\frac{1}{2}}^+ = \frac{v_k^+ - v_{k-1}^+}{h}, \qquad k = j, j+1$

(FD4) Computation of $d^+(j)$ and α_j^+ using LHHR from d^+'s.

(FD5) $f^+ = v_j^+ + d^+(j) \cdot h \cdot \eta(\alpha_j^+)$

(FD6) $v_k^- = \frac{1}{2} \cdot (v_k - M_{j+\frac{1}{2}} \cdot u_k^n), \qquad k = j, j+1, j+2$

(FD7) $d_{k+\frac{1}{2}}^- = \frac{v_{k+1}^- - v_k^-}{h}, \qquad k = j, j+1$

(FD8) Computation of $d^-(j)$ and α_j^- using LHHR from d^-'s.

(FD9) $f^- = v_{j+1}^- - d^-(j+1) \cdot h \cdot \eta(-\alpha_{j+1}^-)$

(FD10) $\hat{f}_{j+\frac{1}{2}} = f^+ + f^-$

end

If we use the LHR method instead of LHHR, we obtain another PHM method that we refer to as CPHM-REF.

Some remarks are in order here.

Remark 1. From the algorithm it follows that numerical fluxes are functions of four variables. This is also true if we use any local reconstruction method instead of LHHR. However, the third order ENO reconstruction in this case (due to Shu and Osher [10]) gives numerical fluxes that are functions of six variables. That point is crucial to distinguish the behavior between local and nonlocal upwind third order methods. As we will see in numerical experiments, the spreading of "noise" coming from the singularities of the solution is clearly reduced in local methods. Since "local smoothing" does not use second differences of the data, then the "corners" of the solution (jumps in first derivative) are well preserved during the evolution, as numerical experiments show (compare Figs. 1(a) and 1(b)).

Remark 2. The "pseudocode" that describes Algorithm PHM-REF was made for better comprehension and it is not optimal in the sense of computational cost. In spite of obvious improvements, the cost of the algorithm is a bit higher than third order ENO. The cost depends on the computation of the first differences and the evaluation of the parameters d, α, and the function η. To evaluate η at values close to zero, it may be computationally convenient to use a Padé approximant near zero because of the removable discontinuity at zero. On the other hand, local methods constructed in this way are stable and accurate for high CFL constants and also reduce computational cost.

Remark 3. To sharpen contact discontinuities, we have used the Yang artificial compression method (see [11] that is applicable to the PHM-REF Algorithm in the version given by Shu and Osher in [10, p. 42, Algorithm 3.2, formula (3.2)]. We have found that this method works efficiently for "small" CFLs. The interested reader can find details about that method in [11].

Remark 4. Scalar multidimensional initial value problems of the form

$$u_t + \sum_{i=1}^{d} f_i(u)_{x_i} = 0, \tag{58}$$

$$u(x,0) = u_0(x) \tag{59}$$

are approximated by applying the 1D procedure to each of the terms $f_i(u)_{x_i}$ in (58) keeping all other variables fixed. Then the Runge–Kutta method (10) is used with CFL numbers shrunk by a factor d^{-1}. Indeed, a classical CFL restriction

$$\frac{\Delta t}{h} \cdot \max_u |f'(u)| \le \lambda_0$$

is replaced by

$$\Delta t \cdot \max_u \sum_{i=1}^{d} \frac{1}{h_i} |f_i'(u)| \le \lambda_0.$$

4. Numerical experiments. We combine the algorithms PHM-REF, CPHM-REF, and ENO3-REF (described in §2), with the LHHR, the LHR method, and the Shu–Osher third order ENO reconstruction methods (described in §3), respectively. If we add the label AC, that means that the Yang artificial compression method in the version of Shu and Osher (see [10]) has been applied. We have run most examples for different CFLs and time levels, but here we only include what we consider as representatives. In Figs. 1–6, circles are approximate numerical solutions and solid lines are piecewise-linear functions that interpolate exact solutions computed either at grid points or at points of a finer grid. The 3D figures represent level curves of the numerical solution.

Example 1. To study the accuracy of our methods, we consider the scalar linear equation

$$u_t + u_x = 0 \qquad 0 \le x < 1 \tag{60}$$

with the 1-periodic smooth initial data

$$u(x,0) = \frac{1}{2}\left(\frac{1}{2} + \sin(2\pi x)\right). \tag{61}$$

We have solved (60), (61) at $t = 1$ with CFL $= 0.8$ and the refinement sequence $N = 20, 40, 80, 160$ grid points for the PHM-REF and CPHM-REF methods, respectively, and in Table 4 we show the corresponding L_∞-errors and L_1-errors that compare with the exact solution. We observe that both methods are $O(h^{3/2})$ accurate in the L_∞-norm because of the loss of accuracy at local extrema. In Table 5, pointwise errors and numerical orders r_{20} and r_{40} are shown for the method CPHM-REF computed by Richardson extrapolation, on 20 and 40 grid points taken as starting grids, respectively. We list only the values corresponding to the first half interval because the errors and numerical orders are symmetric. Figures for the PHM method are similar. Thus, the behavior is clearly better than TVD methods.

Example 2. We consider the following periodic initial value problem:

$$u_t + u_x = 0 \qquad -1 \le x < 1, \tag{62}$$

$$u(x,0) = \begin{cases} \sin\left(\pi \cdot \frac{x+0.3}{0.6}\right) & \text{if } -0.3 \le x \le 0.3, \\ 0 & \text{otherwise.} \end{cases} \tag{63}$$

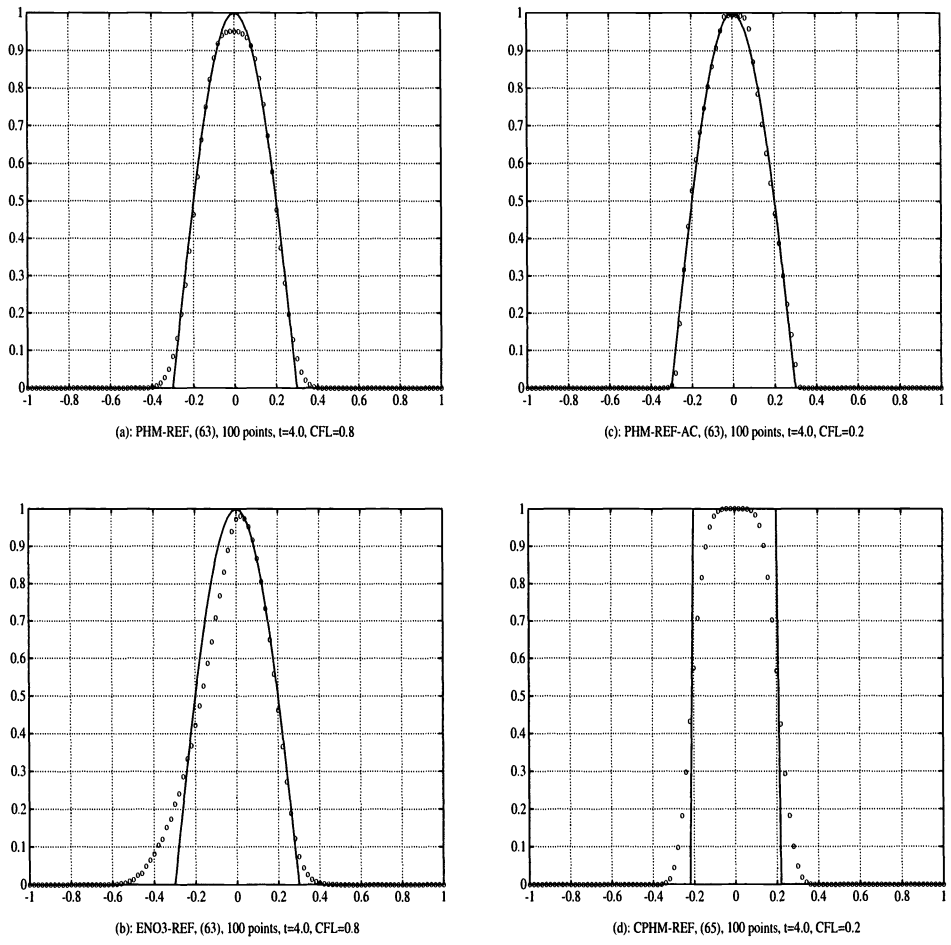


FIG. 1.

TABLE 4
Numerical solution of $u_t + u_x = 0$, $0 \le x < 1$ $u(x,0) = \frac{1}{2}\left(\frac{1}{2} + \sin(2\pi x)\right)$ *at* $t = 1$ *and* CFL $= 0.8$.

	L_∞-error		L_1-error	
N	PHM-REF	CPHM-REF	PHM-REF	CPHM-REF
20	$6.45 \cdot 10^{-2}$	$5.14 \cdot 10^{-2}$	$2.53 \cdot 10^{-2}$	$1.80 \cdot 10^{-2}$
40	$2.36 \cdot 10^{-2}$	$1.79 \cdot 10^{-2}$	$6.90 \cdot 10^{-3}$	$4.50 \cdot 10^{-3}$
80	$8.60 \cdot 10^{-3}$	$6.30 \cdot 10^{-3}$	$1.70 \cdot 10^{-3}$	$1.10 \cdot 10^{-3}$
160	$3.10 \cdot 10^{-3}$	$2.20 \cdot 10^{-3}$	$5.32 \cdot 10^{-4}$	$3.39 \cdot 10^{-4}$

We solve (62), (63) at t=2 and CFL $= 0.8$. In Table 6 we list the L_∞-error and L_1-error for a refinement sequence with $N = 20, 40, 80, 160$ for PHM-REF and CPHM-REF methods, comparing with the exact solution. We observe that both methods retain first order accuracy in the L_∞-norm in spite of the presence of two jumps in the derivative of the solution, which shows the good resolution of corners in both methods. In Table 7 we show pointwise errors and numerical orders r_{20} and r_{40} computed at 20 points following the same method as in Table 5.

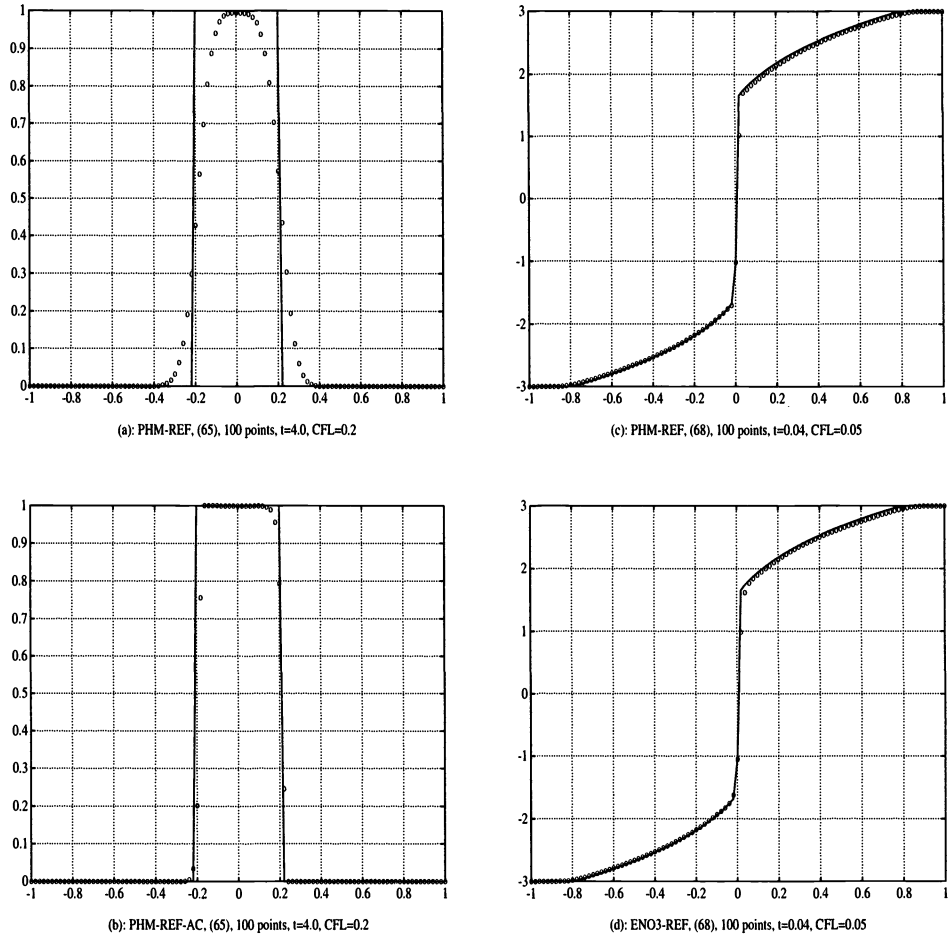
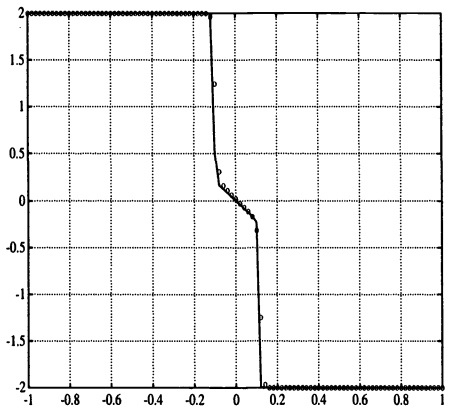


FIG. 2.

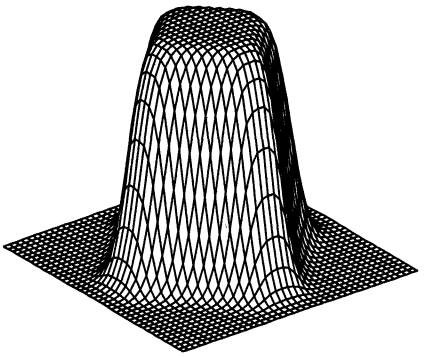
TABLE 5

Numerical orders r_{20} and r_{40} for the CPHM-REF *method with the periodic smooth data* (61) *at* $t = 1$ *and* $CFL = 0.8$.

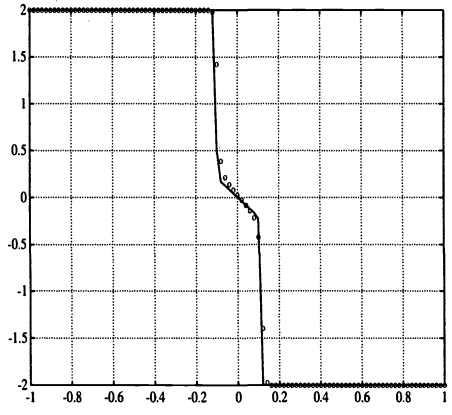
Abscissa	20	40	80	160	r_{20}	r_{40}
.00	$.66 \cdot 10^{-2}$	$-.10 \cdot 10^{-3}$	$.67 \cdot 10^{-5}$	$-.92 \cdot 10^{-6}$	5.96	3.80
.05	$-.65 \cdot 10^{-2}$	$.46 \cdot 10^{-3}$	$-.25 \cdot 10^{-4}$	$-.36 \cdot 10^{-5}$	3.85	4.49
.10	$-.11 \cdot 10^{-1}$	$-.20 \cdot 10^{-2}$	$.59 \cdot 10^{-4}$	$-.74 \cdot 10^{-5}$	2.21	4.94
.15	$.20 \cdot 10^{-2}$	$-.55 \cdot 10^{-2}$	$-.39 \cdot 10^{-3}$	$-.22 \cdot 10^{-4}$	.56	3.76
.20	$.32 \cdot 10^{-1}$	$.33 \cdot 10^{-2}$	$-.18 \cdot 10^{-2}$	$-.48 \cdot 10^{-3}$	2.50	1.98
.25	$.52 \cdot 10^{-1}$	$.18 \cdot 10^{-1}$	$.63 \cdot 10^{-2}$	$.22 \cdot 10^{-2}$	1.54	1.52
.30	$.32 \cdot 10^{-1}$	$.50 \cdot 10^{-2}$	$-.18 \cdot 10^{-2}$	$-.67 \cdot 10^{-3}$	1.99	2.60
.35	$.85 \cdot 10^{-2}$	$-.61 \cdot 10^{-2}$	$-.96 \cdot 10^{-3}$	$-.14 \cdot 10^{-6}$	1.50	2.43
.40	$-.15 \cdot 10^{-1}$	$-.41 \cdot 10^{-2}$	$.10 \cdot 10^{-3}$	$-.96 \cdot 10^{-5}$	1.38	5.21
.45	$-.15 \cdot 10^{-1}$	$-.38 \cdot 10^{-3}$	$-.33 \cdot 10^{-4}$	$-.44 \cdot 10^{-5}$	5.40	3.61
.50	$-.66 \cdot 10^{-2}$	$.99 \cdot 10^{-4}$	$-.82 \cdot 10^{-5}$	$-.10 \cdot 10^{-5}$	5.96	3.90



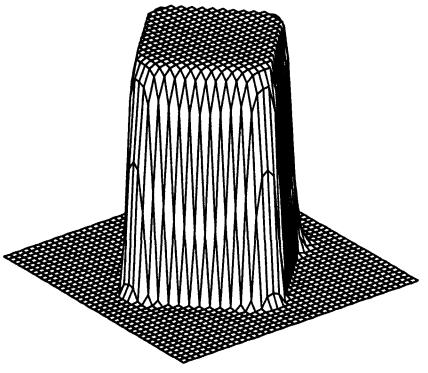
(a): PHM-REF, (69), 100 points, t=0.2, CFL=0.05



(c): PHM-REF, (71), 40x40 grid points, t=2, CFL=.4



(b): ENO3-REF, (69), 100 points, t=0.2, CFL=0.05



(d): PHM-REF-AC, (71), 40x40 grid points, t=2, CFL=.1

FIG. 3.

Numerical orders marked with an asterisk (∗) correspond to transition points of jumps of the first derivative of the solution and are meaningless. We have tested our schemes with longer time levels, observing that both methods are not sensitive to the CFL number. We illustrate in Fig. 1(a) the behavior of PHM-REF for CFL = 0.8 at t=4. It compares with that of the Shu–Osher third order ENO method shown in Fig. 1(b), showing the value of the localness for upwind schemes. Figure 1(c) shows the PHM-REF method with the Yang artificial compression, (see [10] and [11]) where a lower CFL number is used for such a filter to be efficient.

Example 3. We consider the following periodic initial value problem:

$$u_t + u_x = 0 \qquad -1 \le x < 1, \tag{64}$$

$$u(x,0) = \begin{cases} 1 & \text{if } -0.2 \le x \le 0.2, \\ 0 & \text{otherwise.} \end{cases} \tag{65}$$

We solve (64), (65) at t=4. Figures 1(d) and 2(a) show the solution with CFL = 0.2 for the CPHM-REF and PHM-REF, respectively, and we can observe that discontinuities smear less

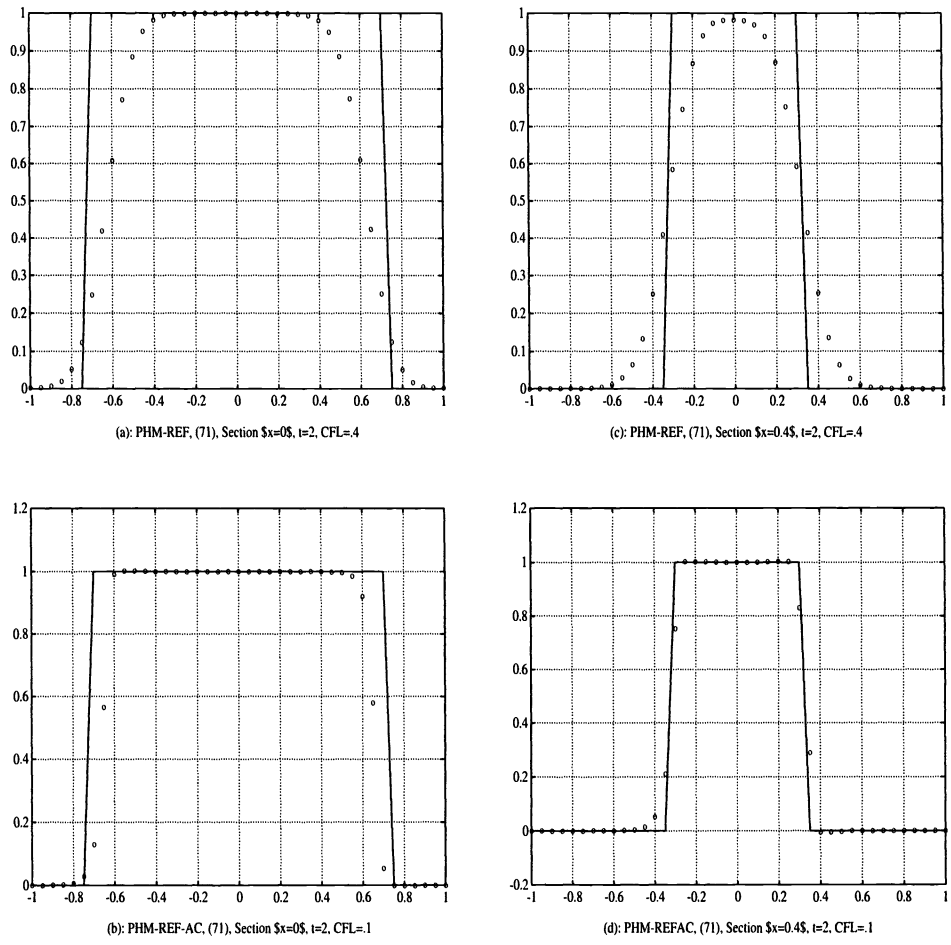


FIG. 4.

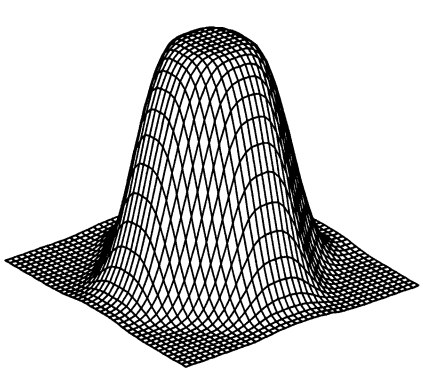
TABLE 6

Numerical solution of $u_t + u_x = 0$, $-1 \le x < 1$ $u(x,0) = \sin\left(\pi \frac{x+0.3}{0.6}\right)$ *for* $-0.3 \le x \le 0.3$ *and* $u(x,0) = 0$ *otherwise, at* $t = 2$ *and* CFL $= 0.8$.

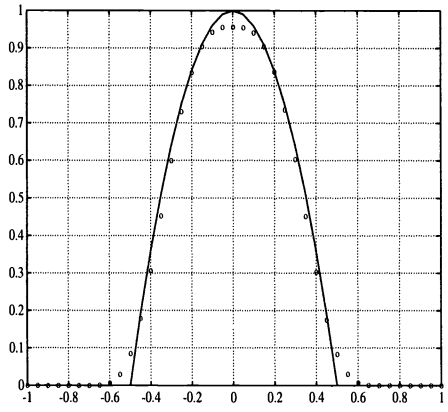
	L_∞-error		L_1-error	
N	PHM-REF	CPHM-REF	PHM-REF	CPHM-REF
20	$4.03 \cdot 10^{-1}$	$3.68 \cdot 10^{-1}$	$2.01 \cdot 10^{-1}$	$1.82 \cdot 10^{-1}$
40	$1.62 \cdot 10^{-1}$	$1.51 \cdot 10^{-1}$	$6.84 \cdot 10^{-2}$	$5.92 \cdot 10^{-2}$
80	$8.40 \cdot 10^{-2}$	$8.06 \cdot 10^{-2}$	$2.96 \cdot 10^{-2}$	$2.69 \cdot 10^{-2}$
160	$4.96 \cdot 10^{-2}$	$4.72 \cdot 10^{-2}$	$1.34 \cdot 10^{-2}$	$1.19 \cdot 10^{-2}$

for CPHM-REF than for PHM-REF. The artificial compression method is used for PHM-REF (with CFL $= 0.2$) to solve the same problem. The effect of such a filter is shown in Fig. 2(b).

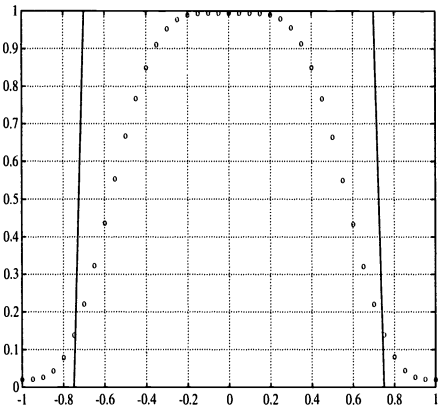
Example 4. PHM-REF solves nonlinear scalar conservation laws with a convex flux and different initial data and, therefore, we omit any comment or picture about this case. To test the PHM-REF scheme with more difficult problems, we consider two Riemann problems for



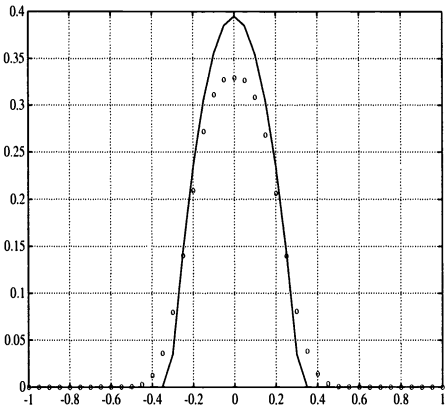
(a): CPHM-REF, (71), 40x40 grid points, t=16, CFL=.4



(c): CPHM-REF, (72), Section $x=0$, t=2, CFL=.2



(b): CPHM-REF, (71), Section $x=0$, t=16, CFL=.4



(d): CPHM-REF, (72), Section $x=-0.55$, t=2, CFL=.2

FIG. 5.

the following nonconvex flux example.

$$u_t + (f(u))_x = 0, \tag{66}$$

where

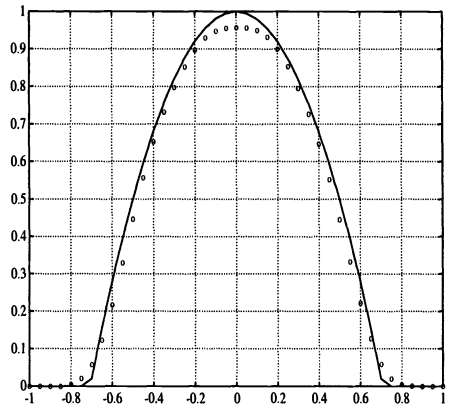
$$f(u) = \frac{1}{4}(u^2 - 1)(u^2 - 4) \tag{67}$$

with the following initial data:

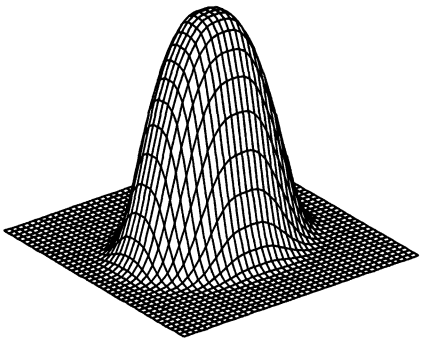
$$u(x,0) = -3 \quad \text{if } x \le 0 \quad \text{and} \quad u = 3 \quad \text{if } x > 0, \tag{68}$$

$$u(x,0) = 2 \quad \text{if } x \le 0 \quad \text{and} \quad u = -2 \text{if } x > 0. \tag{69}$$

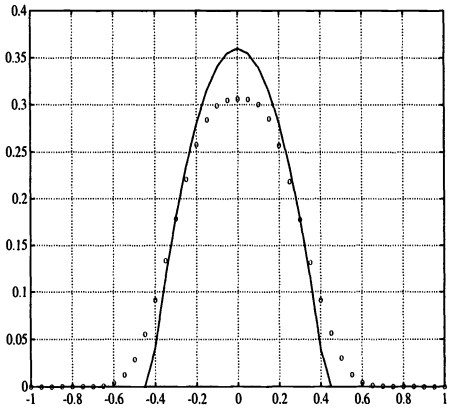
For the Riemann problem (68), instead of the expected sonic rarefaction fan that appears for convex fluxes, a nonconvex sonic stationary shock at $x = 0$ is developed. Secondly, a



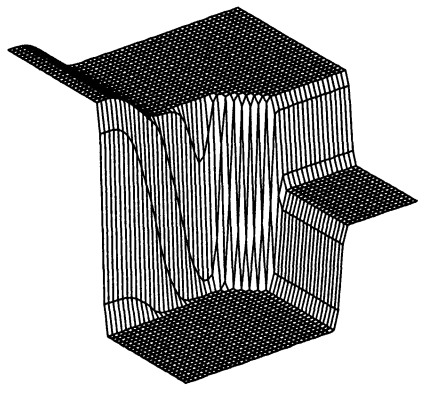
(a): CPHM-REF, (72), Section $y=0$, t=2, CFL=.2



(c): CPHM-REF, (72), 40x40 grid points, t=2, CFL=.2



(b): CPHM-REF, (72), Section $y=-0.4$, t=2, CFL=.2



(d): PHM-REF, (73), (-1,-0.2,0.5,0.8), 50x50 points, t=1

FIG. 6.

nonconvex sonic centered rarefaction fan breaks the initial discontinuity in two for the problem (69) when it evolves in time. A complete discussion about the exact solution to these problems can be found in [6]. Our scheme solves well both problems. In Figs. 2(c) and 2(d) we compare the numerical resolution of (68) with PHM-REF and ENO3-REF, respectively, at $t = 0.04$ with CFL $= 0.05$, and we observe that the shock (mainly the right side) is less viscous for our method than for ENO3-REF. The Riemann problem (69) is also solved for both methods at $t = 0.2$. The numerical solutions are represented in Figs. 3(a) and 3(b), observing that PHM-REF better solves the nonconvex sonic rarefaction.

Example 5. We consider the 2D linear equation

$$u_t + u_x + u_y = 0 \qquad -1 \le x, y \le 1, \tag{70}$$

where u is 2-periodic in x and y. To study the behavior of the PHM-REF scheme with contact discontinuities and corners in 2D, we test the following two initial data:

$$u(x, y, 0) = \begin{cases} 1 & \text{if } (x, y) \in S, \\ 0 & \text{otherwise,} \end{cases} \tag{71}$$

TABLE 7
Numerical orders r_{20} and r_{40} for the PHM-REF *method with the periodic data* (63) *at* $t = 2$ *and* CFL $= 0.8$

abscisa	20	40	80	160	r_{20}	r_{40}
−1.0	$-.24 \cdot 10^{-4}$	$.60 \cdot 10^{-5}$	$.66 \cdot 10^{-7}$	$-.43 \cdot 10^{-9}$	2.35	6.49
−.90	$.41 \cdot 10^{-3}$	$.58 \cdot 10^{-4}$	$.24 \cdot 10^{-5}$	$.87 \cdot 10^{-8}$	2.66	4.55
−.80	$.68 \cdot 10^{-3}$	$.13 \cdot 10^{-3}$	$.94 \cdot 10^{-5}$	$.13 \cdot 10^{-6}$	2.16	3.72
−.70	$.11 \cdot 10^{-2}$	$.23 \cdot 10^{-3}$	$.27 \cdot 10^{-4}$	$.11 \cdot 10^{-5}$	2.08	2.97
−.60	$-.59 \cdot 10^{-2}$	$.34 \cdot 10^{-3}$	$.59 \cdot 10^{-4}$	$.55 \cdot 10^{-5}$	4.46	2.42
−.50	$-.44 \cdot 10^{-1}$	$-.23 \cdot 10^{-3}$	$.94 \cdot 10^{-4}$	$.16 \cdot 10^{-4}$	7.09	2.06
−.40	−.14	$-.24 \cdot 10^{-1}$	$-.22 \cdot 10^{-2}$	$.26 \cdot 10^{-4}$	2.38	3.29
−.30	−.31	−.16	$-.84 \cdot 10^{-1}$	$-.50 \cdot 10^{-1}$	1.01*	1.12*
−.20	$.78 \cdot 10^{-2}$	$.87 \cdot 10^{-2}$	$.37 \cdot 10^{-1}$	$.62 \cdot 10^{-2}$	−5.15*	−.12*
−.10	.28	$.67 \cdot 10^{-1}$	$-.14 \cdot 10^{-1}$	$-.46 \cdot 10^{-2}$	1.38	3.06
.0	.40	.15	$.47 \cdot 10^{-1}$	$.17 \cdot 10^{-1}$	1.23	1.85
.10	.29	$.83 \cdot 10^{-1}$	$-.10 \cdot 10^{-1}$	$-.67 \cdot 10^{-2}$	1.14	4.66
.20	$.29 \cdot 10^{-1}$	$-.59 \cdot 10^{-2}$	$.23 \cdot 10^{-1}$	$.35 \cdot 10^{-2}$	.29*	.57*
.30	−.32	−.16	$-.76 \cdot 10^{-1}$	$-.46 \cdot 10^{-1}$	.85*	1.47*
.40	−.14	$-.19 \cdot 10^{-1}$	$-.11 \cdot 10^{-2}$	$-.29 \cdot 10^{-4}$	2.74	4.07
.50	$-.43 \cdot 10^{-1}$	$-.89 \cdot 10^{-3}$	$-.81 \cdot 10^{-4}$	$-.96 \cdot 10^{-5}$	5.71	3.49
.60	$-.80 \cdot 10^{-2}$	$-.30 \cdot 10^{-3}$	$-.36 \cdot 10^{-4}$	$-.25 \cdot 10^{-5}$	4.85	2.97
.70	$-.16 \cdot 10^{-2}$	$-.18 \cdot 10^{-3}$	$-.15 \cdot 10^{-4}$	$-.51 \cdot 10^{-6}$	3.05	3.56
.80	$-.95 \cdot 10^{-3}$	$-.95 \cdot 10^{-4}$	$-.51 \cdot 10^{-5}$	$-.77 \cdot 10^{-7}$	3.25	4.16
.90	$-.48 \cdot 10^{-3}$	$-.39 \cdot 10^{-4}$	$-.15 \cdot 10^{-5}$	$-.87 \cdot 10^{-8}$	3.56	4.65
1.0	$-.24 \cdot 10^{-4}$	$.60 \cdot 10^{-5}$	$.66 \cdot 10^{-7}$	$-.43 \cdot 10^{-9}$	2.35	6.49

where

$$S = \left\{ (x, y) : |x - y| < \frac{1}{\sqrt{2}},\ |x + y| < \frac{1}{\sqrt{2}} \right\}$$

is a unit square rotated by an angle of $\frac{\pi}{4}$ (due to Harten, see [9, Example 3]):

$$u(x, y, 0) = \begin{cases} 1 - w & \text{with } w = 2(x^2 + 2y^2) \quad \text{if } w \le 1, \\ 0 & \text{otherwise.} \end{cases} \tag{72}$$

For both examples we have used $\Delta x = \Delta y = \frac{1}{20}$ and a grid of 40×40 points. We solve (71) at $t = 2$ using the PHM-REF scheme to study the smearing and stability of our method and the effect of artificial compression. In Figs. 3(c), (d) and 4(a)–(d), the numerical resolutions are displayed including three dimensional plots and cross sections $x = 0$ and $x = 0.4$. In spite of the smearing near discontinuities, we have observed that the symmetry and the shape of the function are relatively well preserved as well as the efficiency of the artificial compression relative to each cross section. On the other hand, we remark that the smearing near discontinuities for the central cross sections ($x = 0$) is more asymmetric than the one observed for lateral cross sections. For some time we have observed the good behavior (concerning smearing and stability) of the CPHM-REF method and we have run it for (71) at $t = 16$ (8 periods); the numerical results are displayed in Figs. 5(a) and 5(b).

The second initial data (72) is an elliptic paraboloid truncated by the (x, y)-plane. We use this example to study the smearing of corners (jumps in first partial derivatives) in different directions. Figures 5(c) and 5(d) and 6(a)–(c) represent the numerical solution for the CPHM-REF method at $t = 2$ with CFL $= 0.2$ including a three-dimensional (3D) plot and cross sections $x = 0$, $y = 0$, $x = -0.55$, and $y = -0.4$. In this case, we observe that our method smears less in the x-direction than in the y-direction due to the asymmetry of the initial function, and this phenomenon makes the cross sections less accurate than it is expected. For

lateral cross sections, the loss of accuracy at local extrema is also due to the smearing (of the jump in the partial derivative) coming from the other direction and vice versa.

Example 6. We solve a Riemann problem for the 2D Burgers' equation

$$u_t + \left(\frac{u^2}{2}\right)_x + \left(\frac{u^2}{2}\right)_y = 0 \tag{73}$$

with the following Riemann data:

$$u(x, y, 0) = \begin{cases} u_1 & \text{if } x > 0, y > 0, \\ u_2 & \text{if } x < 0, y > 0, \\ u_3 & \text{if } x < 0, y < 0, \\ u_4 & \text{if } x > 0, y < 0. \end{cases} \tag{74}$$

We represent a 2D Riemann data by a four-component vector (u_1, u_2, u_3, u_4), defining a step function by means of (74). There are eight essentially different solution types depending on the order of the numbers u_i's, (see [12] for details). We have tested the PHM-REF scheme and we have observed that it converges to the entropy solution with good resolution in all eight cases. As a sample, in Fig. 6(d) we display the solution of the Riemann problem corresponding to the vector $(-1, -0.2, 0.5, 0.8)$ illustrating the resolution of a shock that is not aligned with the computing grid. The experiment was performed for 50×50 grid points using $\Delta x = \Delta y = \frac{1}{20}$.

5. Concluding remarks. PHM upwind schemes based on fluxes and the Shu–Osher third order TVD Runge–Kutta method designed for scalar conservation laws seemed to work very well in our preliminary numerical tests. Two main advantages with respect to the third order ENO scheme based on fluxes were found: PHM upwind schemes are not sensitive to the CFL number and they are more local in the sense that numerical fluxes depend on four values. In spite of the loss of accuracy of our schemes at local extrema where they may degenerate to $O(h^{\frac{3}{2}})$, according to our numerical experiments, the behavior of the PHM schemes in presence of discontinuities is stable and the viscosity appears to be lower than that of ENO3-REF. To design our schemes, two reconstruction procedures based on piecewise hyperbolic approximations have been introduced and tested. The first procedure works satisfactorily with linear discontinuities and is unstable for nonlinear fluxes. The second procedure appears to be stable for nonlinear scalar conservation laws according to our experience. Some theoretical evidence about the stability of the second procedure was presented in §2.

Acknowledgments. This paper originated and was written when I was visiting the Department of Mathematics at the Univerisity of California at Los Angeles. I thank Stan Osher for several useful discussions and suggestions and for providing a good background about many challenging problems on nonlinear partial differential equations and numerical methods, Rosa Donat for many helpful discussions, and the referees for many valuable suggestions that made this paper substantially better.

REFERENCES

[1] P. COLELLA, AND P. R. WOODWARD, *The piecewise parabolic method for gas dynamics*, J. Comput. Phys., 54 (1984), pp. 174.

[2] A. HARTEN, *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.

[3] ———, *On a class of high resolution total variation stable finite difference schemes*, SIAM J. Numer. Anal., 21 (1984), pp. 1–23.

[4] A. HARTEN AND S. OSHER, *Uniformly high order accurate nonoscillatory schemes* I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.

[5] A. HARTEN, S. OSHER, B. ENGQUIST AND S. CHAKRAVARTHY, *Some results on uniformly high-order accurate essentially non-oscillatory schemes*, Appl. Numer. Math., 2 (1987), pp. 347–377.

[6] A. HARTEN, B. ENGQUIST, S. OSHER AND S. CHAKRAVARTHY, *Uniformly high order accurate essentially non-oscillatory schemes* III, J. Comput. Phys., 71 (1987), pp. 231–303.

[7] S. OSHER AND S. R. CHAKRAVARTHY, *High resolution schemes and the entropy condition*, SIAM J. Numer. Anal., 21 (1984), pp. 955–984.

[8] P. L. ROE, *Approximate Riemann solvers, parameter vectors, and difference schemes*, J. Comput. Phys., 43 (1981), pp. 357–372.

[9] C. W. SHU AND S. J. OSHER, *Efficient implementation of essentially non-Oscillatory shock capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[10] ———, *Efficient implementation of essentially non-Oscillatory shock capturing schemes* II, J. Comput. Phys., 83 (1989), pp. 32–78.

[11] H. YANG, *An artificial compression method for* ENO *schemes. The slope modification method*, J. Comput. Phys., 89 (1990), pp. 125–160.

[12] D. WAGNER, *The Riemann problem in two space dimensions for a single conservation law*, SIAM J. Math. Anal., 14 (1983), pp. 534–559.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 916–937, July 1994

008

A DISPERSION ANALYSIS OF FINITE ELEMENT METHODS FOR MAXWELL'S EQUATIONS*

P. B. MONK† AND A. K. PARROTT‡

Abstract. A dispersion analysis of six different finite element methods for approximating Maxwell's equations in space is presented. The study is limited to two space dimensions and triangular elements, but all the methods considered can be generalised to three dimensions using prismatic elements. Spatial grids consisting of both equilateral and right triangles are considered.

Key words. Maxwell's equations, finite elements, dispersion, mixed methods

AMS subject classifications. 65N30, 78–08

1. Introduction. In this report we shall use a dispersion analysis to compare a variety of different finite elements methods for the spatial discretization of Maxwell's equations. To simplify the computations we shall limit ourselves to considering a two-dimensional model problem, and we shall also consider only triangular elements (see [9] for similar, but less detailed, results on quadrilateral elements).

The two-dimensional Maxwell equations considered here govern the behaviour of the electric field $\boldsymbol{E} = (E_1(\boldsymbol{x}, t), E_2(\boldsymbol{x}, t))$ and the scalar magnetic field $H = H(\boldsymbol{x}, t)$ (where $\boldsymbol{x} = (x_1, x_2) \in \mathbf{R}^2$). Precisely, the fields satisfy the following system:

$$\boldsymbol{E}_t - \vec{\nabla} \times H = 0 \qquad \text{in } \mathbf{R}^2, \tag{1a}$$

$$H_t + \nabla \times \boldsymbol{E} = 0 \qquad \text{in } \mathbf{R}^2, \tag{1b}$$

where $\boldsymbol{E}_t$ denotes the time derivative of $\boldsymbol{E}$ (and similarly H_t) and the two curls are defined by

$$\vec{\nabla} \times H = \left(\frac{\partial H}{\partial y}, -\frac{\partial H}{\partial x} \right)^T \quad \text{and} \quad \nabla \times \boldsymbol{E} = \frac{\partial E_2}{\partial x} - \frac{\partial E_1}{\partial y}. \tag{2}$$

In (1), we have set the constitutive parameters ϵ and μ to one so that the speed of light for these equations is unity.

To specify uniquely a solution of the Maxwell system (1), initial and boundary conditions (or conditions at infinity) are required. However in a dispersion analysis one is interested in plane wave solutions of the equations, and boundary or initial conditions are ignored. For a standard dispersion analysis of the continuous system (1) we seek solutions of the form

$$\boldsymbol{E} = \boldsymbol{E}_0 \exp(i(\boldsymbol{k} \cdot \boldsymbol{x} - \omega t)) \quad \text{and} \quad H = H_0 \exp(i(\boldsymbol{k} \cdot \boldsymbol{x} - \omega t)), \tag{3}$$

where ω, $\boldsymbol{k} = (k_1, k_2)$, $\boldsymbol{E}_0$ and H_0 are independent of $\boldsymbol{x}$ and t. Substituting for $\boldsymbol{E}$ and H in (1) we find that there are three possible values of ω for any given value of $\boldsymbol{k}$:

$$\omega = |\boldsymbol{k}|, \quad \omega = -|\boldsymbol{k}|, \quad \text{and} \quad \omega = 0. \tag{4}$$

The relation between ω and $\boldsymbol{k}$ is called the dispersion relation. When $\omega \neq 0$, the wave vector $\boldsymbol{k}$ is orthogonal to $\boldsymbol{E}_0$ (so that $\boldsymbol{E}_0 \cdot \boldsymbol{k} = 0$) and the electric field is divergence free. For the

*Received by the editors October 13, 1992; accepted for publication April 30, 1993.

†Department of Mathematical Sciences, University of Delaware, Newark, Delaware 19716 (monk@math.udel.edu). The research of this author was supported in part by a grant from Air Force Office of Scientific Research.

‡Oxford University Computing Laboratory, 11 Keble Road, Oxford OX1 3QD, England (kevin.parrott@comlab.ox.ac.uk).

stationary solution (when $\omega = 0$), the magnetic field is zero ($H_0 = 0$) and $\boldsymbol{k} \times \boldsymbol{E}_0 = 0$. In this case the electric field is curl free.

From the dispersion relation, we can easily compute the phase speed of the wave $c_\phi = \omega/|\boldsymbol{k}|$ and the group velocity of the wave $v_g = \nabla_k \omega$. Of these the phase speed (which is exactly unity for the continuous problem) is easier to compute, but the group velocity is more important since it governs the velocity of energy transport in the system [20].

In analyzing the dispersion behaviour of numerical schemes for approximating (1), we suppose that the plane $\mathbf{R}^2$ is tiled by an infinite uniform triangulation. Then, for a given finite element method, we can compute the equations satisfied by the degrees of freedom of the method for the given mesh, and we seek plane wave solutions of the discrete equations. The dispersion relation for the discrete problem shows how plane waves will propagate in the discrete problem.

Computation of the dispersion relations reported in this paper was carried out using Mathematica.

2. Discretization. The mesh we shall use is the infinite uniform mesh generated by the triangle with vertices $(0, 0)$, $(h, 0)$, and $(a, b)h$, where (a, b) is fixed for any given family of meshes and h is an overall scale factor for the size of the triangles in the mesh. Equivalently, the mesh is generated by translates of the "unit" quadrilateral with vertices $(0, 0)$, $(h, 0)$, $((a + 1), b)h$, and $(a, b)h$ (made up of the two triangles with vertices $(0, 0)$, $(h, 0)$, $(a, b)h$, and $(h, 0)$, $((a + 1), b)h$, $(a, b)h$). A portion of the mesh is shown in Fig. 1. On the given mesh, we can construct a finite element discretization of Maxwell's system by using a variety of combinations of finite element spaces for the electric and magnetic fields.

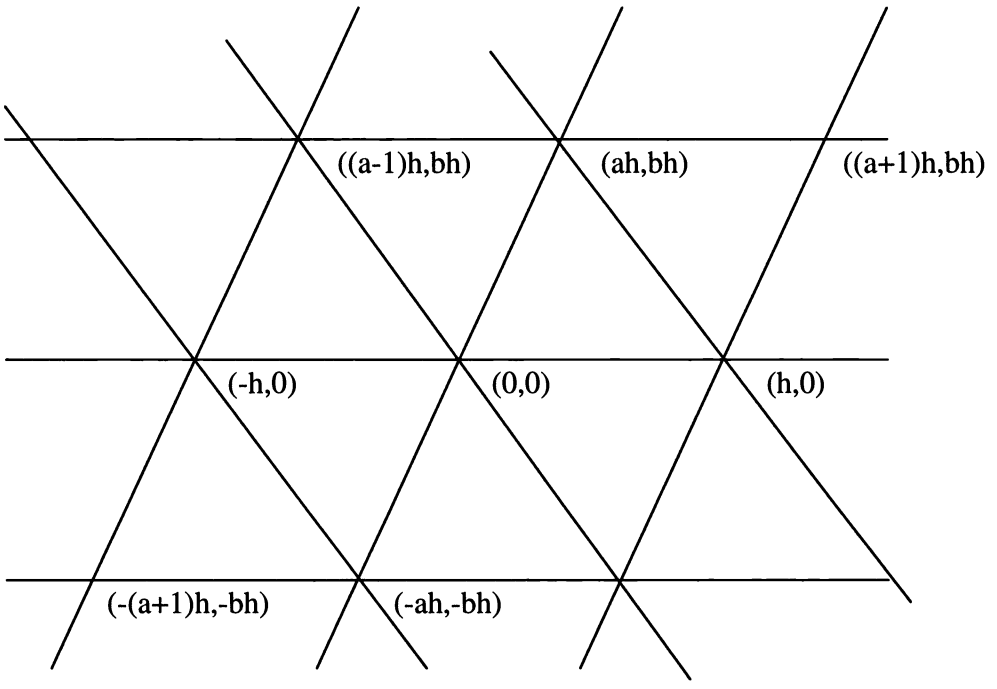


FIG. 1. *A portion of the infinite uniform mesh used in this paper. The mesh is generated by translates (and rotated translates) of the triangle with vertices* $(0, 0)$, $(h, 0)$, *and* $(a, b)h$.

We select a set of vector basis functions $\{\phi_i\}_{i=1}^{\infty}$ for the electric field, and a set of scalar basis functions $\{\psi_j\}_{j=1}^{\infty}$ for the magnetic field. Then the approximate electric field denoted $\boldsymbol{E}^h$

and the approximate magnetic field denoted H^h are given by

$$\boldsymbol{E}^h(\boldsymbol{x},t)=\sum_{i=1}^{\infty}E_i(t)\boldsymbol{\phi}_i(\boldsymbol{x}) \quad \text{and} \quad H^h(\boldsymbol{x},t)=\sum_{j=1}^{\infty}H_j(t)\psi_j(\boldsymbol{x}),$$

where the time-dependent coefficients $\{E_i\}$ and $\{H_j\}$ must be determined. Let us denote by $(\boldsymbol{u},\boldsymbol{v})=\int_{\mathbf{R}^2}\boldsymbol{u}\cdot\boldsymbol{v}\,dA$. Using the method of weighted residuals or Galerkin method, we determine the unknown coefficients as the solution of the following system of differential equations:

$$\sum_{i=1}^{\infty}\frac{dE_i}{dt}(\boldsymbol{\phi}_i,\boldsymbol{\phi}_k)-\sum_{j=1}^{\infty}H_j(\psi_j,\nabla\times\boldsymbol{\phi}_k)=0 \quad \forall k\geq 1, \tag{5a}$$

$$\sum_{j=1}^{\infty}\frac{dH_j}{dt}(\psi_j,\psi_l)+\sum_{i=1}^{\infty}E_i(\nabla\times\boldsymbol{\phi}_i,\psi_l)=0 \quad \forall l\geq 1. \tag{5b}$$

Notice that in deriving (5a) we have integrated the curl term by parts. Only the electric field basis functions $\{\boldsymbol{\phi}_i\}$ need to have a well-defined scalar curl, whereas the magnetic basis functions $\{\psi_j\}$ need not possess any global derivatives and so can be entirely discontinuous across element boundaries. The requirement that $\nabla\times\boldsymbol{\phi}_i$ be well defined implies that the tangential component of each of the basis functions must be continuous across interelement boundaries [14]. Thus even the basis $\{\boldsymbol{\phi}_i\}$ does not need to consist of continuous elements. We remark that even though (5) appears to involve infinite sums, the fact that the basis functions have small support implies that at a given point $\boldsymbol{x}$, all the sums involve only a small number of terms.

In describing the various finite element schemes, we shall make specific (but nonunique) choices for the degrees of freedom of each pair of spaces (one for the electric field and one for the magnetic field). The degrees of freedom then imply a particular basis for the finite element space. However, the dispersion relations (and the solution of the discrete finite element problem) do not depend on the choice of degrees of freedom provided the choice is conforming and unisolvent for the particular finite element space in question. Thus we can simply choose a convenient set of degrees for each space.

In the next section we shall describe some different constructions of the basis functions for $\boldsymbol{E}^h$ and H^h. We note that we need only describe the basis functions and the degrees of freedom on a single triangle, since all the finite elements under consideration are defined elementwise.

3. Finite element spaces. In this section we shall present the finite element schemes analyzed in this paper. We start with the simplest case of continuous linear elements.

3.1. Linear-linear elements. The simplest choice for the finite element spaces for discretizing $\boldsymbol{E}$ and H is to use standard continuous piecewise linear elements for both fields. Thus, on a given triangle K, the electric field is linear and has standard degrees of freedom. If P_1 denotes the set of linear polynomials, then

$$\boldsymbol{E}^h|_K\in(P_1)^2$$

and the degrees of freedom (unknowns) are the values of the vector field at the mesh vertices. We denote the set of electric degrees of freedom on triangle K by Σ_K^E and we write

$$\Sigma_K^E=\left\{\boldsymbol{E}^h(\boldsymbol{a}_i),\quad 1\leq i\leq 3\right\},$$

where $\boldsymbol{a}_i$, $1 \le i \le 3$ are the coordinate vectors of the vertices of triangle K. When we describe other sets of elements, we shall simply give the set of basis functions and the set of degrees of freedom.

For the magnetic field, we also use continuous piecewise linear elements so that on each triangle K:

$$H^h|_K \in P_1 \quad \text{and} \quad \Sigma_K^H = \left\{H^h(\boldsymbol{a}_i),\ 1 \le i \le 3\right\}.$$

The arrangement of degrees of freedom are shown in Fig. 2. It is clear that in this uniform mesh all vertices are equivalent in the sense that the same equations will hold at each vertex. Thus, to compute the dispersion relation we need only compute the equations for $\boldsymbol{E}_1$ and H_1. If we denote by $\boldsymbol{E}_i = (E_{i1}, E_{i2})$, trivial computations show that

$$h\frac{d}{dt}\left[H_1 + \frac{1}{6}(H_2 + H_3 + H_4 + H_5 + H_6 + H_7)\right]$$
$$+\frac{1}{3}\left(\left(\frac{2a-1}{b}\right)E_{21} + 2E_{22} + \left(\frac{a-2}{b}\right)E_{31} + E_{32} - \left(\frac{a+1}{b}\right)E_{41} - E_{42}\right.$$
$$\left. - \left(\frac{2a-1}{b}\right)E_{51} - 2E_{52} - \left(\frac{a-2}{b}\right)E_{61} - E_{62} + \left(\frac{a+1}{b}\right)E_{71} + E_{72}\right) = 0, \tag{6a}$$

$$h\frac{d}{dt}\left[E_{11} + \frac{1}{6}(E_{21} + E_{31} + E_{41} + E_{51} + E_{61} + E_{71})\right]$$
$$-\frac{1}{3}\left(\left(\frac{1-2a}{b}\right)H_2 + \left(\frac{2-a}{b}\right)H_3 + \left(\frac{a+1}{b}\right)H_4 + \left(\frac{2a-1}{b}\right)H_5\right.$$
$$\left. + \left(\frac{a-2}{b}\right)H_6 - \left(\frac{a+1}{b}\right)H_7\right) = 0, \tag{6b}$$

$$h\frac{d}{dt}\left[E_{12} + \frac{1}{6}(E_{22} + E_{32} + E_{42} + E_{52} + E_{62} + E_{72})\right]$$
$$-\frac{1}{3}(-2H_2 - H_3 + H_4 + 2H_5 + H_6 - H_7) = 0. \tag{6c}$$

Equations (6) have one important feature: the terms multiplying the time derivatives are quite simple and do not involve any geometric information (apart from the number and area of the triangles meeting at a vertex). This suggests (as we shall see later) that the linear-linear element family may be mass lumped in a simple way.

The linear-linear scheme described above is a two-dimensional generalisation of the lowest order spline-Galerkin scheme of [19], [11]. Given the very high order of convergence of the one-dimensional spline-Galerkin scheme for the wave equation ($O(h^4)$ at the meshpoints), one might hope that some of these accuracy properties will carry over to the two dimensional case (this will certainly happen if we use a tensor product method on squares, but we are using triangular elements).

To compute the dispersion relation, we substitute for $\boldsymbol{E}$ and H in (6) using (3). The resulting matrix eigenvalue problem gives ω in terms of $\boldsymbol{k}$. For each triangulation (i.e., each choice of (a, b)) we obtain three possible dispersion relations giving ω as a function of $|\boldsymbol{k}|$ and h (see (4) for the continuous case). We report results for two triangulations. The first is when the triangles are equilateral so that $(a, b) = (1/2, \sqrt{3}/2)$. Then if we define $\boldsymbol{\xi} = \boldsymbol{k}h$ we obtain that

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_1^6}{240} - \frac{\xi_1^4\xi_2^2}{32} - \frac{\xi_2^6}{160} + \text{higher order terms} \tag{7}$$

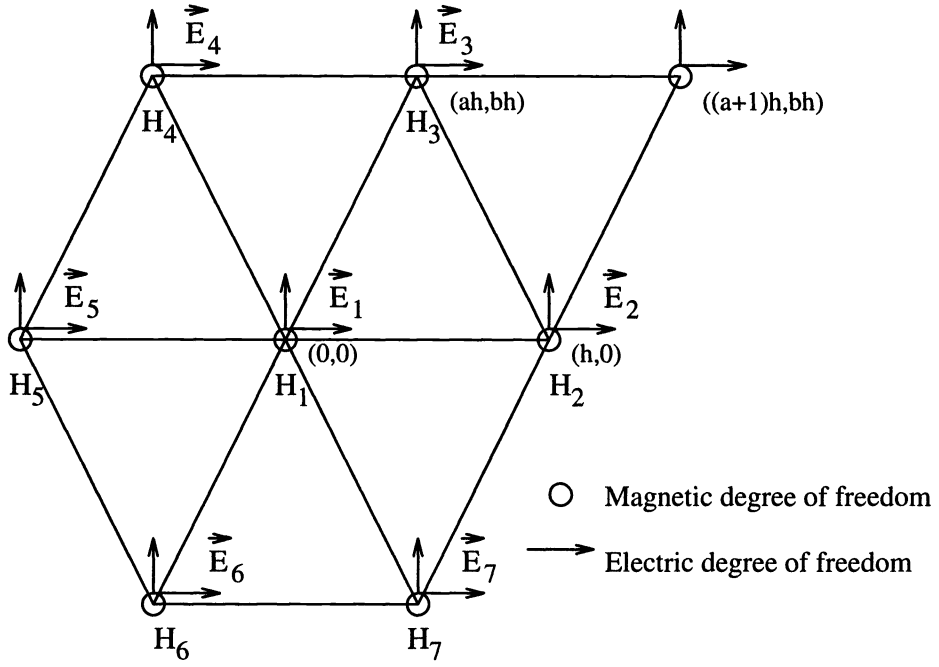


FIG. 2. *Notation used in the discretization of the electric and magnetic fields using continuous piecewise linear elements. The degrees of freedom for the fields are associated with the vertices of the triangles, and the electric degree of freedom is a vector (so* $\boldsymbol{E}_i = \tilde{E}_i = (E_{i1}, E_{i2})$*).*

(the other root is $\omega = 0$ as expected). The above expansion for ω^2 shows that $\omega = |\boldsymbol{k}| + O(h^4)$ so the dispersion relation is approximated to fourth-order accuracy. This is extremely accurate given that we are using linear elements. More remarkably, this fourth-order accuracy is also seen on a rather anisotropic mesh consisting of right-angled triangles. For this triangulation we choose $(a, b) = (0, 1)$ and obtain the following dispersion relation:

$$(8)\quad (\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_1^6}{90} + \frac{\xi_1^5\xi_2}{36} - \frac{\xi_1^4\xi_2^2}{36} - \frac{\xi_1^2\xi_2^4}{36} + \frac{\xi_1\xi_2^5}{36} - \frac{\xi_2^6}{90} + \text{higher order terms.}$$

Again the dispersion relation is approximated to fourth order in h.

The Taylor series for ω given above show the behaviour of ω close to $\boldsymbol{k} = 0$. In Fig. 3(a) we plot the phase speed for the linear-linear elements as a function of $\boldsymbol{\xi}$ on equilateral triangles (we plot phase speed since for some other elements considered in this report we were unable to obtain the dispersion relation in closed form and so could not compute the group velocity in all cases). In Fig. 4(a) we plot the phase speed on the right triangle mesh. The contour lines in these figures are 0.02 apart, so that the size of the region in the contour map containing the origin corresponds to those values of $\boldsymbol{\xi}$ (and hence $\boldsymbol{k}$ and h) in which the phase speed differs by at most two percent from the exact value. This arbitrary value is chosen so that the relative accuracy of different methods can be compared visually.

We can conclude that from the point of view of dispersion, the combination of continuous linear-linear elements is very powerful. However, dispersion analysis does not provide information on the behaviour of the numerical scheme at material interfaces or boundary conditions. Let us point out that a disadvantage of the linear-linear scheme is that the standard perfect conducting boundary condition is tricky to implement and that the finite element scheme must be modified at interfaces where there is a jump in ϵ (since the electric field is not continuous

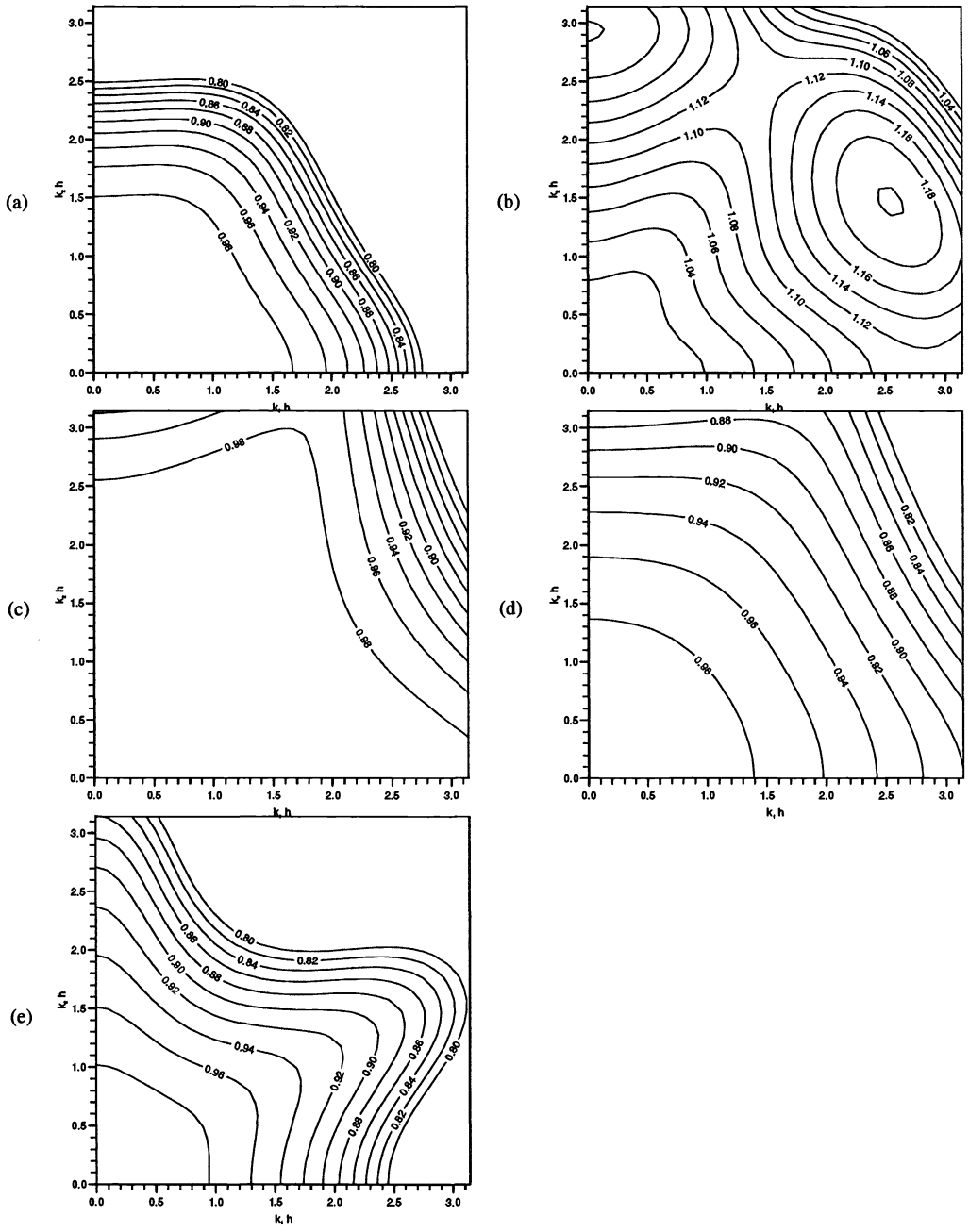


FIG. 3. *Contour maps of phase speed error as a function of* $\xi = kh$ *for five of the finite element schemes considered in this report. Of course the exact phase speed is always unity. In this case the grid is a uniform mesh of equilateral triangles. The contours are* 0.02 *apart* (2% *of correct speed*). *In parts* (a) *and* (c)–(e) *the computed phase velocity is less than the real phase velocity* (*at least when* $|\xi|$ *is small enough*). *Key*: (a) *linear-linear*, (b) *linear-constant*, (c) *edge*(1)*-constant*, (d) *nonconforming constant* (*modified*), *and* (e) *edge*(1)*-linear. Here edge*(1) *refers to the first type edge elements described in* §3.3.

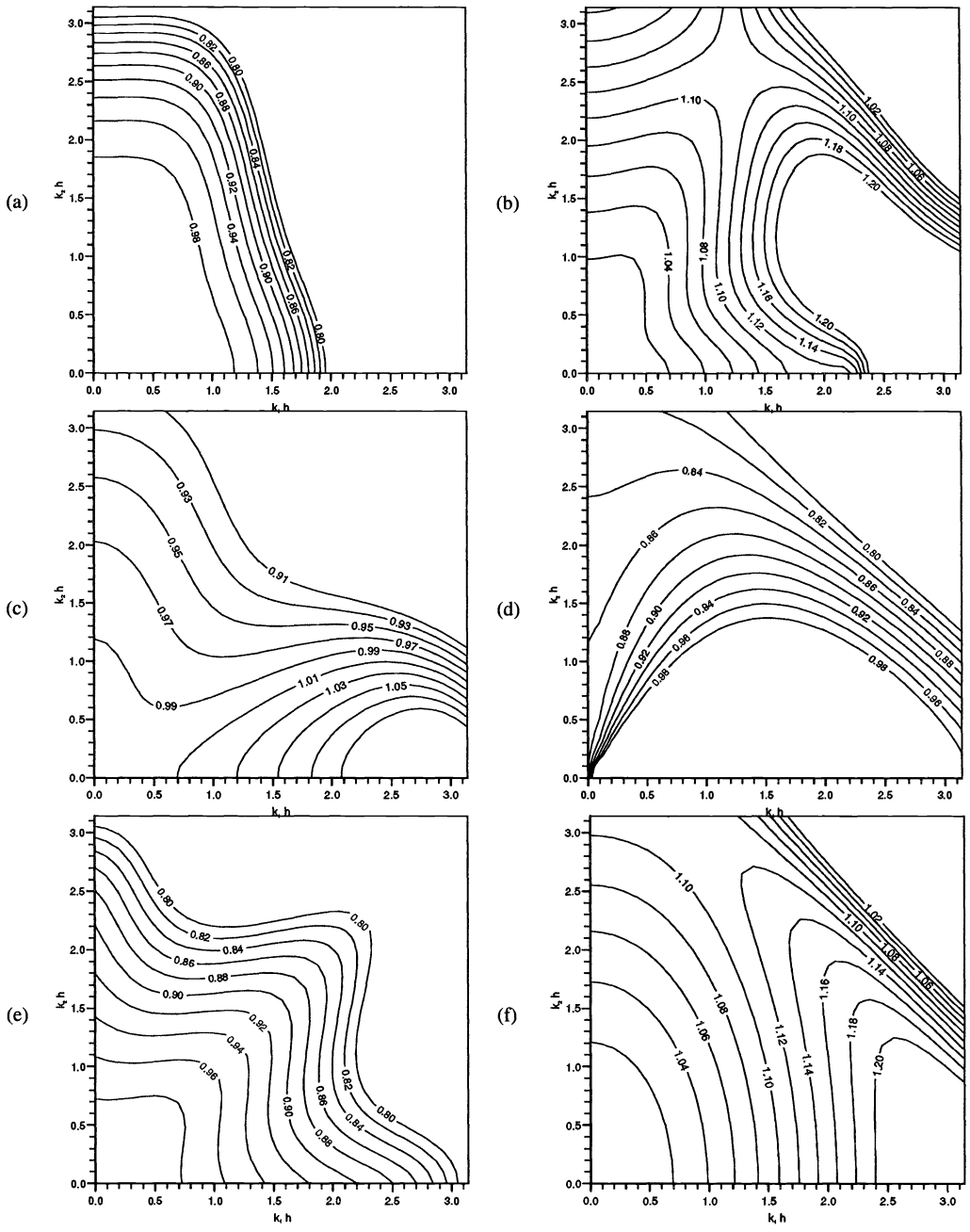


FIG. 4. *Contour maps of phase speed error for the six finite element schemes considered in this report. In this case the grid is a uniform mesh of right-angle triangles. Here we plot* $c_\phi(x\cos(\pi/4) + y\sin(\pi/4), -x\sin(\pi/4) + y\cos(\pi/4))$ *against x and y. This rotation is chosen to display fully the phase speed for the symmetries in this mesh. The contours are* 0.02 *apart* (2% *of correct speed*). *Key*: (a) *linear-linear*, (b) *linear-constant*, (c) *edge*(1)*-constant*, (d) *nonconforming constant* (*modified*), (e) *edge*(1)*-linear, and* (f) *edge*(2)*-constant. Here edge*(1) *refers to the first type edge elements described in* §3.3 *and edge*(2) *refers to the second type edge elements described in* §3.6. *In* (c) *the phase velocity is greater or less than unity depending on the direction of propagation, so we have drawn velocity contours on either side of unity.*

there cf. [5]). A further disadvantage of the method is that charge is not conserved exactly (even in a discrete sense) and the magnetic field is not exactly divergence free. The problems of boundary conditions, interfaces, and conservation make this choice inappropriate for some applications.

We also note that the numerical results in [7] show that the linear-linear scheme on a quadrilateral grid performs rather poorly. This may be because the space- and time-discretization methods are not well balanced (see §5) or may reflect problems with boundary conditions (there are no interfaces in the relevant examples in [7]). Whether these results hold on triangular meshes needs to be investigated.

3.2. Linear-constant elements. In order to obtain a method in which the magnetic field $\boldsymbol{H}$ is exactly divergence free in three dimensions, we can modify the magnetic field space. We use the standard continuous piecewise linear basis for the electric field so that, as in §3.1,

$$\boldsymbol{E}^h|_K \in (P_1)^2 \quad \text{and} \quad \Sigma_K^E = \left\{\boldsymbol{E}^h(\boldsymbol{a}_i), \quad 1 \le i \le 3\right\}.$$

For the magnetic field we use discontinuous piecewise constant elements

$$H^h|_K \in P_0 \quad \text{and} \quad \Sigma_K^H = \left\{H^h\left(\frac{\boldsymbol{a}_1+\boldsymbol{a}_2+\boldsymbol{a}_3}{3}\right)\right\}.$$

The unknown for the magnetic field is associated (arbitrarily) with the centroid of the triangle.

In this case there are three distinct points in the unit cell. The two electric degrees of freedom are associated with the vertices of the mesh and there are two magnetic degrees of freedom each associated with the centroid of a triangle in the unit cell. If we use the notation in Fig. 5, we can easily derive the following equations for the distinct degrees of freedom:

$$h\frac{dH_1}{dt} + \left(-E_{12} - \left(\frac{a-1}{b}\right)E_{11} + E_{22} + \left(\frac{a}{b}\right)E_{21} - \left(\frac{1}{b}\right)E_{31}\right) = 0, \tag{9a}$$

$$h\frac{dH_7}{dt} + \left(E_{82} + \left(\frac{a-1}{b}\right)E_{81} - E_{32} - \left(\frac{a}{b}\right)E_{31} + \left(\frac{1}{b}\right)E_{21}\right) = 0, \tag{9b}$$

$$\begin{aligned} h\frac{d}{dt}\left[E_{11} + \frac{1}{6}(E_{21}+E_{31}+E_{41}+E_{51}+E_{61}+E_{71})\right] - \left(-\left(\frac{a-1}{b}\right)H_1\right. \\ \left. + \left(\frac{1}{b}\right)H_2 + \left(\frac{a}{b}\right)H_3 + \left(\frac{a-1}{b}\right)H_4 - \left(\frac{1}{b}\right)H_5 - \left(\frac{a}{b}\right)H_6\right) = 0, \end{aligned} \tag{9c}$$

$$\begin{aligned} h\frac{d}{dt}\left[E_{12} + \frac{1}{6}(E_{22}+E_{32}+E_{42}+E_{52}+E_{62}+E_{72})\right] \\ -(-H_1+H_3+H_4-H_6) = 0. \end{aligned} \tag{9d}$$

To derive the dispersion relation, we allow H_1, H_7, E_{11}, and E_{12} to vary proportionally to $\exp(i(\boldsymbol{k}\cdot\boldsymbol{x}-\omega t))$ since all four are in the unit cell. Substituting this dependence into the above equations and obtaining $H_2,\ldots,H_6$ and $\boldsymbol{E}_2,\ldots,\boldsymbol{E}_8$ by translation, we obtain a four-by-four matrix eigenvalue problem yielding four values for ω as a function of $\boldsymbol{k}$. Three of these are "physical" (corresponding to the three values of ω we obtained in (4) of §1), and one is nonphysical. We refer to this as a "parasitic" solution. We shall not discuss the parasitic mode more in this case, because this combination of elements can be ruled out as a practical choice on the grounds of accuracy. In Fig. 3(b) we plot the phase speed for the linear-constant elements as a function of ξ on equilateral triangles and in Fig. 4(b) we plot the phase speed on the right triangle mesh. A quick comparison with the phase speed curves of other methods considered in this paper shows that in all cases the linear-constant scheme is not particularly attractive.

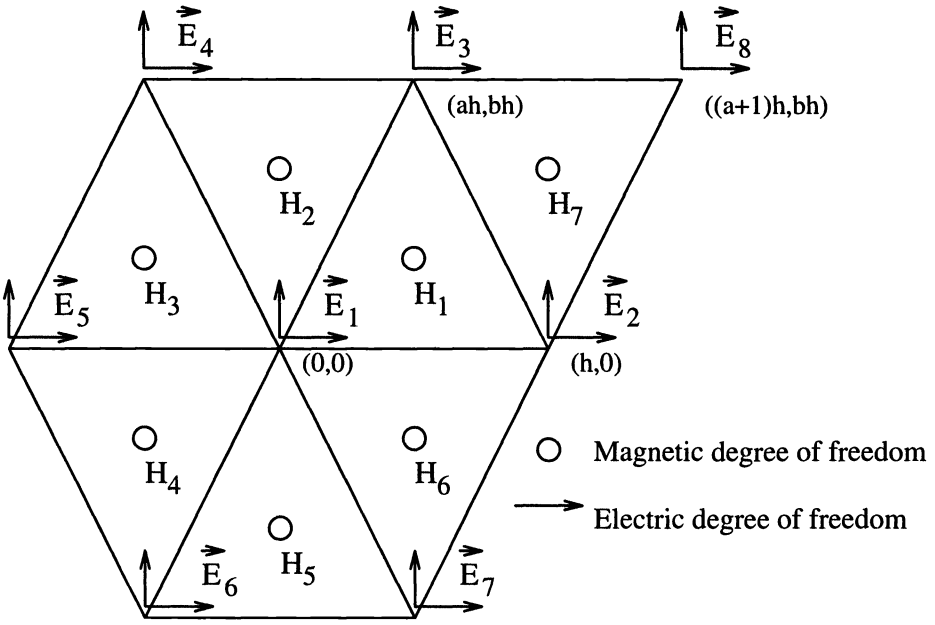


FIG. 5. *Notation used when continuous piecewise linear elements are used to discretize the electric field and piecewise constant elements are used to discretize the magnetic fields (the linear-constant method). The degrees of freedom for the electric field are associated with the vertices of the triangles and those of the magnetic field are associated with the centroids of the triangles.*

Unfortunately, we could not compute, in closed form, the series for the dispersion relation on equilateral triangles. However, for the right-triangle grid we find that

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 + \frac{1}{12}\left(\xi_1^4 - 2\xi_1^3\xi_2 + 4\xi_1^2\xi_2^2 + 2\xi_1\xi_2^3 + \xi_2^4\right) + \text{ higher order terms.}$$

Clearly, the dispersion relation is second order accurate.

3.3. Edge-constant elements. The linear-constant element pair discussed above was chosen to give an exactly divergence-free magnetic field in three dimensions, but all the problems regarding interfaces, boundaries, and charge conservation still remain. Fortunately, use of the edge elements of Nédélec [14] allows all these problems to be circumvented [2], [10]. In this section we shall analyze the use of first type edge elements for the electric field and piecewise constant elements for the magnetic field. To distinguish this element family from other edge element families, we shall sometimes refer to it as the edge(1)-constant family. Thus, on triangle K,

$$\boldsymbol{E}^h|_K = \begin{pmatrix} a_K + b_K y \\ c_K - b_K x \end{pmatrix},$$

where a_K, b_K, and c_K are constants on each triangle K. The degrees of freedom can be taken to be the value of the tangential component of the electric field at the midpoint of each edge (scaled by the length of the edge and h). We denote by $\boldsymbol{a}_{i,j}$ the coordinate of the midpoint of the edge joining $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ so that $\boldsymbol{a}_{i,j} = (\boldsymbol{a}_i + \boldsymbol{a}_j)/2$, and we denote by $l_{i,j}$ the length of this edge. The unit tangent $\boldsymbol{\tau}_{i,j} = (\boldsymbol{a}_j - \boldsymbol{a}_i)/l_{i,j}$. Then the degrees of freedom for the electric field

are

$$\Sigma_K^E = \left\{ \frac{l_{i,j}}{h} \boldsymbol{E}^h(\boldsymbol{a}_{i,j}) \cdot \boldsymbol{\tau}_{i,j}, \quad 1 \le i < j \le 3 \right\}.$$

We remark that this rather complex choice of degree of freedom is made to simplify the equations we shall compute.

The magnetic field space is again taken to be discontinuous piecewise constant elements:

$$H^h|_K \in P_0 \quad \text{and} \quad \Sigma_K^H = \left\{ H^h \left(\frac{\boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{a}_3}{3} \right) \right\}.$$

For this element, the node numbering and notation is summarised in Fig. 6. We easily derive the following equations for the degrees of freedom E_1, E_2, E_3, H_1, and H_2

$$h\frac{dH_1}{dt} + \frac{2}{b}(E_1 - E_2 + E_3) = 0, \tag{10a}$$

$$h\frac{dH_2}{dt} + \frac{2}{b}(E_6 - E_1 - E_7) = 0, \tag{10b}$$

$$h\left[I_{11}\frac{dE_1}{dt} + I_{12}\left(\frac{dE_2}{dt} + \frac{dE_6}{dt}\right) + I_{13}\left(\frac{dE_3}{dt} + \frac{dE_7}{dt}\right)\right] - (H_1 - H_2) = 0, \tag{10c}$$

$$h\left[I_{22}\frac{dE_2}{dt} + I_{12}\left(\frac{dE_1}{dt} + \frac{dE_9}{dt}\right) + I_{23}\left(\frac{dE_3}{dt} + \frac{dE_5}{dt}\right)\right] + (H_1 - H_3) = 0, \tag{10d}$$

$$h\left[I_{33}\frac{dE_3}{dt} + I_{13}\left(\frac{dE_1}{dt} + \frac{dE_8}{dt}\right) + I_{23}\left(\frac{dE_2}{dt} + \frac{dE_4}{dt}\right)\right] - (H_1 - H_4) = 0, \tag{10e}$$

where the scale factors I_{ij} are given as follows:

$$I_{11} = \frac{1}{6b}\left(1 + b^2 + a^2 + a\right) \qquad I_{12} = \frac{1}{12b}\left(1 - (b^2 + a(a-1))\right), \tag{11a}$$

$$I_{13} = -\frac{1}{12b}\left(b^2 + a^2 + a - 1\right) \qquad I_{22} = \frac{1}{6b}\left(b^2 + (a-1)^2 + 2 - a\right), \tag{11b}$$

$$I_{23} = \frac{1}{12b}\left(b^2 + (a-1)^2 - a\right) \qquad I_{33} = \frac{1}{6b}\left(3b^2 + a^2 + (a-1)^2 + a(a-1)\right). \tag{11c}$$

Note that in comparison to the equations for the linear-linear or linear-constant element families (see §3.1(6) and §3.2(9)), the above equations have extremely simple approximations to space-derivative terms (just block-centred differences for the magnetic terms in the electric field equations), but the time-derivative portion of the electric field equation contains complicated terms involving the geometry of the mesh. This suggests that mass lumping must be carried out with caution, and we shall discuss mass lumping in detail later (see §6).

The fact that there are five equations in (10) implies that there will be five values of ω corresponding to each $\boldsymbol{k}$. Three of these correspond to the exact dispersion relations and two are parasitic. Let us discuss the physical solutions first. One eigenvalue is always $\omega = 0$, and we need not discuss this case further. Corresponding to the exact dispersion relation $\omega^2 = |\boldsymbol{k}|^2$, we find a discrete dispersion relation that has an order of convergence depending on the mesh configuration. On an equilateral mesh, the dispersion relation is fourth-order accurate since

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_1^6}{3840} - \frac{\xi_1^4\xi_2^2}{256} + \frac{\xi_1^2\xi_2^4}{768} - \frac{7\xi_2^6}{11520} + \text{ higher order terms.}$$

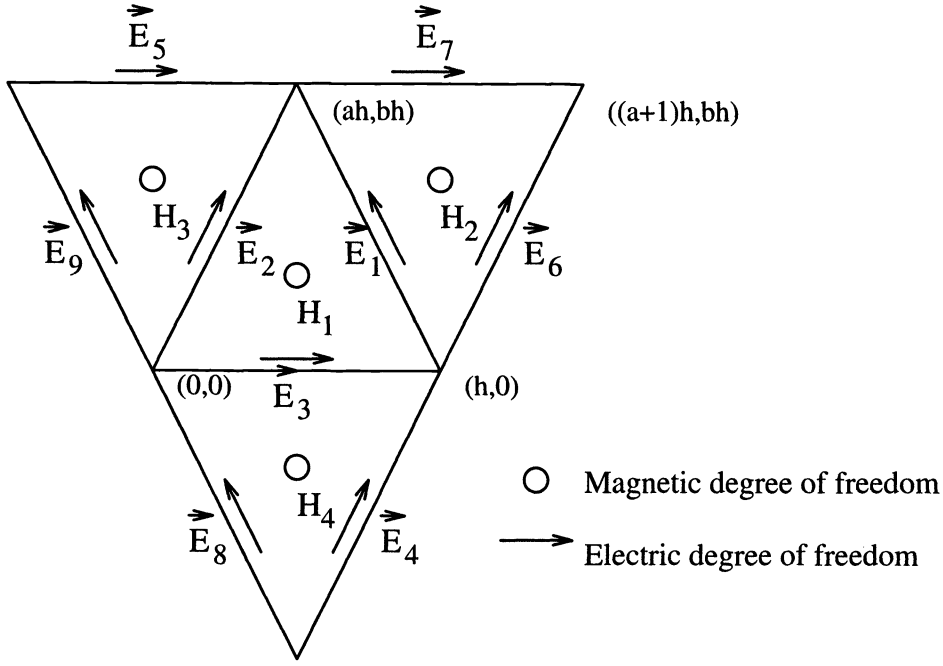


FIG. 6. *Position and numbering of the degrees of freedom for the space consisting of Nédélec's first type edge elements for the electric field and piecewise constant elements for the magnetic field. The degrees of freedom for the electric field are the tangential component of the field at the midpoint of each edge in the mesh. These degrees are also used for the nonconforming-constant space in* §3.4.

This compares very favourably with the corresponding dispersion relation for the linear-linear element family shown in (7) since the order of convergence is the same and the constants are smaller. Figure 3(c) shows that the phase velocity for the edge-linear elements on the equilateral grid and illustrates the surprising accuracy of these elements on an equilateral grid.

Unfortunately, the order of accuracy of the dispersion relation depends on the grid. Using the right-triangle grid we compute that the dispersion relation is now only second-order accurate:

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_1^4}{36} - \frac{\xi_1^3\xi_2}{18} + \frac{\xi_1^2\xi_2^2}{9} - \frac{\xi_1\xi_2^3}{18} - \frac{\xi_2^4}{36} + \text{ higher order terms.}$$

This dispersion relation does not compare well with the fourth-order convergent relation derived using linear-linear elements (see (8)). Figure 4(c) shows the phase velocity as a function of $\boldsymbol{\xi}$ for the edge-constant scheme. The accuracy of the phase velocity compares well with all methods except the linear-linear family, although the waves travel with a phase speed larger than unity in some directions (for example, $\boldsymbol{k} = (1, -1)$) and smaller than unity in other (for example, $\boldsymbol{k} = (1, 1)$).

For the edge-constant family, the parasitic modes have a phase speed that is asymptotically infinite as h tends to zero, but this may be an artifact of the analysis (see §4). On the equilateral grid, the dispersion relation for the two parasitic modes is

$$(\omega h)^2 = 48 - 4\xi_1^2 - 4\xi_2^2 + \text{ higher-order terms,}$$

while on the right-triangle grid the dispersion relation is

$$(\omega h)^2 = 36 - 5\xi_1^2 + 8\xi_1\xi_2 - 5\xi_2^2 + \text{ higher-order terms.} \tag{12}$$

The parasitic modes persist for all $\boldsymbol{k}$ and in the limit as $\boldsymbol{k} \to 0$ (for fixed h), we find a time-dependent solution with $\omega^2 = 48/h^2$ in the case of equilateral grids and $\omega^2 = 36/h^2$ for the right-triangle grid. For the right-triangle grid, the limiting eigenfunction corresponding to $\omega = 6/h$ is (using the notation of Fig. 6)

$$E_1 = 1, \quad E_2 = -1, \quad E_3 = 1, \quad H_1 = i, \quad \text{and} \quad H_2 = -i.$$

The complex values for the magnetic field can be understood as phase shifts relative to the electric field.

At first sight it may seem that parasitic modes are surprising since it is known that the use of edge elements avoids "spurious" modes in cavity resonator computations [3], [6]. However, spurious modes and parasitic modes are different phenomena. Spurious modes are incorrect eigenvalues (either by virtue of magnitude or multiplicity) that pollute the smaller eigenvalues in the discrete spectrum, whereas the parasitic modes we report correspond to large eigenvalues but with an incorrect velocity.

3.4. Nonconforming-constant elements. The use of tangential degrees of freedom for the electric field allow the simple implementation of the perfect conducting boundary condition. Another element family with tangential degrees of freedom is the P_1-nonconforming elements. In this section we examine the use of P_1-nonconforming elements for the electric field and piecewise constant elements for the magnetic field. This combination of finite elements is suggested and computational results are presented in [1]. We shall use the notation summarised in Fig. 6. The formal definition of the spaces are as follows:

$$\boldsymbol{E}^h|_K = E_1\boldsymbol{\tau}_{2,3}\phi_1(\boldsymbol{x}) + E_2\boldsymbol{\tau}_{3,1}\phi_2(\boldsymbol{x}) + E_3\boldsymbol{\tau}_{1,2}\phi_3(\boldsymbol{x}) \in (P_1)^2,$$

where $\phi_i = 1 - 2\lambda_i$ and λ_i is the ith barycentric coordinate function (i.e., the linear function such that $\lambda_i(\boldsymbol{a}_j) = \delta_{i,j}$, $1 \le j \le 3$). The degrees of freedom can be taken to be the value of the tangential component of the electric field at the midpoint of each edge:

$$\Sigma_K^{\boldsymbol{E}} = \left\{ \frac{bh}{l_{i,j}} \boldsymbol{E}^h(\boldsymbol{a}_{i,j}) \cdot \boldsymbol{\tau}_{i,j}, \ 1 \le i < j \le 3 \right\}.$$

It is clear that this choice of space for the electric field cannot be good in general since the above electric field is discrete divergence free in the sense of [18]. Nevertheless, for divergence-free fields (such as the waves we are computing here) this choice of space should be good.

Of course, since the above space is nonconforming for the curl operator, the curl and integrals appearing in (5) must be understood as being defined element by element.

The magnetic field space is again taken to be discontinuous piecewise constant elements:

$$H^h|_K \in P_0 \quad \text{and} \quad \Sigma_K^H = \left\{ H^h \left(\frac{\boldsymbol{a}_1 + \boldsymbol{a}_2 + \boldsymbol{a}_3}{3} \right) \right\}.$$

On deriving the dispersion relation for this method, we see an immediate problem. Even on an equilateral grid, the wave speed is incorrect. In fact, as h tends to zero, the asymptotic wave speed is $\sqrt{2}$. This fact is pointed out in [1], where a Petrov–Galerkin and mass-lumping scheme is introduced to compensate for the incorrect velocity. In effect, this procedure modifies the coefficients of the time derivatives of the electric degrees of freedom by introducing a factor of two at suitable places. It is inappropriate for us to go into details of the Petrov–Galerkin mass-lumping scheme, and we instead report the modified equations directly:

$$h\frac{dH_1}{dt} + \frac{2}{b^2}\left(l_1^2 E_1 - l_2^2 E_2 + l_3^2 E_3\right) = 0, \tag{13a}$$

$$h\frac{dH_2}{dt} + \frac{2}{b^2}\left(l_6^2 E_6 - l_1^2 E_1 - l_7^2 E_7\right) = 0, \tag{13b}$$

$$h\frac{2}{3}\frac{dE_1}{dt} - (H_1 - H_2) = 0, \tag{13c}$$

$$h\frac{2}{3}\frac{dE_2}{dt} + (H_1 - H_4) = 0, \tag{13d}$$

$$h\frac{2}{3}\frac{dE_1}{dt} - (H_1 - H_3) = 0, \tag{13e}$$

where l_i is the length of edge on which degree of freedom E_i is defined (see Fig. 6).

As for the edge-constant element family, there are five eigenvalues ω for each $\boldsymbol{k}$. We shall discuss only the physical values. On an equilateral mesh this method has a second-order convergent dispersion relation and

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_1^4}{48} - \frac{\xi_1^2\xi_2^2}{24} - \frac{\xi_2^4}{48} + \text{ higher-order terms.}$$

Although only second order, the dispersion relation is quite isotropic, which might allow the use of a suitable time-stepping scheme to improve the order of accuracy.

Unfortunately, on the right triangle grid, the method in (13) fails as is shown in Fig. 4(d). The method gives an incorrect wave speed even in the limit as $h \to 0$. It is clear that the simple-minded modification of the discrete equations that we have presented here is not convergent for all meshes. It is possible that the Petrov–Galerkin mass-lumped scheme of [1] will work well on a general mesh, but it is clear from our results here that caution must be exercised when using nonconforming elements in the electromagnetic context.

3.5. Edge-linear elements. We have seen that the edge elements have the advantage over continuous elements from the point of view of charge conservation and the implementation of boundary conditions. On the other hand, standard continuous elements (see §3.1) seem to have better dispersion properties. In addition, the linear-linear element combination has the aesthetic advantage of treating both electric and magnetic fields identically, while the edge-constant element family treats the electric and magnetic fields quite differently. For these reasons it is tempting to consider a method based on using edge elements for both the electric and magnetic fields (in three dimensions). If prismatic elements are used in three dimensions and the method restricted to two dimensions (by assuming suitable polarization of the solution), one obtains the method presented in this subsection. The electric field space is the first type edge element space introduced in §3.3, and the magnetic field space is the standard piecewise linear continuous space. Thus

$$\boldsymbol{E}^h|_K = \begin{pmatrix} a_K + b_K y \\ c_K - b_K x \end{pmatrix}$$

with

$$\Sigma_K^{\boldsymbol{E}} = \left\{ \frac{l_{i,j}}{h}\boldsymbol{E}^h(\boldsymbol{a}_{i,j}) \cdot \boldsymbol{\tau}_{i,j} 1 \le i < j \le 3 \right\}$$

and

$$H^h|_K \in P_1 \quad \text{with} \quad \Sigma_K^H = \left\{ H^h(\boldsymbol{a}_i),\ 1 \le i \le 3 \right\}.$$

The notation used in deriving the finite difference equations for this pair of finite element spaces is shown in Fig. 7. Clearly, the unit cell has four degrees of freedom associated with

it: H_1, and E_1, E_2 and E_3 (so we expect one parasitic mode). We easily derive the following equations for the degrees of freedom:

$$\text{(14a)}\quad h\left[I_{11}\frac{dE_1}{dt}+I_{12}\left(\frac{dE_2}{dt}+\frac{dE_6}{dt}\right)+I_{13}\left(\frac{dE_3}{dt}+\frac{dE_7}{dt}\right)\right]-\frac{1}{3}(H_1-H_4)=0,$$

$$\text{(14b)}\quad h\left[I_{22}\frac{dE_2}{dt}+I_{12}\left(\frac{dE_1}{dt}+\frac{dE_9}{dt}\right)+I_{23}\left(\frac{dE_3}{dt}+\frac{dE_5}{dt}\right)\right]-\frac{1}{3}(H_5-H_2)=0,$$

$$\text{(14c)}\quad h\left[I_{33}\frac{dE_3}{dt}+I_{13}\left(\frac{dE_1}{dt}+\frac{dE_8}{dt}\right)+I_{23}\left(\frac{dE_2}{dt}+\frac{dE_4}{dt}\right)\right]-\frac{1}{3}(H_3-H_6)=0,$$

$$\text{(14d)}\quad h\frac{d}{dt}\left[H_1+\frac{1}{6}(H_2+H_3+H_5+H_6+H_7+H_8)\right]+\frac{2}{3b}(E_1-E_5-E_{10}-E_{11}+E_{12}+E_4)=0,$$

where the scale factors $I_{i,j}$ are given in (11).

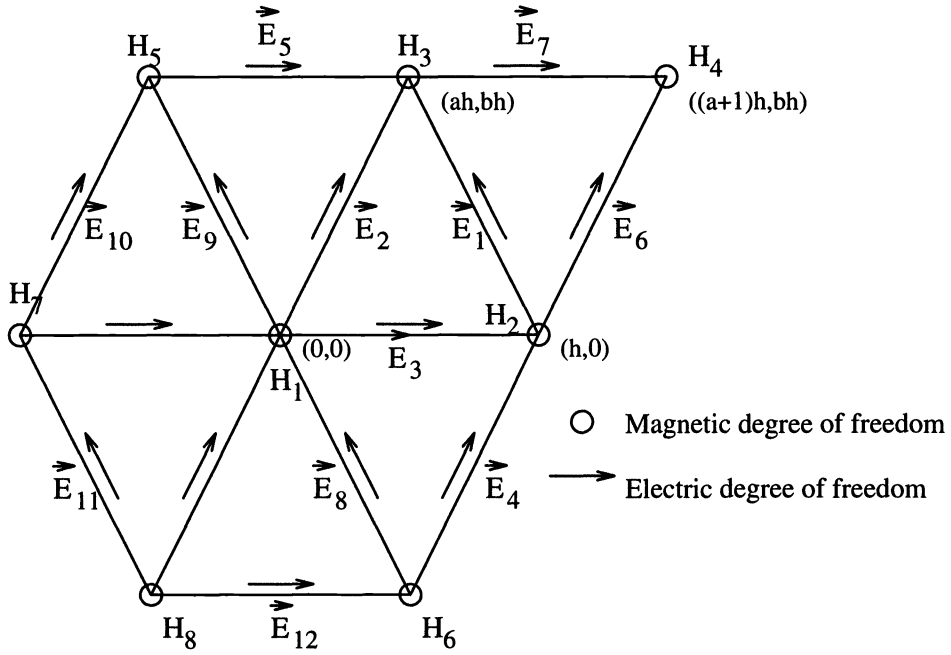


FIG. 7. *Notation used to describe the edge-linear element family. Degrees of freedom for the electric field are associated with edges of the triangles, while those for the magnetic field are associated with vertices.*

Using these equations we can compute the dispersion relation for this method (there is one parasitic mode). On equilateral triangles we find the disappointing result that the method is only second-order accurate since

$$(\omega h)^2=\xi_1^2+\xi_2^2-\frac{\xi_1^4}{24}-\frac{\xi_1^2\xi_2^2}{12}-\frac{\xi_2^4}{24}+\text{ higher-order terms.}$$

The error term above is isotropic, and thus can be compensated for to some extent by the correct choice of a time-marching scheme and a correct time step. A plot of the phase velocity as a function of ξ for this method is shown in Fig. 3(e). Obviously, the method is not as accurate as either the linear-linear or the edge-constant scheme.

On the right-angle triangle mesh, the method is again second order and we find that

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_1^4}{12} - \frac{\xi_2^4}{12} + \text{ higher-order terms.}$$

We remark that this dispersion relation is the same as for the Yee finite difference scheme on rectangles (at least to the order shown) [21]. The phase velocity is shown in Fig. 4(e). The method does not compare particularly favourably with the other methods. In view of these results we shall not discuss parasitic modes since it seems that the edge-linear scheme outlined here has little to offer in comparison with the edge-constant scheme.

3.6. Second type edge-constant elements. The last element family we shall consider is another edge element family due to Nédélec [15] (we refer to this family as second type edge elements or edge(2)-constant elements). This family was also suggested and used extensively in [13], [12]. In this case the electric field is discretized using an edge element family that contains all linear polynomials so that

$$\boldsymbol{E}^h|_K \in (P_1)^2$$

for each element K. This choice results in six degrees of freedom per triangle. There are a number of possible choices for these degrees and we elect to use the tangential component of the electrical field along each edge of the triangle at the vertices (see Fig. 8). Let us recall, however, that the dispersion relation is independent of this choice and depends only on the structure of the space under consideration. Thus

$$\Sigma_K^E = \left\{ \boldsymbol{E}(a_i) \cdot \tau_{i,j} \text{ for } 1 \le i, j \le 3 \text{ and } i \ne j \right\}.$$

For the magnetic field, we again use the piecewise constant field so that

$$H^h|_K \in P_1 \quad \text{with} \quad \Sigma_K^H = \left\{ H^h(\boldsymbol{a}_i),\ 1 \le i \le 3 \right\}.$$

We refer to the discretization as the edge(2)-constant element family.

In this case the unit cell has eight unknowns (and so we expect five parasitic modes). In fact, for the right-triangle grid, we find that $\omega = 0$ is an eigenvalue of multiplicity four, and that there is a pair of fast-moving parasitic solutions in addition to the physically realistic dispersion relations. The equations for this system are quite complex in general, and for this example we compute only the phase velocity as a function of $\boldsymbol{\xi}$ for the right-triangle grid (see Fig. 4(f)). In comparing Fig. 4(f) to other results we should remember that for a given mesh the second type edge elements have the most unknowns per unit area. In particular, there are twice as many electric field unknowns per unit area as compared to the first type edge elements (see Fig. 4(c)).

4. Parasitic modes and wave-guides. Here we want to explain the parasitic modes found while analysing some of the element families in the previous section (particularly the edge element families). We shall show that they are a natural by-product of the method of analysis. For simplicity we shall consider only the right-triangle mesh. When we compute the dispersion relation for edge elements (or other element families in the previous section), we consider a unit cell $K = [0, h] \times [0, h]$ and seek an approximate solution of Maxwell's equations on this cell such that the solution behaves like a plane wave when translated through a distance of h in the x or y direction. Precisely, we are computing a very crude (two triangle) finite element approximation to the problem of finding $\boldsymbol{u}$ and v such that

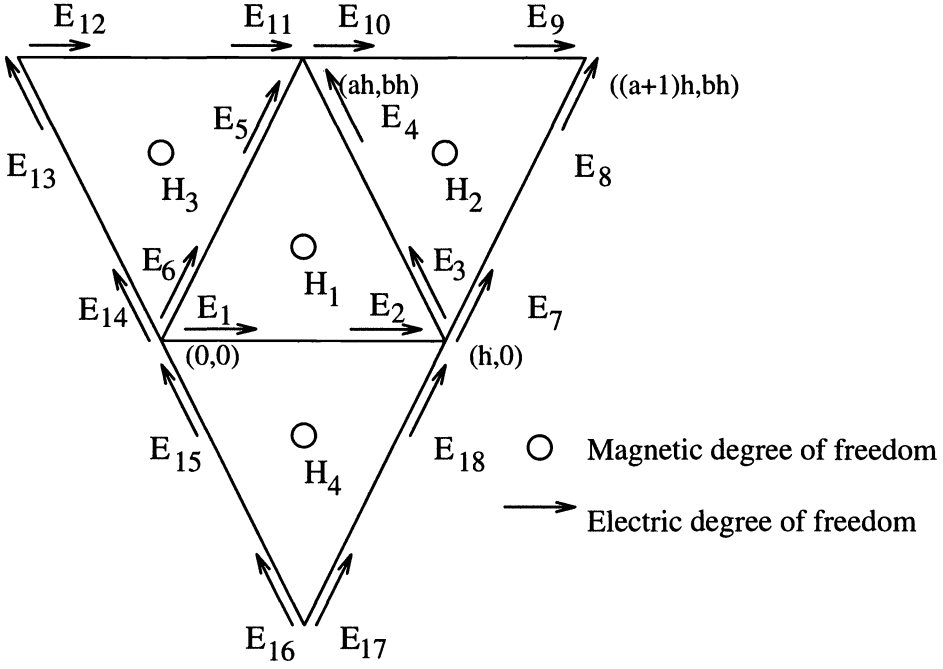


FIG. 8. *Position and numbering of the degrees of freedom for the space consisting of Nédélec's second type edge elements for the electric field and piecewise constant elements for the magnetic field. Electric field degrees of freedom are the tangential components of the electrical field along the triangle sides at each vertex.*

$$-i\omega \boldsymbol{u} - \vec{\nabla} \times v = 0 \quad \text{in } K, \tag{15a}$$

$$-i\omega v - \nabla \times \boldsymbol{u} = 0 \quad \text{in } K, \tag{15b}$$

$$u_1(x, y+h) = u_1(x, y)e^{ik_2 h} \quad \text{in } K, \tag{15c}$$

$$u_2(x+h, y) = u_2(x, y)e^{ik_1 h} \quad \text{in } K. \tag{15d}$$

Equations (15a) and (15b) are just the time-harmonic Maxwell equations corresponding to (1) and can be derived by assuming that $\boldsymbol{E} = \exp(-i\omega t)\boldsymbol{u}$ and $H = \exp(-i\omega t)v$. The second two equations (15c) and (15d) ensure that the fields behave like a plane wave under translations of length h in the x or y direction.

By taking the scalar curl of (15a) and using (15b) we find that v satisfies the Helmholtz equation

$$\Delta v + \omega^2 v = 0 \quad \text{in } K. \tag{16}$$

The quasi-periodicity conditions (15c) and (15d) can be rewritten in terms of derivatives of v by using the two components of (15a). After cancellation of factors in i and ω, we obtain

$$v_y(x, y+h) = v_y(x, y)\exp(ik_2 h), \tag{17a}$$

$$v_x(x+h, y) = v_x(x, y)\exp(ik_1 h). \tag{17b}$$

This system can be solved by separation of variables in the usual way. We find that in order for (17) to have a nontrivial solution it is necessary that $\omega = \omega_{m,n}$, where m and n are integers and

$$\omega_{m,n}^2 = k_1^2 + k_2^2 + \frac{2nk_1\pi}{h} + \frac{2mk_2\pi}{h} + \left(\frac{2n\pi}{h}\right)^2 + \left(\frac{2m\pi}{h}\right)^2.$$

Note that if $n = 0$ and $m = 0$, we recover the expected solution $\omega^2 = k_1^2 + k_2^2$, which corresponds to the exact solution of the dispersion relation discussed in the Introduction. However, if m or n are positive we obtain values of ω that are larger than $|\boldsymbol{k}|$ and hence the phase velocity of the solution, which is just $\omega/|\boldsymbol{k}|$, is larger than the speed of light. This is a standard observation for wave-guides (equations (17) are an example of an unusual wave-guide!) [17].

If we select $m = 0$ and $n = 1$ we obtain

$$\omega_{0,1}^2 = k_1^2 + k_2^2 + \frac{2k_1\pi}{h} + \left(\frac{2\pi}{h}\right)^2.$$

Thus as a function of h the leading term in the series is

$$\omega_{0,1}^2 = \frac{4\pi^2}{h^2} + O\left(\frac{1}{h}\right).$$

Furthermore, $4\pi^2 \approx 39.5$. Observe that the leading coefficient in equation (12) is 36. Given that we have approximated the eigen-problem (17) with only two triangles our computed value is remarkably close to the actual value. It is not clear why we do not pick up the first-order terms (in k_1) in the expansion (12) but maybe a manifestation of the use of a very coarse grid.

The above discussion suggests that the results of the numerical dispersion computations carried out in the previous section should be more properly compared to a wave-guide computation than to the actual dispersion computation for a continuous problem. In this light the "parasitic modes" observed are nothing more than approximations to higher-order wave guide modes (or Bloch modes). They are in fact inherent in Maxwell's equations and should not be expected to reveal themselves as nonphysical waves in a real computation.

5. Time stepping. To obtain a practical method for solving Maxwell's equations, the semidiscrete equations (5) must be discretized in time. We shall consider only the standard leapfrog technique, which has been used with great success to time-step finite difference schemes for Maxwell's equations [21]. The electric field is discretized at times $t_n = n\Delta t$, while the magnetic field is discretized at the half time steps $t_{n+1/2} = (n + 1/2)\Delta t$ where $\Delta t > 0$ is the time step. If E_i^n denotes an approximation to $E_i(t_n)$ and correspondingly $H_j^{n+1/2}$ denotes an approximation to $H_j(t_{n+1/2})$, then the fully discrete time-domain finite element procedure is

$$\text{(18a)} \qquad \sum_{i=1}^{\infty}\left(\frac{E_i^{n+1} - E_i^n}{\Delta t}\right)(\phi_i, \phi_k) - \sum_{j=1}^{\infty} H_j^{n+1/2}(\psi_j, \nabla \times \phi_k) = 0 \qquad \forall k \geq 1,$$

$$\text{(18b)} \qquad \sum_{j=1}^{\infty}\left(\frac{H_j^{n+3/2} - H_j^{n+1/2}}{\Delta t}\right)(\psi_j, \psi_l) + \sum_{i=1}^{\infty} E_i^{n+1}(\nabla \times \phi_i, \psi_l) = 0 \qquad \forall l \geq 1.$$

A dispersion analysis of the fully discrete scheme shows that, regardless of the finite element used, if we denote by $\omega_{\Delta t}$ the dispersion relation for (18) and by ω_0 the dispersion relation for (5), then

$$\omega_{\Delta t} h = \frac{2}{\lambda} \sin^{-1}\left(\frac{h\lambda\omega_0}{2}\right),$$

where $\lambda = \Delta t/h$. Expanding $\omega_{\Delta t}$ as a sequence in λ and $\boldsymbol{\xi}$ and assuming that

$$\omega_0^2 = k_1^2 + k_2^2 + (\alpha k_1^4 + \beta k_1^3 k_2 + \gamma k_1^2 k_2^2 + \delta k_1 k_2^3 + \mu k_2^4)h^2 + \text{ higher-order terms}$$

(where α, β, γ, δ and μ are constants) we find that

$$(\omega_{\Delta t} h)^2 = (\omega_0 h)^2 + \frac{h^2}{12}(k_1^2 + k_2^2)^2 \Delta t^2 + \text{ higher-order terms.}$$

Thus the leapfrog time step just contributes an extra isotropic second-order term in Δt. The necessary condition for stability is that

$$\lambda \left(\max_{(\xi_1, \xi_2)} |h\omega_0(\xi_1, \xi_2)| \right) \leq 2$$

(recall that with our definitions, $(\omega_0 h)$ is a function of $\boldsymbol{\xi}$ alone (independent of h)). From the above inequality, we see that a necessary condition for stability is $\Delta t \leq Ch$ for some constant C depending on the finite element method and mesh. Unfortunately, even with and exact expression for ω_0 it is difficult to derive a sharp value for the constant C, and so we will not report the constants here.

For the linear-linear element family, which has a fourth-order dispersion relation, we obtain that the fully discrete dispersion relation has the form

$$\omega_{\Delta t} = |\boldsymbol{k}|(1 + O((\Delta t)^2 + h^4))$$

and in order to make the error terms of the same order, we must take $\Delta t = O(h^2)$. This implies an extremely small time step. If we take the time step $\Delta t = Ch$, we see that essentially all the error in the fully discrete case is due to time-stepping error, and that we have reduced the order of accuracy of the scheme to second order. It is clear that a more complex time discretization strategy (maybe such as the Taylor–Galerkin method [4]) is required if the linear-linear element family is to be used to its full potential.

All the remaining methods (with the exception of edge elements on equilateral triangles) have a second-order accurate dispersion relation, and so the leapfrog method is quite appropriate as a time stepping scheme in these cases. The leapfrog time-stepping scheme is particularly attractive for those schemes in which $\omega^2 \leq |\boldsymbol{k}|^2$ since in that case effects of time-stepping error (increasing the numerical ω) and spatial error (decreasing the numerical ω) will cancel to some extent.

6. Mass lumping. It is clear from (18) that if the matrices multiplying the time difference terms are not diagonal we must invert a matrix at each time step in order to compute the solution. This is time consuming and so the numerical scheme is often modified by a more or less ad hoc procedure ("mass-lumping") to obtain a diagonal matrix and so a truly explicit scheme. We shall confine our comments on mass-lumping to the linear-linear and edge(1)-constant element families. Furthermore, we shall consider only the semidiscrete problem.

Mass lumping of the linear-scheme is quite standard. We simply approximate the inner products (ϕ_i, ϕ_j) and (ψ_j, ψ_l) in (18) using the following quadrature rule on each triangle K

$$\int_K f(\boldsymbol{x})\, d\boldsymbol{x} \approx \frac{\text{area}(K)}{3} \left(f(\boldsymbol{a}_1) + f(\boldsymbol{a}_2) + f(\boldsymbol{a}_3) \right),$$

where $\boldsymbol{a}_i$ is the coordinate of the ith vertex of K. Using this quadrature scheme element by element, we conclude that the mass lumped linear-linear scheme is

$$2h\frac{d}{dt}H_1 + \frac{1}{3}\left(\left(\frac{2a-1}{b}\right)E_{21} + 2E_{22} + \left(\frac{a-2}{b}\right)E_{31} + E_{32} - \left(\frac{a+1}{b}\right)E_{41} - E_{42}\right.$$

$$\left. - \left(\frac{2a-1}{b}\right)E_{51} - 2E_{52} - \left(\frac{a-2}{b}\right)E_{61} - E_{62} + \left(\frac{a+1}{b}\right)E_{71} + E_{72}\right) = 0, \tag{19a}$$

$$2h\frac{d}{dt}E_{11} - \frac{1}{3}\left(\left(\frac{1-2a}{b}\right)H_2 + \left(\frac{2-a}{b}\right)H_3 + \left(\frac{a+1}{b}\right)H_4 + \left(\frac{2a-1}{b}\right)H_5\right.$$

$$\left. + \left(\frac{a-2}{b}\right)H_6 - \left(\frac{a+1}{b}\right)H_7\right) = 0, \tag{19b}$$

$$2h\frac{d}{dt}E_{12} - \frac{1}{3}(-2H_2 - H_3 + H_4 + 2H_5 + H_6 - H_7) = 0. \tag{19c}$$

Again we can compute the dispersion relation for this scheme. The fourth-order convergence of the fully consistent lumped scheme is lost (this is easy to see in one dimension). For equilateral triangles, the dispersion relation becomes

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_1^4}{4} - \frac{\xi_1^2\xi_2^2}{2} - \frac{\xi_2^4}{4} + \text{ higher-order terms},$$

while for the right-triangular grid we find that

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_2^4}{3} + \frac{\xi_1\xi_2^3}{3} - \frac{2\xi_1^2\xi_2^2}{3} + \frac{\xi_1^3\xi_2}{3} - \frac{\xi_1^4}{3} + \text{ higher-order terms}.$$

The situation for the edge-constant scheme is more complex. The magnetic field equations are mass-lumped without modification since we use piecewise constant functions to discretize the magnetic field. It turns out that simple mass lumping of the electric field equations cannot be carried out on an arbitrary grid. In order to mass lump the electric field equations, we can again try to use a suitable quadrature scheme to approximate the inner product (ϕ_i, ϕ_j). To obtain a lumped system, we need to use the degrees of freedom as quadrature points. In the case of edge elements, these degrees are the tangential components of the electric field at the midpoints of the edges so that the quadrature scheme is nonstandard. By requiring that the quadrature be accurate for all pairs of constant vectors on a given triangle K we find that we must use the quadrature

$$\int_K \boldsymbol{A}(\boldsymbol{x}) \cdot \boldsymbol{B}(\boldsymbol{x})\, d\boldsymbol{x} \approx \frac{\text{area}(K)}{b^2}\left(\left[a(a-1) + b^2\right]\boldsymbol{A}(\boldsymbol{a}_{1,2}) \cdot \tau_{1,2}\boldsymbol{B}(\boldsymbol{a}_{1,2}) \cdot \tau_{1,2}\right.$$
$$+ \left[a((a-1)^2 + b^2)\right]\boldsymbol{A}(\boldsymbol{a}_{2,3}) \cdot \tau_{2,3}\boldsymbol{B}(\boldsymbol{a}_{2,3}) \cdot \tau_{2,3}$$
$$\left. + \left[(1-a)(a^2 + b^2)\right]\boldsymbol{A}(\boldsymbol{a}_{1,3}) \cdot \tau_{1,3}\boldsymbol{B}(\boldsymbol{a}_{1,3}) \cdot \tau_{1,3}\right).$$

Using this quadrature, and taking into account that the degrees of freedom for the edge elements contain an edge length factor, we obtain the following mass-lumped equations for the edge-constant element family:

$$I_{11}^{lm}\frac{dE_1}{dt} - \frac{1}{h}(H_1 - H_2) = 0, \tag{20a}$$

$$I_{22}^{lm}\frac{dE_2}{dt} + \frac{1}{h}(H_1 - H_3) = 0, \tag{20b}$$

$$I_{33}^{lm}\frac{dE_3}{dt} - \frac{1}{h}(H_1 - H_4) = 0, \tag{20c}$$

$$\frac{dH_1}{dt} + \frac{2}{bh}(E_1 - E_2 + E_3) = 0, \tag{20d}$$

$$\frac{dH_2}{dt} + \frac{2}{bh}(E_6 - E_1 - E_7) = 0, \tag{20e}$$

where

$$I_{11}^{lm} = a/b, \qquad I_{22}^{lm} = (1-a)/b \quad \text{and} \quad I_{33}^{lm} = (a(a-1)+b^2)/b.$$

We remark that the equations (20) are exactly the equations for the covolume method [16], [8] applied to (1) on the given grid, so at least for the given uniform grid, the covolume and mass-lumped edge element methods are identical. Of course, the nonlumped finite element method has the advantage of higher-order convergence of the dispersion relation on some grids, and the possibility of handling nonacute triangles (or more general grids). This is at the cost of inverting a matrix at every time step.

Applying (20) in the case of the equilateral triangulation we obtain the following dispersion relation

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{\xi_2^4}{48} - \frac{\xi_1^2\xi_2^2}{24} - \frac{\xi_1^4}{48} + \text{higher-order terms.}$$

Again, by lumping, we have reduced the order of approximation in the dispersion relation. An even nastier surprise awaits us if we try to lump in the case of the right-triangle grid. In that case $I_{11}^{lm} = 0$. Of course, if the triangles are not acute (if $a < 0$ or $a > 1$), the quadrature mass lumping technique gives negative values for some of the coefficients of the time derivatives. In this case, the time-stepping scheme may become unstable.

Proceeding in the case of the right-angle triangle grid, if we mass lump and accept the fact that, since $I_{11}^{lm} = 0$, certain time derivatives do not appear in the lumped equations, we can still compute a dispersion relation. Remarkably, there are only three eigenvalues corresponding to the three physical eigenvalues (the parasitic modes vanish). We find that

$$(\omega h)^2 = \xi_1^2 + \xi_2^2 - \frac{1}{12}\left(\xi_1^4 + \xi_2^4\right) + \text{higher-order terms}$$

(to compute this solution, we have used the generalised eigenvalue and Taylor series routines in MAPLE). This is exactly the same dispersion relation (to the order shown) as for the Yee scheme on the corresponding quadrilateral mesh. The mass lumping has simply forced us to use a more appropriate element for the given grid.

Clearly, the fact that the mass-lumping scheme on right-angle triangles results in a coefficient of zero for some of the time derivatives would complicate the programming of the fully discrete Maxwell system. Thus it is reasonable to ask if a different mass-lumping strategy would produce a more tractable system (i.e., positive weights for the time derivatives). If we allow for general coefficients of the electric field time derivatives in (20) and consider only right-angle triangles (so $a = 0$ and $b = 1$), we have the system (see Fig. 6)

$$\alpha_{11}\frac{dE_1}{dt} - \frac{1}{h}(H_1 - H_2) = 0, \tag{21a}$$

$$\alpha_{22}\frac{dE_2}{dt} + \frac{1}{h}(H_1 - H_3) = 0, \tag{21b}$$

$$\alpha_{22}\frac{dE_3}{dt} - \frac{1}{h}(H_1 - H_4) = 0, \tag{21c}$$

$$\frac{dH_1}{dt} + \frac{2}{h}(E_1 - E_2 + E_3) = 0, \tag{21d}$$

$$\frac{dH_2}{dt} + \frac{2}{h}(E_6 - E_1 - E_7) = 0. \tag{21e}$$

Here we have set the coefficients of dE_2/dt and dE_3/dt to be the same on the grounds of symmetry. Using a dispersion analysis, we obtain that for the physically relevant dispersion relation

$$(\omega h)^2 = \frac{(\alpha_{11} + \alpha_{22})(\xi_1^2 + \xi_2^2) - 2\alpha_{11}\xi_1\xi_2}{\alpha_{22}(\alpha_{22} + 2\alpha_{11})} + \text{higher-order terms.}$$

To make the dispersion relation second order in h, we are forced to choose $\alpha_{11} = O(h^2)$ and

$$\frac{\alpha_{11} + \alpha_{22}}{\alpha_{22}(\alpha_{22} + 2\alpha_{11})} = 1 + O(h^2).$$

Thus to first order in h, the only choice for α_{11} and α_{22} is $\alpha_{11} = 0$ and $\alpha_{22} = 1$ as predicted by our derivation using the quadrature method. It appears that it is possible to perform simple mass-lumping for the edge-constant family only if the triangles are all acute.

7. Conclusion. On the basis of a dispersion analysis, it is clear that some combinations of finite element spaces are not very attractive (linear-constant, edge(1)-edge(1), edge(2)-constant), and some methods need more analysis before becoming truly reliable (nonconforming-constant). Two methods, the linear-linear scheme and the edge(1)-constant scheme have significant advantages and disadvantages. The linear-linear scheme has excellent dispersion properties on both meshes studied here and can be mass-lumped (at the cost of a decrease in accuracy). However, it is complicated to deal with boundary conditions, and the divergence conditions on the field are only approximated. On the other hand, the edge(1)-constant method is more sensitive to the mesh showing fourth-order convergence on the equilateral mesh, but only second order on the right-angled mesh. Furthermore, the method has parasitic modes. Mass-lumping of the edge-constant scheme is possible provided the mesh contains only strictly acute triangles. On the other hand, the method deals with boundary conditions and divergence conditions in a natural way. It has been suggested to use edge elements (second type edge elements) at interfaces and standard linear elements away from interfaces [13], [12], and this type of method may indeed be a good way of combining the strengths of both element families.

The results in this paper are applicable in three dimensions if prismatic elements are used to discretize Maxwell's equations. The more interesting case of tetrahedral elements is yet to be analyzed.

REFERENCES

[1] J. J. AMBROSIANO, S. T. BRANDOM, AND R. LOHNER, *A New Weighted Residual Finite Element Method for Computational Electromagnetics in the Time Domain*, Tech. report UCRL-JC-106408, Lawrence Livermore National Laboratory, Livermore, CA, 1991.

[2] A. BOSSAVIT, *A rationale for "edge elements" in* 3-D *fields computations*, IEEE Trans. Magnetics, 24 (1988), pp. 74–79.

[3] ———, *Solving Maxwell equations in a closed cavity, and the question of 'spurious' modes*, IEEE Trans. Magnetics, 26 (1990), pp. 702–705.

[4] J. DONEA, *Taylor-Galerkin method for convective transport problems*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 101–120.

[5] R. LEE AND N. MADSEN, *A mixed finite element formulation for Maxwell's equations in the time domain*, J. Comput. Phys., 88 (1990), pp. 284–304.

[6] V. LEVILLAIN, *Eigenvalue appoximation by mixed methods for resonant inhomogenous cavities with metallic boundaries*, Math. Comp., 58 (1992), pp. 11–20.

[7] N. MADSEN AND R. ZIOLKOWSKI, *Numerical solution of Maxwell's equations in the time domain using irregular nonorthogonal grids*, Wave Motion, 10 (1988), pp. 583–596.

[8] B. MCCARTIN AND J. F. DICELLO, *Three dimensional finite difference frequency domain scattering computation using the control region approximation*, IEEE Trans. Magnetics, 25 (1989), pp. 3092–3094.

[9] P. MONK, *A finite element method for approximating the time-harmonic Maxwell equations*, Numer. Math., 63 (1992), pp. 243–261.

[10] ———, *An analysis of Nédélec's method for the spatial discretization of Maxwell's equations*, J. Comput. Appl. Math., 47 (1993), pp. 101–121.

[11] K. MORTON AND A. PARROTT, *Generalized Galerkin methods for first-order hyperbolic equations*, J. Comput. Phys., 36 (1980), pp. 249–270.

[12] G. MUR, *The finite-element modeling of three-dimensional time-domain electromagnetic fields in strongly inhomogeneous media*, IEEE Trans. Magnetics, 28 (1992), pp. 1130–1133.

[13] G. MUR AND A. DE HOOP, *A finite-element method for computing three-dimensional electromagnetic fields in inhomogeneous media*, IEEE Trans. Magnetics, MAG-21 (1985), pp. 2188–2191.

[14] J. NÉDÉLEC, *Mixed finite elements in* $\mathbf{R}^3$, Numer. Math., 35 (1980), pp. 315–341.

[15] ———, *A new family of mixed finite elements in* $\mathbf{R}^3$, Numer. Math., 50 (1986), pp. 57–81.

[16] R. NICOLAIDES, *Direct discretization of planar div-curl problems*, SIAM J. Numer. Anal., 29 (1992), pp. 32–56.

[17] M. SCHWARTZ, *Principles of Electrodynamics*, Dover, New York, 1972.

[18] F. THOMASSET, *Implementation of finite element methods for Navier–Stokes equations*, Springer Series in Computational Physics, Springer-Verlag, New York, 1981.

[19] V. THOMÉE AND B. WENDROFF, *Convergence estimates for Galerkin methods for variable coefficient initial value problems*, SIAM J. Numer. Anal., 11 (1974), pp. 1059–1068.

[20] L. TREFETHEN, *Group velocity in finite difference schemes*, SIAM Rev., 24 (1982), pp. 113–136.

[21] K. YEE, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antennas and Propagation, AP-16 (1966), pp. 302–307.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 938–952, July 1994

009

COLLOCATION SOFTWARE FOR BOUNDARY VALUE DIFFERENTIAL-ALGEBRAIC EQUATIONS*

URI M. ASCHER† AND RAYMOND J. SPITERI‡

Abstract. The methods and implementation of a general-purpose code, COLDAE, are described. This code can solve boundary value problems for nonlinear systems of semi-explicit differential-algebraic equations (DAEs) of index at most 2. Fully implicit index-1 boundary value DAE problems can be handled as well.

The code COLDAE is an extension of the package COLNEW (COLSYS) for solving boundary value ODEs. The implemented method is piecewise polynomial collocation at Gaussian points, extended as needed by the projection method of Ascher–Petzold. For general semi-explicit index-2 problems, as well as for fully implicit index-1 problems, a *selective projected collocation* method is defined and its use is demonstrated. The mesh selection procedure of COLSYS is modified for the case of index-2 constraints. Also discussed is shooting for initial guesses.

The power and generality of the code are demonstrated by examples. COLDAE can be obtained from the electronic library `netlib`.

Key words. differential-algebraic equations, boundary value problems, collocation, projection, software

AMS subject classifications. 65L10, 65L20

1. Introduction.

Many mathematical models arising in various applications can be written as systems of DAEs; see, e.g., [16], [22]–[24]. The numerical solution of initial value DAEs has received much attention recently, and robust general-purpose codes have been written and extensively used [16], [24]. Similar developments for boundary value DAEs have been considerably lagging behind, although such problems arise in a variety of applications; see, e.g., [2], [13]–[15], [18], [25], [26], [28].

In this paper we describe the methods and implementation of a general-purpose code, COLDAE, which can handle boundary value problems for nonlinear systems of semi-explicit DAEs of index at most 2 and of fully implicit index-1 DAEs. The code COLDAE extends the package COLNEW [10], using the projected collocation method of [7] and extensions thereof. To our knowledge, there is currently no other general-purpose code that handles such a class of problems. COLDAE can be obtained from the electronic library `netlib`.

The package COLNEW [10] is a modification of the package COLSYS [4]. COLNEW is a robust code which has been used by many scientists and engineers to solve practical problems that can be formulated as systems of ordinary differential equations (ODEs) with boundary or interface conditions. The ODE system which COLNEW handles has a generally nonlinear, mixed order form

$$\mathcal{D}^{m_i} u_i = f_i(t, \mathbf{z}(\mathbf{u})), \qquad i = 1, \ldots, d, \tag{1.1}$$

where t is the independent variable, $a \le t \le b$, $\mathcal{D}$ denotes differentiation with respect to t, so $\mathcal{D}^j \equiv d^j/dt^j$, $\mathbf{u} = (u_1, \ldots, u_d)^T$ are the dependent variables, and for each t

$$\mathbf{z}(\mathbf{u}) = (u_1, \mathcal{D}u_1, \ldots, \mathcal{D}^{m_1-1}u_1, u_2, \ldots, u_d, \mathcal{D}u_d, \ldots, \mathcal{D}^{m_d-1}u_d)^T.$$

In $\mathbf{z}(\mathbf{u})(t)$ there are $m^* = \sum_{i=1}^{d} m_i$ *differential solution components*. The system (1.1) is subject to m^* side (boundary) conditions, which are each given as a nonlinear relationship in

*Received by the editors December 22, 1992; accepted for publication (in revised form) May 6, 1993.

†Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada V6T 1Z2 (ascher@cs.ubc.ca). The work of this author was partially supported under Natural Sciences and Engineering Research Council of Canada grant OGP0004306.

‡Department of Mathematics, University of British Columbia, Vancouver, British Columbia, Canada V6T 1Z2 (spiteri@math.ubc.ca). The work of this author was partially supported by a Natural Sciences and Engineering Research Council of Canada Centennial Postgraduate Fellowship.

$\mathbf{z}(\mathbf{u})$ at one point,

$$b_j(\beta_j, \mathbf{z}(\mathbf{u})(\beta_j)) = 0, \qquad j = 1, \ldots, m^*, \tag{1.2}$$

where $a \le \beta_1 \le \beta_2 \le \cdots \le \beta_{m^*} \le b$. Thus, (1.2) includes the usual initial value and two-point boundary value problems with separate boundary conditions as special cases.

The underlying numerical method used in COLNEW (COLSYS) is piecewise polynomial collocation at Gaussian points. Thus, with k collocation points at each subinterval of a given mesh,

$$\pi : a = t_0 < t_1 < \cdots < t_N = b, \tag{1.3}$$

the approximate solution for $u_i(t)$ is sought as a piecewise polynomial of order $k + m_i$ (degree $< k + m_i$) on each mesh subinterval $[t_{n-1}, t_n]$ that is globally continuous together with its first $m_i - 1$ derivatives, viz., $u_i^\pi \in \mathcal{P}_{k+m_i,\pi} \cap C^{(m_i-1)}[a, b]$. Then $\mathbf{z}^\pi = \mathbf{z}(\mathbf{u}^\pi)$ is determined[1] by requiring that the ODE (1.1) be satisfied at Nk collocation points and that the m^* boundary conditions (1.2) be satisfied as well. We refer, e.g., to [1], [6], [10], [12] for the theory justifying this method. Loosely speaking, the highlights of this theory are that, assuming sufficient smoothness and convergence of a Newton method to an isolated solution of the problem (1.1), (1.2), the following statements hold.

1. The discrete system for the approximate solution has a stability constant which is close to that of the differential problem for h sufficiently small. Here, $h = \max h_n$, where $h_n = t_n - t_{n-1}$, $n = 1, \ldots, N$.

2. At mesh points there is superconvergence,

$$|z_i(t_n) - z_i^\pi(t_n)| = O(h^{2k}), \; i = 1, \ldots, m^*, \qquad n = 0, \ldots, N. \tag{1.4}$$

3. At points other than mesh points, the convergence order is lower, but local, as follows:

$$\begin{aligned} |\mathcal{D}^j u_i(t) - \mathcal{D}^j u_i^\pi(t)| &= [c_i |\mathcal{D}^{k+m_i} u_i(t_n)| + O(h_n)] h_n^{k+m_i-j} + O(h^{2k}), \\ i &= 1, \ldots, d, \qquad j = 0, \ldots, m_i, \end{aligned} \tag{1.5}$$

where c_i are known constants and $t_{n-1} \le t \le t_n$.

These results hold for problems that are not very stiff and form the theoretical foundation for the mesh selection and error estimation procedures in COLSYS(COLNEW). In particular, the local nature of the leading error term in (1.5) when $k > \max_i m_i$ is exploited.

In COLDAE, we extend the COLSYS class of problems by considering the case where the ODEs in (1.1) also involve m additional dependent variables $\mathbf{y}(t)$ (referred to as *algebraic solution components*) and are supplemented by m additional algebraic relations (constraints). Thus, we have, in place of (1.1),

$$\mathcal{D}^{m_i} u_i = f_i(t, \mathbf{z}(\mathbf{u}), \mathbf{y}), \qquad i = 1, \ldots, d, \tag{1.6a}$$

$$0 = f_i(t, \mathbf{z}(\mathbf{u}), \mathbf{y}), \qquad i = d + 1, \ldots, d + m. \tag{1.6b}$$

This is a semi-explicit DAE. The reasons that we have opted not to directly implement the fully implicit case are that while many applications, especially of higher index, are in the semi-explicit form, the theory, both for the analytic problem and for numerical approximations, is much less solid in the more general case (see [2], [20], [22]). Even in the index-1 fully implicit

[1]We employ the following notational convention: a superscript denotes power, unless it is a Greek letter (in which case it denotes the approximate solution).

case, none of the above three stability and convergence results holds for a straightforward collocation approximation at Gaussian points (although there is often basic convergence [2]). We handle fully implicit index-1 problems by imbedding them in semi-explicit index-2 ones; this is discussed in §4.

For the semi-explicit DAE (1.6), we can do much better by collocating the differential solution components as before, while collocating the algebraic solution components $\mathbf{y}(t)$ using a generally discontinuous piecewise polynomial of order k: $\mathbf{y}^\pi \in \mathcal{P}_{k,\pi}$. This corresponds to treating the constraints (1.6b) like ODEs of order 0, an obvious idea that was conceived by many and implemented a while ago [25]. However, the code of [25] does not extend to the more challenging classes of DAEs considered in this paper.

If the DAE has index 1, i.e., if the matrix E is nonsingular, where $E = (e_{ij})$,

$$e_{ij} = \frac{\partial f_{d+i}}{\partial y_j}, \qquad i, j = 1, \ldots, m, \tag{1.7}$$

then $\mathbf{y}$ can be eliminated in principle, using (1.6b), and substituted into (1.6a) to form an ODE system. Furthermore, a collocation approximation of (1.6b) (i.e., collocating both (1.6a) and (1.6b) at the same Gaussian points) can then be viewed as the usual collocation approximation for this obtained ODE with $\mathbf{y}$ eliminated, and the above cited convergence results are recovered (cf. [2], [22]).

But if E is singular (i.e., the DAE has a higher index), then merely approximating $\mathbf{y}$ in the "natural space" $\mathcal{P}_{k,\pi}$ does not produce a collocation method for which the ODE results hold. The simplest way to see this is to note that by writing $\mathcal{D}\mathbf{w} \equiv \mathbf{y}$, we obtain for $(\mathbf{z}(\mathbf{u}), \mathbf{w})$ at best a fully implicit index-1 DAE [19].

A projection method, which achieves the desired stability and convergence behaviour, was therefore proposed in [7] and [9]. This method applies to a pure index-2 DAE[2], i.e., for the case where $E \equiv 0$ and CB is nonsingular wherever it is evaluated, where $C = (c_{ij})$, $B = (b_{ij})$,

$$c_{ij} = \frac{\partial f_{d+i}}{\partial z_{p_j}}, \qquad i = 1, \ldots, m, \; p_j = \sum_{l=1}^{j} m_l, \; j = 1, \ldots, d, \tag{1.8}$$

$$b_{ij} = \frac{\partial f_i}{\partial y_j}, \qquad i = 1, \ldots, d, \; j = 1, \ldots, m. \tag{1.9}$$

With this method, at the right end of each mesh subinterval, following a collocation step as described above (applied within a quasilinearization step), we modify the $(m_i - 1)$st derivatives of $\mathbf{u}$ by a vector from the range space of B so as to satisfy the constraints (1.6b) at the right-end mesh point.

We restrict our implementation to DAEs of index at most 2 (this includes ODEs, which are DAEs with index 0). For reasons of inherent problem stability and (related) lack of reliable direct discretization methods, we require that higher index problems be (stably!) converted to lower index ones by analytic differentiation (cf. [8]) prior to applying the code.

Many practical higher index problems are formulated in a pure index-2 form. But still many others have a mix of index-1 and index-2 constraints (i.e., E is singular but not 0). The rank of E may well depend on the iterate where this Jacobian matrix is evaluated and it may vary depending on t on the interval $[a, b]$ (e.g., the index may change on a singular

[2]A pure index-2 DAE is often referred to as being in Hessenberg form. We will see that the term "pure" is rather natural.

arc in optimal control problems). COLDAE has the option of handling the more general case at the price of a singular value decomposition (SVD) of E at each mesh point. This feature also allows handling fully implicit index-1 DAEs. That and other implementation details are described in §§2–4. Specifically, in §2 we briefly recall the piecewise polynomial collocation method [2], [12], and in §3 we recall projected collocation [7] and describe our mesh selection modification. In §3 we also explain the requirements that the user-specified multipoint side conditions (1.2) must satisfy. The problem of finding consistent boundary conditions can be nontrivial even in the initial value case. With the generality of COLDAE, we must require that the user supply m^* boundary conditions, even for an index-2 DAE. In the index-2 case only $m^* - m$ independent conditions are needed in theory, but we require that the given boundary conditions include specifying the m constraints (1.6b) at the left-end point a; see the examples in §5. In §4 we then treat general semi-explicit index-2 DAEs, and describe a *selective projected collocation* method. We also describe a simple trick for handling fully implicit index-1 problems. A number of illustrative examples are then presented in §5. We use these examples to also discuss further modifications to COLDAE that also improve the usage of COLNEW, for both DAEs and ODEs.

We end this section with a warning: like any boundary value ODE code, COLDAE will not be successful for all applications. Chances of success can improve significantly if the user takes care to provide information such as an appropriate initial iterate for difficult nonlinear problems (possibly using continuation), an appropriate initial mesh in some extreme cases, and appropriate boundary conditions (the latter being somewhat trickier here than in the ODE case).

2. DAE collocation. For a nonlinear boundary value DAE (1.6), (1.2), we implement a quasilinearization method with a damped Newton scheme, as described in [4], [6], and [10]. Thus we may assume below, for purposes of the presentation, that the DAE problem is linear(ized).

Let $\rho_1 < \rho_2 < \cdots < \rho_k$ be the k canonical collocation points on [0, 1]. In COLDAE we use the zeros of the Legendre polynomial. (This yields a symmetric, algebraically stable difference scheme with $\rho_1 > 0$, $\rho_k < 1$.) The collocation points on a mesh subinterval $[t_{n-1}, t_n]$ are then

$$t_j = t_{n-1} + h_n \rho_j, \qquad j = 1, \ldots, k.$$

An (unprojected) collocation step requires that the DAE (1.6) be satisfied by the approximate solution at the collocation points, viz.,

$$\text{(2.1a)} \qquad \mathcal{D}^{m_i} u_i^\pi(t_j) = f_i(t_j, \mathbf{z}^\pi(t_j), \mathbf{y}^\pi(t_j)), \qquad i = 1, \ldots, d,$$

$$\text{(2.1b)} \qquad 0 = f_i(t_j, \mathbf{z}^\pi(t_j), \mathbf{y}^\pi(t_j)), \qquad i = d+1, \ldots, d+m$$

for $j = 1, \ldots, k$. We use a monomial representation for the piecewise polynomial approximate solution ($i = 1, \ldots, d$),

$$\text{(2.2)} \qquad u_i^\pi(t) = \sum_{l=0}^{m_i - 1} \frac{(t - t_{n-1})^l}{l!} z_{n-1, p+l} + (h_n)^{m_i} \sum_{l=1}^{k} \psi_l \left(\frac{t - t_{n-1}}{h_n} \right) w_{il},$$

where $z_{n-1,p+l} = z_{p+l}^\pi(t_{n-1})$, $p = \sum_{j=1}^{i-1} m_j + 1$, and ψ_l are k polynomials of order $k + m_i$ on [0, 1] satisfying

$$\text{(2.3)} \qquad \mathcal{D}^j \psi_l(0) = 0, \ j = 0, \ldots, m_i - 1, \qquad \mathcal{D}^{m_i} \psi_l(\rho_j) = \delta_{jl}, \ j, l = 1, \ldots, k.$$

It follows that $w_{il} = \mathcal{D}^{m_i} u_i^\pi(t_l)$.[3] A similar representation is used for the algebraic components $\mathbf{y}$ with $m_i = 0$ in the above formulae (hence $\mathbf{w}_l = \mathbf{y}^\pi(t_l) \equiv \mathbf{y}_l$).

Substituting this representation in the collocation equations (2.1), we obtain $(d+m)k$ equations, which we can use to locally eliminate the m_ith derivatives $\mathbf{w}_l$ and the algebraic components $\mathbf{y}_l$ in terms of the mesh values of $\mathbf{z}$,

$$\mathbf{z}_{n-1} = (z_1^\pi(t_{n-1}), \ldots, z_{m^*}^\pi(t_{n-1}))^T. \tag{2.4}$$

It is not difficult to see that for $h_n > 0$ small enough, the local linear system of order $(d+m)k$ which is solved in this process is nonsingular. The matrix involved has a bounded inverse in the case of a DAE of index 1. In the index-2 case, the inverse bound is $O(h_n^{-1})$, but at least for a pure index-2 form this merely corresponds to scaling of the collocation rows corresponding to the constraints. Thus, the approximate solution at any point t in $[t_{n-1}, t_n)$ can be obtained in terms of $\mathbf{z}_{n-1}$ using (2.2). The continuity requirements on $\mathbf{z}^\pi$ are then obtained by evaluating the approximate solution at t_n and equating it to $\mathbf{z}_n$:

$$\mathbf{z}_n = \Gamma_n \mathbf{z}_{n-1} + \gamma_n, \qquad n = 1, \ldots, N. \tag{2.5}$$

A complete linear system of $(N+1)m^*$ equations for $(N+1)m^*$ unknowns is now obtained upon requiring m^* side conditions to be satisfied by $\mathbf{z}^\pi$. The structure of this system, and hence the method for its solution, are the same as encountered with ODEs in COLNEW, and need not be elaborated further here (cf. [10]). We note, however, that our approach to DAEs in all cases is to eliminate the algebraic solution components, obtaining an ODE discretization. This allows us to obtain stability and affects our decisions about the class of problems solved by COLDAE and their error control and mesh selection: in the latter, only the differential solution components $\mathbf{z}$ are considered.

3. Projected collocation for pure index-2 DAEs. While the straightforward collocation method described above is feasible, and converges as $h \to 0$ for all well-conditioned linear problems of index at most 2 (except for a few pathological cases, cf. [2] and references therein), the properties of nonstiff ODE collocation discussed in §1 are retained only for index-1 problems. For a pure index-2 DAE (i.e., $E \equiv 0$, cf. (1.7)), we therefore apply projected collocation [7], [9]. For the mixed-order DAE (1.6), let

$$\mathbf{x} = (\mathcal{D}^{m_1-1}u_1, \mathcal{D}^{m_2-1}u_2, \ldots, \mathcal{D}^{m_d-1}u_d)^T, \tag{3.1}$$

and denote similarly by $\mathbf{x}^\pi$ and $\mathbf{x}_n$ the corresponding subvectors of $\mathbf{z}^\pi$ and $\mathbf{z}_n$. Following a collocation step as described in the previous section, we project

$$\mathbf{x}_n \leftarrow \mathbf{x}^\pi(t_n) + B(t_n)\lambda_n \tag{3.2}$$

with λ_n chosen so that the linearized (1.6b) be satisfied at t_n, $1 \le n \le N$.

The projection requirements allow one to eliminate λ_n locally, obtaining

$$\mathbf{x}_n \leftarrow (I - B(CB)^{-1}C)\mathbf{x}_n - B(CB)^{-1}\mathbf{r} \tag{3.3}$$

(see (1.8), (1.9)), where $\mathbf{r}$ is the corresponding inhomogeneity of the linearized constraints and all quantities are evaluated at t_n. This, in turn, is incorporated into (2.5), where rows $m_1, m_2, \ldots, m_d$ of Γ_n and the corresponding elements of γ_n get multiplied by $(I - B(CB)^{-1}C)(t_n)$ and the term $(B(CB)^{-1}\mathbf{r})(t_n)$ is subtracted from the projected γ_n as

[3] The functions ψ_l depend of course on m_i as well, but note that they need be constructed only once, not $d+m$ times: if $m_1 = \max_i m_i$, say, then ψ_l for all solution components are appropriate derivatives of those used for u_1^π.

well. Once this is done, the rest of the solution process is again the same as for boundary value ODEs.

The constraints (1.6b) are thus satisfied at all mesh points except t_0. In COLDAE we use the convention that satisfying the constraints (1.6b) at the left end point $t_0 = a$ is a requirement on the boundary conditions (1.2). Indeed, recall that for an index-2 DAE, only $m^* - m$ side conditions independent of the constraints are required to hold. To be able not to distinguish a priori between index-1 and index-2 problems, COLDAE expects m^* side conditions on $\mathbf{z}(\mathbf{u})$ in any case: for index-2 problems these include the specification of the constraints at the left end point.

With the above described projection, the superconvergence results (1.4) hold [7], [9].[4] Let us quickly recall that the proof is obtained by considering a $(d - m) \times d$ matrix function $R(t)$ with normalized, linearly independent rows, which satisfies for each t,

$$RB = 0. \tag{3.4}$$

Thus, multiplying (1.6a) by R eliminates $\mathbf{y}$, and an *essential underlying ODE* (EUODE) is obtained for

$$\mathbf{v}(t) = R\mathbf{u}. \tag{3.5}$$

The results (1.4), (1.5) are retrieved for $\mathbf{v}$ (this is simple to see, particularly when R is constant), and the projection on the constraint manifold at mesh points then helps to retrieve the superconvergence (1.4) for $\mathbf{u}$ as well.

The expression obtained in place of (1.5), however, is for $\mathbf{v}$, not $\mathbf{u}$. The first term on the right-hand side of (1.5) is equidistributed when selecting a new mesh, and this now relates to higher derivatives of $\mathbf{v}$. In other words, the error equidistribution should be done on the constraint manifold. In practice, we approximate R for this purpose by a piecewise constant function, and thus project the approximate solution $\mathbf{x}^\pi$ on each mesh subinterval $[t_{n-1}, t_n]$, multiplying it by the already computed $(I - B(CB)^{-1}C)(t_n)$. Divided differences are then used to estimate the higher derivatives of $\mathbf{v}$, as in COLSYS (see, e.g., [6, §9.3]). For nonlinear problems, this is done only after convergence of the nonlinear iteration has been achieved on a current mesh.

4. Selective projected collocation for the general case. In this section we consider the general case for (1.6) where the matrix $E(t)$ is possibly singular but not necessarily zero. We will in fact allow the rank of $E(t)$ to vary with t, except that we require a constant rank on each mesh subinterval in the numerical approximation. This means, in practical terms, that we allow for switching points $\hat{t}_l$ where the rank of E may change. These points must become part of any mesh in COLDAE, so their location should be known in advance; however, there are standard tricks (see, e.g., [6]) to convert a problem with unknown switching points to one with known switching points, provided we know how many such points there are.

Given a matrix $E(t)$ (using (1.7) for the linearized DAE (1.6)), consider its pointwise SVD:

$$E = U\Sigma V^T, \tag{4.1}$$

where U and V are orthogonal matrices and

$$\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} \tag{4.2}$$

[4]Note that now the global continuity has been lowered: $u_i^\pi \in \mathcal{P}_{k+m_i,\pi} \cap C^{(m_i-2)}[a, b]$, with continuity from the right in $\mathbf{x}^\pi$ at mesh points.

with S a nonsingular diagonal matrix of rank r. (In general, $r = r(t, \mathbf{z}(\mathbf{u})(t), \mathbf{y}(t))$.) Let

$$U = (U_1\, U_2), \qquad V = (V_1\, V_2), \tag{4.3}$$

where U_1 and V_1 consist of the first r columns of U and V, respectively, and U_2 and V_2 are the rest. Writing

$$\mathbf{y} = V\psi = V_1\psi_1 + V_2\psi_2, \tag{4.4}$$

we have

$$E\mathbf{y} = U_1 S\psi_1, \tag{4.5}$$

so ψ_1 can be eliminated from the constraints. This is the "index-1 part" of the algebraic unknown vector $\mathbf{y}$. Moreover, clearly $U_2^T E = 0$, so upon multiplying the linear (1.6b) by U_2^T and substituting $\psi = V^T\mathbf{y}$ in (1.6a) we obtain a DAE in pure index-2 form for $\mathbf{u}$ and ψ_2, the latter being the "pure index-2 part" of the algebraic unknown vector $\mathbf{y}$. In this transformed DAE of pure form, the condition for index-2 is clearly that $U_2^T C B V_2$ be nonsingular.

Consider next applying a collocation scheme without projection, as described in §2, to the linear (1.6). The transformations just described are all pointwise and involve no derivatives, so the collocation equations for the transformed DAE are identical to the transformed collocation equations applied to the original form.

The only difference arises for the projected collocation method, where the projection should be carried out only onto the constraint manifold corresponding to the pure index-2 variables ψ_2. But this is conceptually simple now: the procedure described in §3 applies here with no change, except that

$$B \leftarrow BV_2, \qquad C \leftarrow U_2^T C. \tag{4.6}$$

The resulting method is referred to as selective projected collocation.

In COLDAE, the user has the following three options.

1. *Collocating with no projection.* This is good for ODEs and for semi-explicit index-1 boundary value problems. For index-2 problems and for fully implicit index-1 problems, one of the next two options is usually preferable.

2. *Projected collocation for pure index-2.* This is the preferred option if the user knows that the problem to be solved is indeed of this form (everywhere in t).

3. *Selective projected collocation.* This "rich man's option" corresponds to the case described in the present section. At the right end of each subinterval (within a quasilinearization iteration) the code uses LINPACK routines to decompose E as in (4.1). The unprojected collocation procedure is as usual, except that at the end of each subinterval processing, a projection step is performed as in §3, but using (4.6). These updated matrices are also used for the mesh selection procedure.[5]

Fully implicit index-1 DAEs. The mixed index-2 option discussed in this section also allows dealing with fully implicit index-1 problems. Consider the system of m equations

$$\mathbf{g}(t, \mathbf{x}, \mathcal{D}\mathbf{x}) = \mathbf{0} \tag{4.7}$$

(we suppress higher derivatives for notational simplicity), and denote

$$\mathbf{y} = \mathcal{D}\mathbf{x}, \quad E = \mathbf{g}_{\mathbf{y}}, \quad C = \mathbf{g}_{\mathbf{x}}. \tag{4.8}$$

[5]The expense of SVD computations is not very siginificant for small problems. Note that the decompositions are carried out only on $m \times m$ matrices. To deal with larger problems more efficiently, we are considering smooth factorizations [21] and [27]. This will be reported on in the near future.

Applying SVD to E of (4.8) as before and using the notation (4.1)–(4.5), it follows that near a given solution (4.7) has index 0 if the rank is $r \equiv m$ and index 1 if $r < m$, r constant, and $U_2^T C V_2$ is nonsingular (see, e.g., [19] and [23]). This allows us to pose the index-1 (4.7) as the semi-explicit DAE of index at most 2:

$$\begin{aligned} \mathcal{D}\mathbf{x} &= \mathbf{y}, \\ \mathbf{0} &= \mathbf{g}(t, \mathbf{x}, \mathbf{y}). \end{aligned} \tag{4.9}$$

This DAE, subject to the same boundary conditions on $\mathbf{x}$ that come with (4.7), can be solved by COLDAE using the general index option of selective projected collocation.

5. Numerical examples and further discussion. In the tables below we employ the following notation, unless otherwise specified: N_j denotes the size of the final mesh needed to satisfy the error tolerances (as estimated by the code), when using the jth projection option among the three mentioned at the end of the previous section—if no such convergence is reached then this is indicated by $*$; $erru_j$ denotes the maximum error on all components of $\mathbf{u}$ measured at 101 equidistant points, in case the exact solution is known, when the jth projection method is used; $erry_j$ denotes similar errors on $\mathbf{y}$ components; $ermsh_j$ denotes similarly the maximum error in $\mathbf{u}$ at mesh points on the final mesh.

5.1. Projected collocation and mesh selection.

Example 1. Consider for $0 \le t \le 1$,

$$\begin{aligned} x_1' &= \left(\nu - \frac{1}{2-t}\right) x_1 + (2-t)\nu y + q_1(t), \\ x_2' &= \frac{\nu - 1}{2-t} x_1 - x_2 + \left(\nu - 1 - \frac{\nu p(t)}{2+t}\right) y + q_2(t), \\ 0 &= (t + 2 - p(t)) x_1 + (t^2 - 4) x_2 + r(t) \end{aligned}$$

with $x_1(0) = 1$ and $p(t)$ a known function to be specified below. This example has been analyzed, in slight variations and with $p \equiv 0$, in [7] and [8]. Here $\nu \ge 1$ is a parameter. The inhomogeneities $\mathbf{q}$ and r are chosen to be

$$\begin{aligned} \mathbf{q} &= \begin{pmatrix} \frac{3-t}{2-t} e^t \\ \left(2 + \frac{(\nu+2)p(t) + p'(t)}{t^2-4} - \frac{2tp(t)}{(t^2-4)^2}\right) e^t \end{pmatrix}, \\ r &= -(t^2 + t - 2) e^t \end{aligned}$$

so that the exact solution is $x_1 = e^t$, $x_2 = (1 + \frac{p(t)}{t^2-4}) e^t$, $y = -\frac{e^t}{2-t}$.

First we let $p(t) \equiv 0$. While this is a linear, pure index-2 initial value problem, it can be particularly nasty, depending on the value of ν:

- The stability constant for unprojected collocation is exponentially large in ν, while that for projected collocation (and for the problem itself) grows only linearly in ν.
- Even the projected collocation scheme exhibits a behaviour common to nonstiff integration methods, requiring a small step-size h when $\nu h \gg 1$ (see [8]).

In Table 5.1 we record results of running COLDAE with and without projection. The initial mesh in all cases was uniform with $N = 5$, the number of collocation points per mesh subinterval was $k = 4$, a maximum mesh size of 100 subintervals was imposed and the tolerance on both components of $\mathbf{u} = \mathbf{x}$ was $1.e-5$. (Note also the additional boundary condition used, $x_1(0) - 2x_2(0) = -1$, as dictated by the constraint.)

The advantage of using the projected collocation method is clearly demonstrated for this example. Note the recovery of the superconvergence results for moderate values of ν.

TABLE 5.1
Errors for Example 1.

ν	N_1	$erru_1$	$erry_1$	$ermsh_1$	N_2	$erru_2$	$erry_2$	$ermsh_2$
1	10	.86e–8	.10e–4	.86e–8	10	.12e–8	.87e–5	.89e–15
10	10	.13e–4	.23e–3	.13e–4	10	.15e–7	.87e–5	.80e–11
50	*				10	.44e–6	.86e–5	.80e–7
100	*				10	.37e–6	.87e–5	.11e–6

Next, we consider this DAE with

$$p(t) = -\left(1 + \operatorname{erf}\left(\frac{t - 1/3}{\sqrt{2\varepsilon}}\right)\right)$$

with the parameter $0 < \varepsilon \ll 1$. This function varies rapidly (like $\varepsilon^{-1/2}$) around $t = \frac{1}{3}$, and this is reflected in the form of the solution $x_2(t)$. Note that the EUODE remains stable here, because $p \leq 0$, $p' \leq 0$.

We have used this example to test our mesh selection procedure (the fact that this is an initial value problem is immaterial here: the approach remains as if this were a boundary value problem). In Figs. 5.1 and 5.2 we display the mesh consisting of every second point of the final COLDAE mesh (cf. [4]) and the solution x_2 based on that mesh, when running for $\varepsilon = 1.e-5$, $\nu = 20$ (the latter excludes the nonprojection option $proj_1$), using a tolerance $1.e-5$ on both components of $\mathbf{x}$, a uniform initial mesh of five subintervals, and $k = 4$. For Fig. 5.1, we used essentially the COLSYS mesh selection procedure for $\mathbf{x}$. While the layer region has been detected, the mesh to the left of the layer is clearly much finer than it could have been. In general, it has been our experience that this procedure needs a smaller h in the current mesh, to recognize the need to redistribute it, than the same procedure requires with the projected solution. For Fig. 5.2 we projected $\mathbf{x}$ on the algebraic solution manifold, as described at the end of §3, before applying the mesh selection procedure. The resulting mesh has the layer better located, and therefore a smaller mesh (80, instead of 139 subintervals) is needed for achieving roughly the same accuracy.

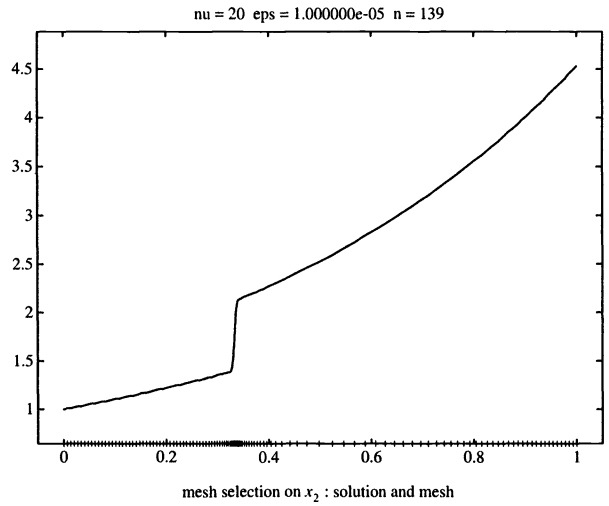


FIG. 5.1. *Example* 1: *Unprojected mesh selection.*

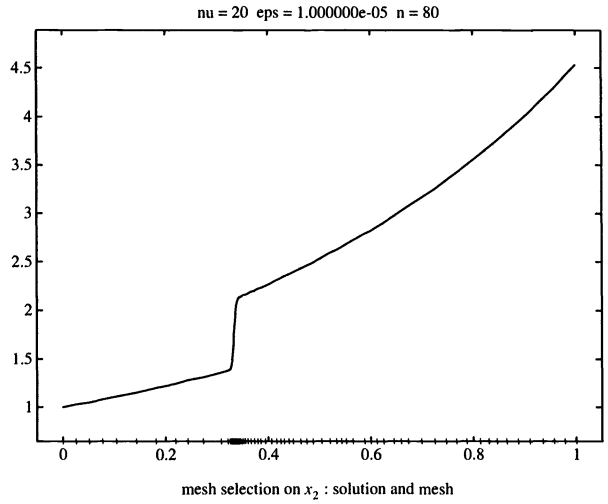


FIG. 5.2. *Example* 1: *Projected mesh selection.*

5.2. Selective projected collocation.

Example 2. The following example has been developed from one proposed by S. Reich (private communication). It is an instance of a nonlinear, semi-explicit DAE of index at most 2:

$$\begin{aligned} x_1' &= (\varepsilon + x_2 - p_2(t))y + p_1'(t), \\ x_2' &= p_2'(t), \\ x_3' &= y, \\ 0 &= (x_1 - p_1(t))(y - e^t), \end{aligned}$$

which we want to solve for the boundary conditions

$$x_1(0) = p_1(0), \quad x_2(1) = p_2(1), \quad x_3(0) = 1.$$

Here $\varepsilon > 0$ is a parameter and p_1, p_2 are given functions. For the results reported in Table 5.2 we chose

$$p_1 = p_2 = \sin t.$$

This problem has two isolated solutions that can be easily computed.

• One solution is obtained by setting $y = e^t$. This yields $x_3 = e^t$, $x_2 = p_2(t)$ (in any case), and $x_1 = p_1(t) + \varepsilon(e^t - 1)$. The linearized problem around the exact solution has index 1 and nothing exciting happens in addition, except that the conditioning deteriorates as $\varepsilon \to 0$.

• The other solution is obtained by setting $x_1 = p_1(t)$, which is the other possibility for satisfying the constraint. This yields $x_2 = p_2(t)$, $y = 0$, and $x_3 = 1$. The variational problem at this solution has index 2, with conditioning growing with ε^{-1}. For $\varepsilon = 0$, y and x_3 are not defined in a locally unique way. Yet, letting $e_1 = (x_1^\pi)' - p_1'$ and $e_2 = x_2^\pi - p_2$, we see from the equation for x_1' that for all $\varepsilon \geq 0$ small enough compared to h, the error in y depends only on e_1 and e_2, which in turn depend only on h and not on ε. Thus, the same numerical solution is obtained when $\varepsilon h^{-1} \to 0$. The numerical solution is locally unique and the stability constant deteriorates only as a negative power of h.

In Table 5.2 we record results of running COLDAE with and without projection. The solution to which the code converges depends, of course, on the initial guess. Since the index may be 1 or 2, depending on the current iterate, the option of always projecting is not used, and the options compared are never projecting (option 1) vs. selective projected collocation (option 3). The initial parameter setup for COLDAE was as in the previous example. In addition to the information described for Table 5.1, we also write under "sln" whether the first or the second exact solution is approximated.

TABLE 5.2
Errors for Example 2.

sln	ε	N_1	$erru_1$	$erry_1$	$ermsh_1$	N_3	$erru_3$	$erry_3$	$ermsh_3$
1	1	10	.75e–9	.16e–6	.11e–13	10	.75e–9	.16e–6	.11e–13
1	1.e–4	10	.65e–9	.16e–6	.41e–10	10	.65e–9	.16e–6	.41e–10
1	1.e–8	10	.58e–7	.46e–5	.54e–7	10	.58e–7	.46e–5	.54e–7
2	1	10	.10e–7	.20e–5	.10e–7	10	.12e–8	.19e–6	.14e–14
2	1.e–4	10	.10e–3	.20e–1	.10e–3	10	.12e–4	.19e–2	.48e–9
2	1.e–8	*	*	*	*	40	.11e–3	.73e–1	.16e–6

We note that the values listed in Table 5.2 depend on the initial guess (e.g., for the last entry, significantly better errors were obtained when starting from a different initial guess). For the index-1 solution, both options give precisely the same results, of course. For the index-2 solution, the projection option is significantly better. When we set $\varepsilon = 1.e - 8$, we could not obtain convergence for any method, until we increased the initial mesh size to $N = 20$. Then convergence was obtained for the selective projection method, but not for the unprojected one.

5.3. Fully implicit, index-1 problems. A class of problems where (selective) projected collocation proves useful, is fully implicit index-1 DAEs (4.7) converted to semi-explicit index-2 DAEs (4.9). We have used COLDAE to solve the example in [3] in this way (the unprojected collocation method does very poorly here and the projected method does very well, similar to Example 1). We have also solved a version of the detonation problem considered in [15]. This is an instance where the availability of a general-purpose tool like COLDAE allows one to spend less time on developing special-purpose methods and codes.

The price paid in solving (4.9) instead of (4.7) is that the size of the system appears to have doubled. However, note that in applications (e.g., the detonation problem), often only part of the given system is not in semi-explicit form to begin with. Thus, the size of the resulting higher index DAE can be much less than doubled. Also, the resulting index-2 DAE is often already in pure form, so no SVD is needed for its solution in such cases. Finally, we remark that if the problem contains inhomogeneities with jump discontinuities, then such functions should be defined like the approximate solution, i.e., with a mesh point placed at the discontinuity location and the inhomogeneity defined to be continuous from the right. Appropriately discontinuous approximate solutions are then possible.

5.4. Shooting for initial guesses. Let us turn now to the question of solving initial value DAEs using COLDAE. This is certainly possible, in principle, viewing initial value problems as a special case of boundary value problems. But the code does not take any special advantage of the relative simplicity and locality of initial value problems, and therefore it is not competitive with initial value codes in general. One major difference is in the amount of storage required by COLDAE, which increases linearly with the number of steps N, whereas the amount of storage required by initial value codes is usually independent of N. The latter does not hold, however, if the approximate solution is to be known for all values t in a given interval $[a, b]$ simultaneously. That is the case when the intended use of the initial value solution is as an initial approximation ("guess") for a boundary value problem solution.

This is sometimes a useful idea (not only for DAEs, but also for boundary value ODEs). It may happen that a major difficulty in practice when solving nonlinear boundary value problems is in obtaining a sufficiently good initial guess to start a convergent damped Newton iteration from. Some users have in fact claimed that this difficulty gives shooting methods an advantage over COLSYS. But if a shooting method can indeed be applied (i.e., if the conditioning of the initial value problem is not much worse than the conditioning of the boundary value problem, and initial value solutions starting from a do reach b, cf., e.g., [6]), then we can also shoot *once* to obtain an initial approximation in the context of COLDAE.

This can be conveniently done in one calling (driver) program that first calls COLDAE to solve a (user-defined) initial value problem, and then calls COLDAE again to solve the desired boundary value DAE problem by continuation from the solution of the initial value one. Thus, the user has only to guess the initial values—the initial guess for other values of t is taken, for example, to be the same, i.e., constant. The quasilinearization iteration of COLDAE then amounts to a sort of waveform method. To facilitate this possibility further, we have implemented an option in COLDAE that performs uncontrolled (standard) Newton iterations without damping, until convergence of the nonlinear iteration is hopefully obtained. This has been used to obtain good initial guesses for some examples in the class of problems described next.

5.5. Optimal control and parameter estimation. There are many applications in which a DAE for state variables involves control functions. The control is to be determined so as to minimize some objective functional. For instance, in robotics one considers problems of trajectory optimization (see [18] and [28])

$$\min J = \psi(\mathbf{p}(T), \mathbf{p}'(T)) + \int_0^T L(\mathbf{p}(t), \mathbf{p}'(t), \mathbf{u}(t), t)dt \tag{5.1}$$

subject to the constrained multibody equations

$$M(\mathbf{p})\mathbf{p}'' = \mathbf{f}(\mathbf{p}, \mathbf{p}', \mathbf{u}) - G^T(\mathbf{p})\lambda, \tag{5.2a}$$

$$\mathbf{0} = \mathbf{g}(\mathbf{p}), \tag{5.2b}$$

and some side conditions

$$\mathbf{b}(\mathbf{p}(0), \mathbf{p}'(0), \mathbf{p}(T), \mathbf{p}'(T)) = \mathbf{0}, \tag{5.3}$$

where $\mathbf{p}$ are generalized body positions, M is a positive definite mass matrix, and $G = \partial\mathbf{g}/\partial\mathbf{p}$ is a constraint matrix with a full row rank. The control $\mathbf{u}$ appears in the applied forces $\mathbf{f}$, and the objective can be, for example, to find the trajectory that takes the system from one specified position to another in a minimum amount of time T.

Replacing (5.2b) by its derivative

$$\mathbf{0} = \mathbf{g}'(\mathbf{p}) = G(\mathbf{p})\mathbf{p}'$$

or by a combination

$$\mathbf{0} = \gamma\mathbf{g}(\mathbf{p}) + G(\mathbf{p})\mathbf{p}' \tag{5.4}$$

($\gamma > 0$, cf. [11]), the state equations form an index-2 DAE. The necessary conditions for this problem yield a boundary value DAE for $\mathbf{p}$, $\mathbf{p}'$, λ, and their adjoint variables (i.e., the obtained size is double that of (5.2); see, e.g., [17]). Assume that L is such that the resulting DAE has index 2. This is a boundary value problem even if the conditions (5.3) are given only at

$t = 0$. When specifying the boundary conditions for COLDAE, note that the constraint (5.2b) and its adjoint constraint equation must be specified by the boundary conditions at $t = 0$, and that these constraints *are* satisfied automatically at $t = T$, so the conditions at T must be *complementary* (see Example 3 below).

The use of COLDAE to solve a system like (5.1)–(5.3) has the advantage of availablity of a general-purpose software. However, we do not recommend it as a general "cure." First, there may be many equations in (5.2); second, the controls are often restricted by inequalities and this cannot be handled directly by COLDAE. Another note is that by concentrating on the necessary conditions for an extremum of (5.1)–(5.3), the sense of minimization is lost, so some information is not being used. Nonetheless, COLDAE is an available tool that can minimize the human effort in writing special-purpose programs for certain applications.

A somewhat-related problem is that of parameter estimation. A given DAE of a type covered by COLDAE depends on unknown parameters ω. The problem is to recover the parameters so that the DAE solution best fits given observations on the solution. Such problems are ill posed, and can be difficult to solve.

Example 3. Consider

$$\begin{aligned} x_1' &= x_2 + x_1 y, \\ x_2' &= -\omega^2 x_1 + x_2 y, \\ 0 &= \left(\frac{\pi}{3}\right)^2 x_1^2 + x_2^2 - 1, \\ x_1(0) &= 0, \qquad x_2(0) = 1, \end{aligned}$$

where ω is a constant parameter. This is a pure index-2 DAE and the exact solution for $\omega = \bar{\omega} := \frac{\pi}{3}$ is

$$x_1 = \omega^{-1} \sin \omega t, \quad x_2 = \cos \omega t, \quad y = 0.$$

Now, suppose that we are to find ω that best fits an observed function $r(t)$, where the observations are on $x_1(t) + x_2(t)$, $0 \le t \le 2$. The necessary conditions for minimizing $\frac{1}{2}\int_0^2 (x_1 + x_2 - r)^2 dt$ yield the boundary value DAE problem with six ODEs and two constraints, for $x_1, x_2, \omega, \lambda_1, \lambda_2, \nu, y$, and μ,

$$\begin{aligned} x_1' &= x_2 + x_1 y, \\ x_2' &= -\omega^2 x_1 + x_2 y, \\ \omega' &= 0, \\ \lambda_1' &= -y\lambda_1 + \omega^2 \lambda_2 - 2\left(\frac{\pi}{3}\right)^2 x_1 \mu - (x_1 + x_2 - r), \\ \lambda_2' &= -\lambda_1 - y\lambda_2 - 2x_2\mu - (x_1 + x_2 - r), \\ \nu' &= 2\omega x_1 \lambda_2, \\ 0 &= \left(\frac{\pi}{3}\right)^2 x_1^2 + x_2^2 - 1, \\ 0 &= x_1\lambda_1 + x_2\lambda_2, \\ x_1(0) &= 0, \quad x_2(0) = 1, \quad \nu(0) = 0, \quad \lambda_2(0) = 0, \\ \nu(2) &= 0, \quad x_2(2)\lambda_1(2) - \left(\frac{\pi}{3}\right)^2 x_1(2)\lambda_2(2) = 0. \end{aligned}$$

We now solve this problem twice, using COLDAE.

1. Setting $r(t) = \bar{\omega}^{-1} \sin \bar{\omega} t + \cos \bar{\omega} t$, we recover $\omega = \bar{\omega}$ to machine precision (with 20 uniform mesh elements and 4 collocation points per element).

2. Setting $r(t)$ to be the piecewise linear interpolant of the values $\bar{\omega}^{-1} \sin \bar{\omega} t + \cos \bar{\omega} t$ at 21 equidistant points, we recover $\omega = 1.3792$, which, as an approximation to $\bar{\omega}$, is in a relative error of 31.7%. Smoothing $r(t)$ by passing a cubic spline through these 21 points and rerunning has not helped much. Obviously, this problem is not well conditioned.

5.6. On regularization. Various authors have proposed *regularizing* a given DAE by adding to (1.6b) terms involving $\epsilon \mathbf{y}'$, turning the problem (1.6) into an ODE. The parameter ϵ, $0 < \epsilon \ll 1$, must be taken small to ensure that the solution of the obtained problem does not deviate much from the desired one. (In this description we do not consider stabilization methods, which maintain the exact solution in a reformulated problem, as regularization methods.) With COLDAE one can of course solve the obtained boundary value ODE. However, this approach is often unnecessary, and may introduce stiffness where none existed. It is not difficult to conjure up examples where the obtained regularized problem is more difficult to solve than the original one. An exception is a singularity point, where the local index of the DAE increases at that one point. We proceed with an example for the latter.

Example 4. The DAE

$$\begin{aligned} x_1' &= x_2 y - \alpha J, \\ x_2' &= y - 1, \\ 0 &= x_1 - \left(\frac{J^2}{y} + y \right) \end{aligned}$$

has been investigated in [5] as a simple model for the hydrodynamic semiconductor equations. The boundary conditions are

$$y(0) = y(\beta) = \bar{n}.$$

Here J, α, $\bar{n} > J$, and β are known positive constants. The resulting solution y is positive on $[0, \beta]$. The linearization of this DAE clearly has index 1, except where $y = J$. Cases where y becomes less than J (starting at $y(0) > J$) correspond to transonic flow. The solution is then discontinuous where y jumps back from $(< J)$ to $(> J)$ (see [5]).

In such a case, where y is discontinuous at an unknown location at which there is a singularity, one may be better off replacing the DAE by a regularizing ODE. The method used in [5] is equivalent to replacing the constraint above by

$$\epsilon y' = x_1 - \left(\frac{J^2}{y} + y \right)$$

insisting that the constraint be satisfied at $t = 0$, viz.,

$$x_1(0) = \frac{J^2}{\bar{n}} + \bar{n}.$$

Solutions with rather sharp layer profiles were obtained in [5] using COLDAE (i.e., COLNEW for the regularized boundary value ODE) by applying continuation in ϵ.

While this example shows that replacing the DAE by a regularizing ODE can be beneficial, we note that there are various types of possible singularities and that this is not necessarily a cure for all problems. Also, the regularized ODE and its additional boundary condition must be chosen with care.

REFERENCES

[1] U. ASCHER, *Collocation for two-point boundary value problems revisited*, SIAM J. Numer. Anal., 23 (1986), pp. 596–609.

[2] ———, *On numerical differential algebraic problems with application to semiconductor device simulation*, SIAM J. Numer. Anal., 26 (1989), pp. 517–538.

[3] ———, *On symmetric schemes and differential-algebraic equations*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 937–949.

[4] U. ASCHER, J. CHRISTIANSEN, AND R. RUSSELL, *Collocation software for boundary value* ODEs, ACM Trans. Math. Software, 7 (1981), pp. 209–222.

[5] U. ASCHER, P. MARKOWICH, P. PIETRA, AND C. SCHMEISER, *A phase plane analysis of transonic solutions for the hydrodynamic semiconductor model*, Math. Models Methods Appl. Sci., 1 (1991), pp. 347–376.

[6] U. ASCHER, R. MATTHEIJ, AND R. RUSSELL, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equation*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[7] U. ASCHER AND L. PETZOLD, *Projected implicit Runge–Kutta methods for differential-algebraic equations*, SIAM J. Numer. Anal., 28 (1991), pp. 1097–1120.

[8] ———, *Stability of computational methods for constrained dynamics systems*, SIAM J. Sci. Comput., 14 (1993), pp. 95–120.

[9] ———, *Projected collocation for higher-order higher-index differential-algebraic equations*, J. Comput. Appl. Math., 43 (1992), pp. 243–259.

[10] G. BADER AND U. ASCHER, *A new basis implementation for a mixed order boundary value* ODE *solver*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 483–500.

[11] J. BAUMGARTE, *Stabilization of constraints and integrals of motion in dynamical systems*, Comput. Math. Appl. Mech. Engrg., 1 (1976), pp. 1–16.

[12] C. DE BOOR AND B. SWARTZ, *Collocation at Gaussian points*, SIAM J. Numer. Anal., 10 (1973), pp. 582–606.

[13] B. P. BOUDREAU, *A steady-state diagenetic model for dissolved carbonate species and pH in the porewaters of oxic and suboxic sediments*, Geochimica et Cosmochimica Acta, 51 (1987), pp. 1185–1196.

[14] B. P. BOUDREAU AND D. E. CANFIELD, *A provisional diagenetic model for pH in anoxic porewaters: Application to the* FOAM *site*, J. Marine Research, 46 (1988), pp. 429–455.

[15] M. BRAITHWAITE, T. FARRAN, I. GLADWELL, P. LYNCH, A. MINCHINTON, I. PARKER, AND R. THOMAS, *A detonation problem posed as a differential/algebraic boundary value problem*, Math. Engrg. Ind., 3 (1990), pp. 45–57.

[16] K. BRENAN, S. CAMPBELL, AND L. PETZOLD, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, North-Holland, Amsterdam, 1989.

[17] A. BRYSON AND Y.C. HO, *Applied Optimal Control*, Ginn and Co., Waltham, MA, 1969.

[18] F. DELEBEQUE AND R. NIKOUKAH, *On simulation and control of multibody mechanical systems*, Lecture at the ARO Workshop on Mechanical Systems and Vehicle Simulation, Raleigh, NC, Nov. 5–7, 1992.

[19] C. W. GEAR, *Differential-algebraic equation index transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 39–47.

[20] C. W. GEAR AND L. R. PETZOLD, ODE *methods for the solution of differential-algebraic systems*, SIAM J. Numer. Anal., 21 (1984), pp. 716–728.

[21] P. GILL, W. MURRAY, M. SAUNDERS, G .W. STEWART, AND M. WRIGHT, *Properties of a representation of a basis for the null space*, Math. Prog., 33 (1985), pp. 172–186.

[22] E. GRIEPENTROG AND R. MARZ, *Differential-Algebraic Equations and their Numerical Treatment*, Teubner-Texte Math. 88, Teubner, Leipzig, 1986.

[23] E. HAIRER, Ch. LUBRICH, AND M. ROCHE, *The numerical solution of differential-algebraic systems by Runge–Kutta methods*, Lecture Notes in Math Vol. 1409, Springer-Verlag, New York, 1989.

[24] E. HAIRER AND G. WANNER, *Solving Ordinary Differential Equations II*, Springer-Verlag, New York, 1991.

[25] M. HO, *A collocation solver for systems of boundary-value differential/algebraic equations*, Computers and Chem. Engrg., 7 (1983), pp. 735–737.

[26] J. S. LOGSDON AND L. T. BIEGLER, *Accurate solution of differential-algebraic optimization problems*, Indust. Engrg. Chemistry Research, 28 (1989) pp. 1628–1639.

[27] A. MACIEJEWSKI AND C. KLEIN, *The singular value decomposition: computation and applications to robotics*, Internat. J. Robotics Res., 8 (1989), pp. 63–79.

[28] M. STEINBACH, H. G. BOCK, AND R. W. LONGMAN, *Time-optimal extension or retraction in polar coordinate robots: a numerical analysis of the switching structure*, Proc. AIAA Guidance, Navigation and Control, Boston, Aug. 1989.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 953–958, July 1994

010

SHIFTING STRATEGIES FOR THE PARALLEL QR ALGORITHM*

DAVID S. WATKINS†

Abstract. The use of high-order generalized Rayleigh quotient shifting strategies is advocated as a means of improving the parallel performance of the double-shift QR algorithm for nonsymmetric eigenvalue problems.

Key words. QR algorithm, parallel implementation, shifts

AMS subject classifications. 65F15, 15A18

Consider the problem of finding all eigenvalues of a real or complex square matrix A. Assume that A has no special structure to exploit. Thus it is not banded, sparse, or Hermitian, for example. The QR algorithm [6], [21] has been the method of choice for solving such problems for the past 30 years. However, in recent years it has been suggested that the days of the QR algorithm may be numbered, for the QR algorithm has proven difficult to parallelize. The purpose of this note is to suggest that such predictions are at best premature. A simple modification of the shifting strategy promises to improve the parallel performance of the QR algorithm significantly.

Recall that the QR algorithm is an iterative process that produces a sequence of unitarily similar matrices (A_k) that converges to something like upper triangular form, thereby revealing the eigenvalues. Before the QR iterations are begun, the matrix is reduced to upper Hessenberg form by a similarity transformation, so that the QR algorithm can operate on upper Hessenberg matrices. Thus each A_k is upper Hessenberg. To accelerate convergence, we shift the matrix at each step by a shift that approximates an eigenvalue (or so we hope).

An important feature of the QR algorithm is that it has an implicit version that allows several QR steps (as many as one wants) to be taken at a time. The value of this feature was recognized immediately and has been exploited for the past 30 years in the following way. If A is a real matrix, one would prefer to work entirely in real arithmetic. However, A may have some (perhaps many) pairs of complex conjugate eigenvalues. If one wishes to achieve rapid convergence to these eigenvalues, one must use complex shifts, thereby introducing complex arithmetic. Fortunately, if one performs two steps at once with complex conjugate shifts σ and $\bar{\sigma}$, the result is real. Furthermore, the implicit double step can be carried out in real arithmetic. For this reason, the standard implementations of the QR algorithm for real, nonsymmetric matrices perform double steps. For details see [6], [7], [18], or [21], for example.

A few years ago Bai and Demmel [3] proposed the *multishift* QR algorithm, which performs QR steps of multiplicity higher than two. (By a QR step of *multiplicity* m we mean m single steps carried out simultaneously by the implicit algorithm.) Steps of high multiplicity (say 30 or 60) entail a great deal more arithmetic than the standard single and double steps; the amount of arithmetic is roughly proportional to the multiplicity of the step. Furthermore, the arithmetic in a step with high multiplicity can be organized as matrix-vector and even matrix-matrix operations, thus allowing the steps to be programmed in high-level basic linear algebra subprograms (BLAS). The expected payoff of this approach is that the high-level BLAS can be implemented in an optimal way for each computer, exploiting the vector pipelines, hierarchical memory, parallel processors, or whatever features a given machine happens to have. This was a very promising idea.

If one wishes to perform, say, 60 steps at once, one needs 60 shifts. Where are we to get so many shifts? The standard strategy for the double QR algorithm is to use as shifts the

*Received by the editors September 16, 1992; accepted for publication (in revised form) June 11, 1993.

†Department of Pure and Applied Mathematics, Washington State University, Pullman, Washington 99164-3113. Current address, 6835 24th Avenue NE, Seattle, Washington 98115-7037 (na.watkins@na-net.ornl.gov).

eigenvalues of the 2×2 submatrix in the lower right-hand corner of the matrix. The natural extension of this strategy for steps of multiplicity m is to calculate the eigenvalues of the $m \times m$ submatrix in the lower right-hand corner of the matrix and use these as shifts. Although this might seem costly, it is actually relatively inexpensive as long as $m \ll n$, where n is the order of the matrix. Recalling that the complexity of the Hessenberg eigenvalue problem is $O(n^3)$, we see that we can easily afford to compute 60 shifts in this way if, say, $n = 600$. This shifting strategy is designed to force the entry $a_{n-m+1,n-m}$ to zero, thereby isolating an $m \times m$ submatrix (and m eigenvalues) in the lower right-hand corner. Watkins and Elsner [20] have shown that the local convergence rate is usually quadratic. (The problem of finding an efficient, *globally* convergent shifting strategy remains open.) We call this strategy the *generalized Rayleigh quotient* shift strategy, because in the case where $m = 1$, it reduces to the Rayleigh quotient shift. Various researchers have experimented with different strategies, but this is the best that has been found so far.

When Bai and Demmel programmed the multishift QR algorithm with the generalized Rayleigh quotient shift, they found out that the algorithm has a flaw. At the outset it must grind for many iterations before the first batch of eigenvalues is deflated from the matrix. The problem becomes worse as the value of m is increased. Once the first deflation has taken place, the algorithm begins to behave as one would hope, but by this time the damage has already been done. For example, when I applied the QR algorithm with $m = 38$ to a random matrix with $n = 250$, it took 20 iterations (of multiplicity 38) before the first deflation took place. This is too many. The double QR algorithm typically isolates about one pair of eigenvalues per five iterations, at least in the early stages. A step of multiplicity m has about the same cost in flops as $m/2$ double steps. Therefore, if the multishift QR algorithm is to be competitive, it must isolate a block of m eigenvalues in about five (multiple) iterations. Thus the algorithm took something like four times as many iterations as we would have liked. Even though the algorithm behaved well for the remainder of the run, it still took 71 seconds on a DECstation 5000 (working in double precision) to find all of the eigenvalues. In contrast, the standard EISPACK [12] code HQR (post- 1989 version from NETLIB) took 24 seconds for the same matrix on the same workstation. If the multishift QR algorithm is to be a success, this problem must be overcome. So far nobody has found a solution. A multishift QR program has been included in LAPACK [1], but the designers have decided to release it with $m = 6$, which is small enough that the problem of delayed convergence is not observed. This modest value of m hardly allows the kind of performance that had been hoped for. At this point it appears that multishift QR was a good idea that did not pan out.

It is tempting to blame the shifting strategy. Perhaps in the early iterations the shifts just are not good enough. This is what I thought until an eye-opening experiment by Dubrulle [5] showed otherwise. Dubrulle wrote two codes that perform multiple steps in two different ways. They both use the generalized Rayleigh quotient shift strategy to get m shifts, where m is an even number, say, 30. The first code is a multishift code; that is, it performs steps of multiplicity m, just as we have been discussing. The other uses the m shifts to perform a batch of $m/2$ double steps, one after the other. In principle, that is, in the absence of roundoff errors, the two codes should yield identical results. In practice the outcomes were quite different. Whereas the multishift code had the convergence problem that we have just been discussing, the code that performed batches of double steps had satisfactory convergence properties. I replicated these results with experiments of my own. For example, I ran code that performed steps with $m = 38$, implemented as batches of 19 double steps, on the same 250×250 matrix as mentioned above. After only five such batches, the first chunk of 33 eigenvalues deflated from the problem. Although we would have preferred to get 38 eigenvalues, 33 was not bad. The main point is that the deflation happened after five batches, not 20. The modified code took only 33 seconds to find all of the eigenvalues, so it was almost competitive with HQR.

These experiments demonstrate that the fault does not lie with the generalized Rayleigh quotient shift strategy. The fact that the two codes behaved so differently when they should have produced identical results shows clearly that the problem lies with roundoff errors. Because the QR algorithm is forward unstable, small differences in the roundoff errors can cause huge differences in the result. Somehow the roundoff errors destroy high-multiplicity QR steps; subdiagonal entries that are supposed to converge to zero simply do not. The process is not yet understood.[1]

The performance of the code that does double steps in batches can be made even better by changing the deflation strategy. In the original implementation the code did not check for the possibility of deflation except between batches. I wrote the code that way because I wanted it to mimic the multishift QR code exactly, except in the way the step was executed. However, if one performs the multiple step as a batch of double steps, one has the possiblity of checking for possible deflations after each double step. This costs little, and it is the right way to implement the step on a sequential machine. Otherwise deflations that occur in the middle of a batch can be washed out and lost. With this more vigilant strategy, the code that performed double steps in batches of 19 was able to finish off the 250×250 matrix in 24 seconds, the same as HQR.

It is interesting to look at the pattern of deflations. The first deflations took place in the third batch, during which 18 eigenvalues were isolated. Two batches later another 22 eigenvalues emerged. On the following batch, 13 more came out, and each subsequent batch delivered a few more. Thus with this deflation strategy the eigenvalues emerge in smaller bunches but more frequently.

With the new deflation strategy the batch algorithm is not really a high-multiplicity algorithm in any sense. It is simply an implementation of the double QR algorithm with a novel shift strategy, the generalized Rayleigh quotient shift strategy with large m. My code does not actually work with a fixed value of m; it adjusts m depending on the size of the matrix. As the deflations take place, the code must operate on submatrices of various dimensions. If the size of the submatrix that is being processed at a given moment is k, it takes m to be an even number that is approximately $k^{2/3}$, except that m is not allowed to exceed 50. (Note that $38 \approx 250^{2/3}$.) Also m is set to 2 whenever $k \leq 50$. This includes the submatrices from which the shifts are calculated. Of course there are numerous other strategies that one might try. Which strategy is best depends on the computer, the data structures, and other implementation details.

With these parameters, the generalized Rayleigh quotient shift strategy has proved to be competitive with standard implementations of the QR algorithm. Table 1 illustrates run times for four implementations of the QR algorithm on a 600×600 matrix on one processor of a Cray X-MP in single precision. The matrix was obtained by reducing a matrix of normally distributed random numbers (mean 0, variance 1) to upper Hessenberg form using the EISPACK code ELMHES.

The HQR code is from NETLIB. The LAPACK code is the aforementioned multishift code with multiplicity 6. GR is an LR/QR hybrid written by Haag and Watkins [4]. Here I have the parameters set so that it performs straight QR. GR uses the standard $m = 2$ shift strategy. GRBS (block shift) is identical to GR, except that it uses the generalized Rayleigh quotient shift strategy as specified above. GR is included here so that we can compare apples with apples. Clearly, the switch to the Rayleigh quotient shift strategy gives improved performance on this problem. Of course, one should not place too much importance on these exact numbers; they vary from one matrix to the next. Furthermore, none of these codes has been optimized

[1] Parlett and Le [11] have analyzed forward instability of single QR steps ($m = 1$) in the symmetric case. Watkins [19] has discussed the nonsymmetric case, also for $m = 1$.

TABLE 1

	Time (seconds)	Time (relative)
EISPACK's HQR	16.24	1.00
LAPACK's SHSEQR ($m = 6$)	15.99	0.98
GR	14.50	0.89
GRBS	11.81	0.73

for the Cray. It was left to the compiler to decide what to vectorize. The important point is not that GRBS beats the other codes but that it is competitive. Having raced GRBS against HQR and other codes on various machines, using random and nonrandom matrices of various sizes, I have found that GRBS is sometimes faster, sometimes slower, but generally in the same ballpark.

The most important advantage of the new shifting strategy becomes evident when we think about implementing the algorithm in parallel. A double QR step on a Hessenberg matrix proceeds roughly as follows. An initial similarity transformation creates a small bulge in the Hessenberg form at the upper left hand corner of the matrix. The subsequent transformations return the matrix to upper Hessenberg form by chasing the bulge step by step down the diagonal until it disappears off the edge of the matrix. If we have 60 shifts in hand, we have enough to do 30 such bulge chases. Once we have set one bulge in motion, there is no need to complete the step before setting the second bulge in motion. In fact, once the first bulge has travelled just a small distance down the diagonal, there is room to start the second bulge. Shortly thereafter we can start a third bulge, and so on. In this way we can chase a large number of bulges at once in pipeline fashion. This scheme generates enough arithmetic to keep a good many processors busy. The arithmetic is taking place not only on the diagonal, but all along the rows and columns. It should be possible to implement this scheme on parallel computers of both shared memory and distributed memory types [15], [17]. Also, the scheme could be implemented in a systolic array.

The idea of pipelining QR steps is not new. For example, it has been considered by Heller and Ipsen [8], Stewart [13], van de Geijn [14], [16], and Kaufman [9], but the idea has not caught on in a big way so far. An important reason for this is that until now nobody has advocated changing the shifting strategy. It has been assumed that one should continue to use the standard shifts, the eigenvalues of the lower right-hand 2×2 submatrix. The entries of this submatrix are among the last to be computed in a QR step, for the bulge is chased from top to bottom. If one wishes to start a new step before the bulge for the current step has reached the bottom of the matrix, one is forced to use old shifts because the new ones are not available yet. If one wants to keep a steady stream of, say, four bulges running in the pipeline, as in Fig. 1, one is forced to use shifts that are three iterations out of date. Van de Geijn [16] analyzed the effect of using such shifts, which he calls deferred shifts, on the rate of convergence. He showed that if deferred shifts are used, the convergence rate ceases to be quadratic but remains superlinear. Precisely, he showed that if the shifts are deferred by h iterations, the convergence rate is τ, where τ is the unique positive root of $t^{h+1} - t^h - 1 = 0$. One easily shows that τ lies strictly between 1 and 2, which means that the convergence is superlinear but subquadratic. When $h = 1$, which means that the shifts are one iteration out of date, $\tau = (1 + \sqrt{5})/2 \approx 1.62$, the golden ratio. This is the same as the convergence rate of the secant method for solving nonlinear equations [2]. As h is increased, τ decreases steadily. When $h = 3$, $\tau \approx 1.38$, and when $h = 20$, $\tau \approx 1.11$. Thus we see that if we pipeline large numbers of bulges using the standard shifting strategy, the convergence rate will be degraded severely.

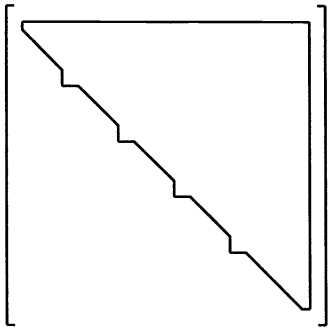


FIG. 1. *Pipelined QR steps.*

In contrast to these results, if we use the generalized Rayleigh quotient shifting strategy with, say, $m = 40$, we can chase up to 20 bulges in pipeline without losing quadratic convergence. There is, however, a catch. If we wish to maintain quadratic convergence, we must finish each batch, that is, we must empty the pipeline, before computing the shifts for the next batch. If we wish to keep the pipeline full at all times, we cannot avoid using old shifts. However, by choosing m large or spacing the bulges appropriately (the optimal strategy depending on the architecture), we can always organize the computation so that the shifts are never more than one batch old. The analysis of van de Geijn is valid not only for the standard shifts but for generalized Rayleigh quotient shifts as well. Therefore, if our shifts are just one batch out of date, we get a convergence rate of about 1.62, which is far better than 1.11, for example.

Some preliminary results. Kaufman [10] has obtained some preliminary timings for pipelined QR steps using a generalized Rayleigh quotient shifting strategy on a MASPAR massively-parallel computer. This is a single instruction multiple data (SIMD) machine consisting of an array of 16384 very slow processors connected to a much faster front-end processor. To get any benefit at all from the parallel processing unit, one must make very good use of the processors. If an algorithm does not have a great deal of inherent parallelism, then there is no point in using the parallel unit at all. For example, to calculate all of the eigenvalues of a 400×400 matrix using HQR took 178 seconds on the front end versus 1074 seconds on the parallel unit.

Kaufman wrote a version of QR that uses the generalized Rayleigh quotient shifting strategy with $m = n/10$ and performs batches of $m/2$ double steps. Each batch is completed before the new shifts are calculated, so the shifts are always up to date. Checks for possible deflations are performed only between batches. The shifts are computed by HQR on the front end, and once the size of the matrix is deflated to $3m$, the remaining submatrix is processed by HQR on the front end. Some timing results for random matrices are given in the Table 2.

TABLE 2

	Run times in seconds		
n	HQR on front end	HQR on parallel unit	New shift strategy on parallel unit
400	178	1074	223
600	621	2829	472
800	1469		845

The most striking result is that the new algorithm is some five times faster than HQR on the parallel unit. For large enough matrices, it even beats HQR on the front end by a substantial amount. From the trend illustrated by the table, we can expect that the new algorithm will perform even better on matrices that are larger still.

The same tests were performed using the upper Hessenberg matrices given by $h_{ij} = i - j$ for $n = 400$, 600, and 800 with similar results.

Kaufman also wrote a code for symmetric, tridiagonal matrices that uses the generalized Rayleigh quotient shift strategy and chases large batches of bulges in pipeline. This code also performed well in certain situations. On one processor of a Cray Y- MP it was significantly faster than a number of the fastest symmetric QR codes known. Some results are given in [9].

Acknowledgment. I would like to thank Linda Kaufman for very generously permitting me to include her preliminary results in this paper.

REFERENCES

[1] E. ANDERSON ET. AL., *LAPACK Users' Guide*, Society for Industrial and Applied Mathemathics, Philadelphia, 1992.

[2] K. E. ATKINSON, *An Introduction to Numerical Analysis,* John Wiley and Sons, New York, 1978.

[3] Z. BAI AND J. DEMMEL, *On a block implementation of the Hessenberg multishift QR iteration*, Internat. J. High Speed Comput., 1 (1989), pp. 97–112.

[4] J. B. HAAG AND D. S. WATKINS, *QR-like algorithms for the nonsymmetric eigenvalue problem*, ACM Trans. Math. Software, 19 (1993), pp. 407–418.

[5] A. A. DUBRULLE, *The multishift QR algorithm—is it worth the trouble?*, TR 6320-3558, IBM Scientific Center, Palo Alto, CA, 1991.

[6] J. G. F. FRANCIS, *The QR transformation, parts* I *and* II, Computer J., 4 (1961), pp. 265–272, 332–345.

[7] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, 1989.

[8] D. E. HELLER AND I. C. F. IPSEN, *Systolic networks for orthogonal equivalence transformations and their applications*, Conference on Advanced Research in VLSI, Massachusetts Institute of Technology, Cambridge, 1982.

[9] L. KAUFMAN, *A parallel QR algorithm for the symmetric eigenvalue problem*, J. Parallel and Distributed Computing, to appear.

[10] ———, Private communication, 1993.

[11] B. N. PARLETT AND J. LE, *Forward instability of tridiagonal QR*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 279–316.

[12] B. T. SMITH ET. AL., *Matrix Eigensystem Routines, EISPACK Guide*, 2nd ed., Springer-Verlag, New York, 1970.

[13] G. W. STEWART, *A parallel implementation of the QR algorithm*, Parallel Comput., 5 (1987), pp. 187–196.

[14] 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, 1987.

[15] ———, *Storage schemes for parallel eigenvalue algorithms*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. H. Golub and P. Van Dooren, eds., NATO ASI Series, Springer Verlag, 1991, pp. 639–647.

[16] ———, *Deferred shifting schemes for parallel QR methods*, SIAM J. Matrix Anal. Appl., 14 (1993) pp. 180–194.

[17] R. A. VAN DE GEIJN AND D. G. HUDSON, *An efficient parallel implementation of the nonsymmetric QR Algorithm*, Proc. 4th Conf. on Hypercube Concurrent Computers and Applications, Monterey, CA, March, 1989.

[18] D. S. WATKINS, *Fundamentals of Matrix Computations*, John Wiley and Sons, New York, 1991.

[19] ———, *Forward stability and transmission of shifts in the QR algorithm,* SIAM J. Matrix Anal. Appl., to appear.

[20] D. S. WATKINS AND L. ELSNER, *Convergence of algorithms of decomposition type for the eigenvalue problem*, Linear Algebra Appl., 143 (1991), pp. 19–47.

[21] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford University, 1965.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 959–976, July 1994

011

A SHADOWING LEMMA APPROACH TO GLOBAL ERROR ANALYSIS FOR INITIAL VALUE ODES*

SHUI-NEE CHOW† AND ERIK S. VAN VLECK‡

Abstract. The authors show that for dynamical systems that possess a type of piecewise hyperbolicity in which there is no decrease in the number of stable modes, the global error in a numerical approximation may be obtained as a reasonable magnification of the local error. In particular, under certain conditions the authors prove the existence of a trajectory on an infinite time interval of the given ordinary differential equation uniformly close to a given numerically computed orbit of the same differential equation by allowing for different initial conditions. For finite time intervals a general result is proved for obtaining a posteriori bounds on the global error based on computable quantities and on finding and bounding the norm of a right inverse of a particular matrix. Two methods for finding and bounding/estimating the norm of a right inverse are considered. One method is based upon the choice of the pseudo or generalized inverse. The other method is based upon solving multipoint boundary value problems (BVPs) with the choice of boundary conditions motivated by the piecewise hyperbolicity concept. Numerical results are presented for the logistic equation, the forced pendulum equation, and the space discretized Chafee–Infante equation.

Key words. numerical initial value ODEs, global error analysis, piecewise hyperbolicity

AMS subject classification. 65L

1. Introduction.

In this paper we consider initial value ordinary differential equations and their discretization. We investigate both theoretically and numerically the global error in computing numerical approximations. It is well known (see [Ge], [S1]) that the global error between the numerical approximation and the actual trajectory with the same initial condition may become large. In fact, for a numerical method of order p with a fixed stepsize of h and a differential equation with Lipschitz constant L, classical error estimates are of the form $\exp(Lt)h^p$ at time t. On the other hand, local error control provides a tight bound in many instances. For example, the local error is a good estimate of the global error for many stiff initial value problems in which one-sided Lipschitz constants appear. Our contribution is to show that global errors stay reasonably bounded for the wider class of initial value problems that are piecewise hyperbolic with no decrease in the number of stable modes where in fact the number of stable modes may increase. These are not stable initial value problems in the classical sense, and the global error may not be of the order of the local error. We will show that if the initial condition for the discrete approximation is allowed to differ from that of the continuous trajectory, then for a large class of problems the global error may be represented as a reasonable magnification of the local error. This is important when one is employing numerical simulations to study qualitative as well as quantitative features of dynamical systems. In this paper we reserve the word orbit to denote a discrete sequence of points and the word trajectory to denote a continuous function of time.

We consider the class of problems in which there is no decrease in the number of stable modes of the linear variational equation along solution paths. This is reminiscent of the case in which there are several hyperbolic fixed points (i.e., saddle points) and a trajectory that passes near these fixed points in which the dimension of the stable manifold of these fixed points is not decreasing. This situation arises in certain space discretized parabolic partial differential equations that occur as models of chemical, biological, and other physical systems.

*Received by the editors September 14, 1992; accepted for publication (in revised form) May 25, 1993.

†School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332 (chow@math.gatech.edu). The work of this author was supported in part under National Science Foundation grant DMS-9005420.

‡Department of Mathematics and Statistics, Simon Fraser University, Burnaby, British Columbia V5A 1S6 Canada. Current address, Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401 (erikvv@lyapunov.mines.colorado.edu). The work of this author was supported in part under Natural Sciences and Engineering Research Council of Canada grant OGP0121873.

Our method is based upon showing that there exists a nearby trajectory in which there is no local error and is somewhat similar to defect correction (see [S2]). Our approach is different from defect correction in that we do not attempt to provide a correction, but instead estimate the magnification of the local error that gives the global error. In [Be1] it is shown that in a neighborhood of a single hyperbolic fixed point, the discrete stable and unstable manifolds of the map defined by the numerical method converge to the stable and unstable manifolds of the fixed point of the continuous problem. Aspects of backward error analysis for initial value ordinary differential equations are studied in [E2] and under certain conditions it is shown that even on an infinite time interval there exists a nearby equation that is solved exactly by the numerical result.

The idea of showing that near an orbit with a small local error there exists an orbit with no local error is formalized in the dynamical systems community in terms of the shadowing lemma. Results in this direction were first given by Anosov [A] and Bowen [Bo] for uniformly hyperbolic maps on a differential manifold. These results were generalized, and recently an analytic proof of the shadowing lemma has been given in [CLP] under the assumption of exponential dichotomy. The infinite time result presented in this paper is proven under the assumption of piecewise exponential dichotomy in which the rank of the projection onto the stable subspace is, under certain assumptions, allowed to increase with time.

Numerical methods for computing the global error for maps, where the local error is comprised solely of roundoff error (as opposed to roundoff error and discretization error when one is solving differential equations numerically), were initially given in [HYG1] and [HYG2] for the logistic map and the Henon map. These mappings are not uniformly hyperbolic in the sense of Anosov and Bowen, but are on average hyperbolic. The methods used in [HYG1] and [HYG2] are based upon interval arithmetic to provide a sequence of intervals containing both the orbit with the small local error and the orbit with no local error. Other numerical methods for shadowing of maps have been given in [CP1], [CP2], and [CVV2]. In [SY] a new proof of the shadowing lemma is given and numerical methods are presented that apply to both mappings and ordinary differential equations. The methods in [SY] are based upon performing Newton's method to find a zero of a certain function. They answer a slightly different question than we do, by showing that there exists a noisy discrete approximation with some unknown initial condition near a trajectory of the same problem with a given initial condition. The methods in [SY] provide a rigorous verification that there exists a nearby trajectory and rigorous bounds on the distance from the trajectory to the discrete approximation using Taylor series methods to integrate numerically. Taylor series methods are used to obtain explicit bounds on the local errors although the methods in [SY] may be applied with only estimates of the local errors. The methods presented here do not provide rigorous bounds, but instead provide estimates using existing initial value software and local error estimates provided by the initial value problem (IVP) software. Our purpose is to derive numerical methods for obtaining a posteriori global error estimates that are compatible with existing numerical integration software.

In §2 we present a notion of piecewise hyperbolicity due to Pliss [Pl1] and present results that give sufficient conditions for the existence of a trajectory on the positive real line uniformly close to a discrete numerical approximation when the linear variational equation has this type of piecewise hyperbolicity. We show that there exists a trajectory nearby by showing that there exists a zero of a certain mapping, F. Under certain assumptions, Newton's method will converge to a zero of F given a numerically computed orbit as an initial guess. We do not actually perform Newton's method, but find a bound on the norm of a right inverse of the linearized function DF to prove the existence of a zero of F in a neighborhood of our initial guess. The main result of this section (Theorem 2.3) is essentially a shadowing lemma for the case in which the linear variational equation is piecewise hyperbolic in the sense of Pliss.

This includes the case in which the linear variational equation is exponentially dichotomic. In §3 we state a result for a numerically approximating orbit of finite length and give a simple proof in terms of quantities that are numerically computable. The result is based upon having a right inverse for an approximation of DF and a bound on the norm of this right inverse. The challenge to obtain accurate estimates of the global error is to minimize the norm of the right inverse over the set of all right inverses. Section 4 is devoted to developing numerical methods to estimate quantities necessary to apply the result in §3. Most of our efforts are in finding a suitable right inverse and in providing a bound or estimate on the norm of this right inverse. Two methods are developed. One is based on the choice of the pseudo or generalized inverse as our right inverse, and the second method is based on ideas related to the well conditioning of multipoint BVPs. These multipoint BVPs correspond to right inverses and, motivated by the concept of piecewise hyperbolicity the interior boundary conditions, are chosen to occur at points in which there is an increase in the number of stable modes. To provide estimates of the global error using existing numerical ordinary differential equation (ODE) software, all of our global error estimates are in terms of the supremum norm. Numerical examples are presented in §5. Our methods are applied to the logistic equation, the forced pendulum equation, and the space-discretized Chafee–Infante equation. Conclusions and references are presented in §§6 and 7, respectively.

2. Theoretical aspects. Throughout this paper we consider both sequences and continuous functions. We reserve the notation $\tilde{x}(t)$ for functions and the notation $x := \{x_n\}$ for sequences. Given a function $\tilde{x}(t)$ defined on some possibly infinite real interval and a sequence $\{t_n\}$ with values in this interval, we will write the restriction of $\tilde{x}(t)$ to $\{t_n\}$ as $x := \{x_n\}$ where $x_n := x(t_n)$ for all n. Unless otherwise stated $||\tilde{x}|| = \sup_t ||\tilde{x}(t)||$ and $||x|| = \sup_n ||x_n||$. In this section $||y||$ denotes the Euclidean norm for $y \in \mathbb{R}^N$.

Consider the initial value problem

$$\begin{aligned} \dot{\tilde{x}} &= f(\tilde{x}, t), \\ \tilde{x}(t_0) &= x_0, \end{aligned} \tag{2.1}$$

where $t_0 \in \mathbb{R}$, $\tilde{x}(t) \in \mathbb{R}^N$, $\dot{\tilde{x}} = \frac{d\tilde{x}}{dt}$, and $f \in C^k(\mathbb{R}^N, \mathbb{R}; \mathbb{R}^N)$ for some $k \geq 2$. Let $\phi : \mathbb{R}^N \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^N$ be the associated solution operator so that $\phi(x_0, t_0, t_0) = x_0$ and $\phi(x_0, t_0, t)$ is the solution at time t with initial condition x_0 at t_0. To solve this equation numerically, we consider one-step methods of the form

$$\begin{aligned} x_{n+1} &= x_n + h_n \Theta(f, x_n, t_n, h_n), \\ x_n &\quad \text{given} \end{aligned} \tag{2.2}$$

to advance the solution from t_n to $t_{n+1} := t_n + h_n$.

Given an orbit $x := \{x_n\}_0^\infty$ produced by a one-step method, we define a corresponding piecewise discontinuous function that is double valued at t_n, $n = 1, \ldots, \infty$, called a *pseudo solution* $\{\tilde{x}_n(t)\}_0^\infty$ as

$$\tilde{x}_n(t) = \phi(x_n, t_n, t) \quad \text{for } t_n \leq t \leq t_{n+1} \tag{2.3}$$

$n = 0, \ldots, \infty$ so that $\tilde{x}_n(t_n) = x_n$. Let $\delta_n = \tilde{x}_{n+1}(t_{n+1}) - \tilde{x}_n(t_{n+1})$, $n = 0, \ldots, \infty$ denote the *local error* at the nth iterate, and let $\tilde{x}(t) = \tilde{x}_n(t)$ for $t_n \leq t < t_{n+1}$ so that $\tilde{x}(t)$ is well defined.

Let $l^\infty(\mathbb{N})$ denote the sequences $w = \{w_n\}_0^\infty$ with $w_n \in \mathbb{R}^N$ for all n and $\sup_n ||w_n|| < \infty$. Consider the operator $F : l^\infty(\mathbb{N}) \to l^\infty(\mathbb{N})$, where the nth iterate $(F(w))_n$ is defined for any

$w \in l^\infty(\mathbb{N})$ to be

$$(F(w))_n = w_{n+1} - \phi(w_n, t_n, t_{n+1}) \quad \text{for } n = 0, \ldots, \infty \tag{2.4}$$

so that F measures the local error at each iterate. We wish to find a solution $w \in l^\infty(\mathbb{N})$ of $F(w) = 0$, i.e., a solution of the original IVP (2.1).

Consider the first variation $DF(x) : l^\infty(\mathbb{N}) \to l^\infty(\mathbb{N})$ of $F(x)$ defined by

$$(DF(x)u)_n = u_{n+1} - \phi_x(x_n, t_n, t_{n+1})u_n, \tag{2.5}$$

where $\phi_x := \partial\phi/\partial x_n$.

Given a pseudo solution $\tilde{x}(t)$, we construct (as in [Pa]) a corresponding continuous function $\tilde{z}(t)$ defined by

$$\tilde{z}(t) = \begin{cases} \tilde{x}(t) + (t_{n+1} - t_n)^{-1}(t - s_n)\delta_n, & t_n \le t \le s_n, \\ \tilde{x}(t) + (t_{n+1} - t_n)^{-1}(t - s_n)\delta_{n+1}, & s_n \le t \le t_{n+1}, \end{cases} \tag{2.6}$$

where $s_n = (t_{n+1} + t_n)/2$ so that $\delta = \sup_n \delta_n$ implies $||\tilde{x} - \tilde{z}|| \le \frac{\delta}{2}$.

Let Φ denote the principal matrix solution for the linear variational equation about $\tilde{z}(t)$ so that

$$\partial_t \Phi(t, \tau) = Df(\tilde{z}(t), t)\Phi(t, \tau), \qquad \Phi(\tau, \tau) = I \tag{2.7}$$

for $t \ge \tau$ where $Df := \frac{\partial f}{\partial \tilde{z}}$.

DEFINITION 1. *For positive constants K, λ, the system (2.7) is said to be (K, λ)-hyperbolic on the interval $[a, b]$ if for given $s \in [a, b]$ there exists linear subspaces $S(s)$ and $U(s)$ of dimension k and $N - k$, respectively, such that, if $y_0 \in S(s)$,*

$$||\Phi(t, a)\Phi^{-1}(s, a)y_0|| \le Ke^{-\lambda(t-s)}||y_0||$$

for $t \ge s$ and $s, t \in [a, b]$, and, if $y_0 \in U(s)$,

$$||\Phi(t, a)\Phi^{-1}(s, a)y_0|| \le Ke^{-\lambda(s-t)}||y_0||$$

for $s \ge t$ and $s, t \in [a, b]$.

This is identical to the definition of exponential dichotomy on an interval that may be found in [Co] and [AMR], where $S(s)$ and $U(s)$ represent the decaying and growing solution components, respectively.

Given subspaces L, M in $\mathbb{R}^N$ we say that these subspaces intersect transversally if

$$\dim L + \dim M = \dim(L \cap M) + N.$$

Define $\angle(L, M)$, the angle between subspaces L, M, where $0 \le \angle(L, M) \le \pi/2$ by

$$\cos(\angle(L, M)) = \max_{||y_1||=||y_2||=1} |y_1^T y_2|,$$

where $y_1 \in L$, $y_2 \in M$ and $y_i \perp L \cap M$ for $i = 1, 2$.

It is assumed that (2.7) satisfies the following three hypotheses.

(H1) There is a mesh made up of what we will call the switching points, $\beta_0 < \beta_1 < \cdots < \beta_m < \beta_{m+1}$, where $\beta_0 = t_0$ and $\beta_{m+1} = +\infty$ such that (2.7) is (K, λ)-hyperbolic with decaying/growing solution spaces $S_j(t)$ and $U_j(t)$ on each of the intervals $[\beta_j, \beta_{j+1}]$, for $j = 0, \ldots, m$.

(H2) The inequalities

$$\dim U_j(\beta_{j+1}) > \dim U_{j+1}(\beta_{j+1})$$

are satisfied for $j = 0, \ldots, m-1$, and the subspaces $U_j(\beta_{j+1})$ and $S_{j+1}(\beta_{j+1})$ intersect transversally for $j = 0, \ldots, m-1$.

(H3) There is an $\alpha > 0$ such that

$$\angle(U_j(\beta_{j+1}), S_{j+1}(\beta_{j+1})) > \alpha$$

for $j = 0, \ldots, m-1$.

Remarks.

(i) Note that (H2) implies that necessarily $m \leq N$; i.e, the number of switching points is less than or equal to the dimension of the problem.

(ii) It is shown in [Pl2] that (H1)–(H3) are satisfied for differential equations of the form (2.1) that are periodic in t, hyperbolic on the nonwandering set and satisfy the strict transversality condition (see [Ro1] and [Ro2]).

(iii) The switching points denote points in time where there is a decrease in the number of unstable modes.

Consider now the inhomogeneous linear equation

$$\dot{\tilde{u}}(t) = Df(\tilde{z}(t), t)\tilde{u}(t) + \tilde{g}(t). \tag{2.8}$$

The following theorem shows that (H1)–(H3) are sufficient to imply the existence of uniformly bounded solutions of (2.8).

THEOREM 2.1 ([Pl1]). *There exist constants $T(K, \lambda, \alpha)$ and $\eta(K, \lambda, \alpha)$ such that if* (2.7) *satisfies* (H1)–(H3) *for some switching points $\{\beta_j\}_0^{m+1}$, with*

$$\beta_{j+1} - \beta_j > T(K, \lambda, \alpha)$$

for $j = 1, \ldots, m-1$, then for any continuous function $\tilde{g}(t)$ with

$$||\tilde{g}(t)|| \leq \eta(K, \lambda, \alpha),$$

the system (2.8) *has a solution $\tilde{u}(t)$ satisfying*

$$||\tilde{u}(t)|| \leq 1 \quad \textit{for all } t.$$

The following theorem is an approximate implicit function theorem (basically Newton's method) that we use to find a zero of the function F defined in (2.4) given a sufficiently small local error and a bounded right inverse for $DF(x)$ defined in (2.5). Theorem 2.2 is easily generalized to the case of doubly infinite sequences and finite sequences without change in the proof.

THEOREM 2.2. *Let $F : l^\infty(\mathbb{N}) \to l^\infty(\mathbb{N})$ be a C^2 map. Let x be a point in $l^\infty(\mathbb{N})$ such that $DF(x)$ has a bounded right inverse $DF(x)^\dagger$ and let $\epsilon_0 > 0$ be chosen so that*

$$||DF(x) - DF(w)|| \leq 1/(2||DF(x)^\dagger||) \tag{2.9}$$

for $||w - x|| \leq \epsilon_0$. If $0 < \epsilon \leq \epsilon_0$ and

$$||F(x)|| \leq \epsilon/(2||DF(x)^\dagger||), \tag{2.10}$$

then the equation $F(w) = 0$ has a solution w such that $||w - x|| \leq \epsilon$.

Proof. For the proof see [CVV1]. □

The following theorem gives sufficient conditions for the existence of a trajectory near an orbit $x := \{x_n\}_0^\infty$ with $||F(x)|| \leq \delta$. The result is based upon the uniform boundedness result of Pliss (Theorem 2.1) and an application of Theorem 2.2.

THEOREM 2.3. *Consider the* IVP (2.1) *and assume that it is solved using a numerical method to produce an orbit* $x := \{x_n\}_0^\infty$ *with local error uniformly bounded by* $\delta > 0$. *Let* $\tilde{x}(t)$ *denote the corresponding pseudo solution and* $\tilde{z}(t)$ *the corresponding continuous function constructed as in* (2.6). *Let* $h_{\max} = \sup_n\{h_n\}$ *and* $h_{\min} = \inf_n\{h_n\}$ *denote the maximum and minimum stepsize, respectively. Let* $L_{Df}(\tilde{x}, \gamma)$ *denote a bound on the Lipschitz constant for* Df *in a* γ*-neighborhood of* $\tilde{x}(t)$ *and let* $B_{Df}(\tilde{x}, \gamma)$ *denote a bound on* Df *in a* γ*-neighborhood of* $\tilde{x}(t)$. *Assume that*

(i) *the linear system* (2.7) *satisfies* (H1)–(H3) *with* $\eta(K, \lambda, \alpha)$ *and* $T(K, \lambda, \alpha)$ *defined as in Theorem* 2.1 *with* $\beta_{j+1} - \beta_j > T(K, \lambda, \alpha)$ *for* $j = 1, \ldots, m-1$;

(ii) *the inequality* $c > r$ *is satisfied where* $r = h_{\max} L_{Df}(\tilde{x}, \delta/2)\delta/2 + h_{\max}^2 B_{Df}^2(\tilde{x}, \delta/2)/2$ *and* $c = \eta(K, \lambda, \alpha)h_{\min}(1-\rho)$ *for an arbitrary* ρ *such that* $1 \gg \rho > 0$;

(iii) *the inequality*

$$h_{\max} L_{Df}(\tilde{x}, \epsilon)\epsilon \leq \left(\frac{c+s}{c}\right)^{-1} \frac{c-r}{2}$$

is satisfied for $s := h_{\max}^2 B_{Df}(\tilde{x}, \delta/2)\eta(K, \lambda, \alpha)/2$ *and* $\epsilon := 2\delta(c-r)^{-1}$.

Then there exists a solution $\tilde{w}(t)$ *of the* IVP (2.1) *such that* $||\tilde{w}(t_n) - x_n|| \leq \epsilon$ *for* $n = 0, \ldots, \infty$.

Proof. Since (i) holds, the Pliss Theorem implies that the inhomogeneous equation (2.8) has a solution $\tilde{u}(t)$ satisfying $||\tilde{u}|| \leq 1$ for all continuous functions $\tilde{g}(t)$ such that $||\tilde{g}|| \leq \eta(K, \lambda, \alpha)$. If $g_n := -\int_{t_n}^{t_{n+1}} \tilde{g}(s)ds$ and $u_n := \tilde{u}(t_n)$, then

$$\begin{aligned} u_{n+1} &= u_n + \int_{t_n}^{t_{n+1}} Df(\tilde{z}(s), s)\tilde{u}(s)ds - g_n \\ &= \phi_x(x_n, t_n, t_{n+1})u_n - g_n + r_n, \end{aligned} \tag{2.11}$$

where ϕ_x is defined in (2.5) and

$$\begin{aligned} r_n &= \int_{t_n}^{t_{n+1}} [Df(\tilde{z}(s), s)\tilde{u}(s) - Df(\tilde{x}(s), s)u_n]ds \\ &= \int_{t_n}^{t_{n+1}} [Df(\tilde{z}(s), s) - Df(\tilde{x}(s), s)]u_n ds \\ &\quad + \int_{t_n}^{t_{n+1}} Df(\tilde{z}(s), s)\left[\int_{t_n}^{s} Df(\tilde{z}(\tau), \tau)\tilde{u}(\tau) + \tilde{g}(\tau)d\tau\right] ds. \end{aligned}$$

Then we have $||r_n|| \leq r||u_n|| + h_{\max}^2 B_{Df}(\tilde{x}, \delta/2)\eta(K, \lambda, \alpha)/2 \equiv r||u_n|| + s$.

Let $z_n = z(t_n)$ for $n = 0, \ldots, \infty$ so that by (2.11) we have

$$(DF(x)u)_n = (G(z)u)_n - r_n, \quad \text{for } n = 0, \ldots, \infty,$$

where the linear operator $G(z)$ is defined so that $G(z)u = g$ implies

$$(G(z)u)_n \equiv u_{n+1} - u_n - \int_{t_n}^{t_{n+1}} Df(\tilde{z}(s), s)\tilde{u}(s)ds = g_n \quad \text{for all } n.$$

Given a sequence $g := \{g_n\}_0^\infty$ with $||g_n|| \le c$ there exists a continuous function $\tilde{g}(t)$ such that $||\tilde{g}|| \le \eta(K, \lambda, \alpha)$. By the Pliss Theorem, $G(z)$ is onto; i.e., given any sequence $g \in l^\infty(\mathbb{N})$ there exists a sequence $u \in l^\infty(\mathbb{N})$ such that $G(z)u = g$. Therefore, $G(z)$ has a right inverse $G(z)^\dagger$ with

$$||G(z)^\dagger|| = \sup_{||g|| \le c} \frac{||G(z)^\dagger g||}{c} \le \sup_{||g|| \le c} \frac{\{||u|| : u = G(z)^\dagger g\}}{c} \le c^{-1}.$$

In general, if $u = \{u_n\}_0^\infty$ satisfies $(G(z)u)_n = g_n + r_n$, then $(DF(x)u)_n = g_n$ and $||u|| \le ||G(z)^\dagger||(c + r||u|| + s)$ so that (ii) implies $(1 - r/c)||u|| \le ||G(z)^\dagger||(c + s)$. Thus, if (ii) is satisfied,

$$\begin{aligned} ||DF(x)^\dagger|| &= \sup_{||g|| \le c} \frac{||DF(x)^\dagger g||}{c} \le \sup_{||g|| \le c} \frac{\{||u|| : u = DF(x)^\dagger g\}}{c} \\ &\le ||G(z)^\dagger|| \left(\frac{c+s}{c}\right)\left(\frac{1-r}{c}\right)^{-1} \le \left(\frac{c+s}{c}\right)(c-r)^{-1}. \end{aligned}$$

Now apply the Fixed Point Theorem to F with $||F(x)|| \le \delta$ and $||DF(x)^\dagger|| \le (\frac{c+s}{c})(c - r)^{-1}$. For $\epsilon_0 := \epsilon$ and $||x - w|| \le \epsilon_0$ and using (iii), we have (2.9) satisfied since

$$||DF(x) - DF(w)|| \le h_{\max} L_{Df}(\tilde{x}, \epsilon)||x - w|| \le \left(\frac{c+s}{c}\right)^{-1} \frac{c-r}{2} \le (2||DF(x)^\dagger||)^{-1}.$$

Thus, there exists a solution $w \in l^\infty(\mathbb{N})$ of $F(w) = 0$ such that $||w_n - x_n|| \le \epsilon$. Define $\tilde{w}(t) = \phi(w_n, t_n, t)$ for $t_n \le t < t_{n+1}$ and $n = 0, \ldots, \infty$ to complete the proof. □

3. Numerical aspects. We now consider the case in which we have produced a finite orbit using a numerical method. We would like to know whether there is a trajectory satisfying the same differential equation but with a nearby initial condition such that the trajectory is close to the numerically computed orbit at the mesh points. Our intent is to provide verifiable assumptions given certain numerically computable quantities so that we may apply the theorem and obtain an a posteriori bound on the global error.

In this section, let $||y||$ denote the supremum norm for $y \in \mathbb{R}^N$. Consider the IVP (2.1) and for some finite positive integer M consider the orbit $\{x_n\}_0^M$ produced using (2.2). Let $\tilde{x}_n(t)$ denote the pseudo solution defined as in (2.3) and let $\tilde{x}(t) := \tilde{x}_n(t)$ for $t_n \le t < t_{n+1}$. Let δ denote a bound on the local error. Consider the linearized problem about the pseudo solution

$$\dot{u} = Df(\tilde{x}(t), t)u. \tag{3.1}$$

Define the operator $F : l^\infty(\{0, \ldots, M\}) \to l^\infty(\{0, \ldots, M-1\})$ by

$$(F(x))_n = x_{n+1} - \phi(x_n, t_n, t_{n+1}) \tag{3.2}$$

and its first variation $DF(x) : l^\infty(\{0, \ldots, M\}) \to l^\infty(\{0, \ldots, M-1\})$ by

$$(DF(x)u)_n = u_{n+1} - \phi_x(x_n, t_n, t_{n+1})u_n, \tag{3.3}$$

where ϕ_x is defined as in (2.5).

Let $\tilde{z}(t)$ be a function with the property that $||\tilde{x}(t) - \tilde{z}(t)|| \leq \delta$ for all t. For one-step methods with an associated Taylor polynomial, an obvious choice for the function $\tilde{z}(t)$ is the interpolant defined locally by the Taylor polynomial corresponding to the numerical method. In particular, for the Runge–Kutta Felberg integrator RKF45 used for the examples in §5 we employ the associated fifth-order interpolant. The function $\tilde{z}(t)$ may be discontinuous at the mesh points. Consider the linearization about $\tilde{z}$,

$$\dot{u} = Df(\tilde{z}(t), t)u. \tag{3.4}$$

We define $G(z) : l^\infty(\{0, \ldots, M\}) \to l^\infty(\{0, \ldots, M-1\})$ by

$$(G(z)u)_n = u_{n+1} - \Phi(t_{n+1}, t_n)u_n, \tag{3.5}$$

where Φ is defined as in (2.7).

Let A_n denote a quadrature formula used to approximate $\Phi(t_{n+1}, t_n)$. Let s denote a bound on the relative error in the quadrature approximation, i.e.,

$$||\Phi(t_{n+1}, t_n)u_n - A_n u_n|| \leq s||u_n||. \tag{3.6}$$

Define the operator $H(A) : l^\infty(\{0, \ldots, M\}) \to l^\infty(\{0, \ldots, M-1\})$ by

$$(H(A)u)_n = u_{n+1} - A_n u_n. \tag{3.7}$$

We now state the following theorem similar to Theorem 2.3, but with assumptions that may be easily verified computationally.

THEOREM 3.1. *Consider the* IVP (2.1) *and assume that it is solved numerically producing an orbit* $x := \{x_n\}_0^M$ *with local error uniformly bounded by* $\delta > 0$. *Let* $\tilde{x}(t)$ *denote the corresponding pseudo solution and let* $\tilde{z}(t)$ *denote a function with the property that* $||\tilde{x}(t) - \tilde{z}(t)|| < \delta$ *for all* t. *Let* A_n *denote the quadrature formula used to approximate the linear variational equation about* $\tilde{z}(t)$ *from* t_n *to* t_{n+1}. *Let* $h_{\max} = \sup_n\{h_n\}$ *denote the maximum stepsize. Let* $L_{Df}(\tilde{x}, \gamma)$ *denote a bound on the Lipschitz constant for* Df *in an* γ*-neighborhood of* $\tilde{x}(t)$. *Assume that*

(i) *the inequality* $c > r + s$ *is satisfied where* $||H(A)^\dagger|| \leq c^{-1}$ *and* $H(A)$ *is defined in* (3.7), $H(A)^\dagger$ *is a right inverse of* $H(A)$, $r = h_{\max}L_{Df}(\tilde{x}, \delta)\delta$, *and* s *is defined in* (3.6);

(ii) *the inequality*

$$h_{\max}L_{Df}(\tilde{x}, \epsilon)\epsilon \leq (c - r - s)/2$$

is satisfied for $\epsilon = 2\delta(c - r - s)^{-1}$.

Then there exists a solution $\tilde{w}(t)$ *of the* IVP (2.1) *such that* $||\tilde{w}(t_n) - x_n|| \leq \epsilon$ *for* $n = 0, \ldots, M$.

Proof. Given a sequence $g := \{g_n\}_0^{M-1}$, we have that $u := \{u_n\}_0^M$ is a solution of $(DF(x)u)_n = g_n$ if $(H(A)u)_n = g_n + r_n + s_n$, where s_n is defined to be

$$s_n := [\Phi(t_{n+1}, t_n) - A_n]u_n$$

and

$$r_n := [\phi_x(x_n, t_n, t_{n+1}) - \Phi(t_{n+1}, t_n)]u_n = \int_{t_n}^{t_{n+1}} [Df(\tilde{x}(s), s) - Df(\tilde{z}(s), s)]u_n ds.$$

By (3.6) we have that $||s_n|| \leq s||u_n||$ and we have that

$$||r_n|| \leq h_{\max} L_{Df}(\tilde{x}, \delta)\delta||u_n||,$$

so that $||r_n|| \leq r||u_n||$. For $||g_n|| \leq 1$, we have

$$||u|| \leq ||H(A)^\dagger|| \sup_n \{||g_n|| + ||r_n|| + ||s_n||\} \leq c^{-1}(1 + r||u|| + s||u||)$$

so that by (i)

$$||u|| \leq (c - r - s)^{-1}.$$

Thus,

$$\begin{aligned} ||DF(x)^\dagger|| &= \sup_{||g||=1} ||DF(x)^\dagger g|| \leq \sup_{||g||=1} \{||u|| : u = DF(x)^\dagger g\} \\ &\leq (c - r - s)^{-1}. \end{aligned}$$

Now apply the Fixed Point Theorem to F with $||F(x)|| \leq \delta$ and $||DF(x)^\dagger|| \leq (c - r - s)^{-1}$. For $\epsilon_0 := \epsilon$ and $||x - w|| \leq \epsilon_0$ and using (ii), we have (2.9) satisfied since

$$||DF(x) - DF(w)|| \leq h_{\max} L_{Df}(\tilde{x}, \epsilon)||x - w|| \leq (c - r - s)/2 \leq (2||DF(x)^\dagger||)^{-1}.$$

Thus, there exists a solution $w \in l^\infty(\{0, \ldots, M\})$ of $F(w) = 0$ such that $||w_n - x_n|| \leq \epsilon$. Define $\tilde{w}(t) = \phi(w_n, t_n, t)$ for $t_n \leq t < t_{n+1}$ and $n = 0, \ldots, M - 1$ to complete the proof. □

Remarks. (i) Note that no explicit bounds on the inverses of f or Df or on higher-order derivatives are required.

(ii) The theorem may be applied to maps by simply using δ as a bound on the roundoff error and by setting $r = s = 0$.

4. Algorithms. In this section we present algorithms to estimate quantities needed to apply Theorem 3.1. To apply Theorem 3.1 we must supply a bound on a right inverse of $H(A)$. Since the norm of the right inverse of $H(A)$ measures to a large degree the magnification of the local error that gives the global error, it is advantageous to find, if possible, a right inverse of $H(A)$ that has small norm. Our philosophy is to use existing ODE solvers and other existing software to develop methods for obtaining an accurate estimate of the global error. As such, our error estimates will be in terms of the supremum norm, $||\cdot|| := ||\cdot||_\infty$, since most ODE solvers provide error estimates in this norm. We will use the absolute and relative local error tolerances that are provided by most standard solvers. Our intent is to provide practical estimates but not necessarily rigorous bounds on the global error. Most of our effort will be devoted to finding a suitable right inverse $H(A)^\dagger$ and to obtaining a bound or estimate of $||H(A)^\dagger||_\infty$.

An obvious choice for the right inverse is the pseudo or generalized inverse. In this case finding the right inverse is trivial, but estimating or bounding its norm may be difficult. If we consider $H(A)$ as a matrix and write $H(A)$ in terms of its singular value decomposition, then $H(A) = U\Sigma V^T$ where U, V are orthogonal and Σ is a nonnegative diagonal matrix. If $H(A)$ is full rank, then $||H(A)^\dagger||_2 = 1/\sigma_1$ where σ_1 is the smallest singular value of $H(A)$. The difficulty with using the pseudo inverse is that although the pseudo inverse is optimal in the 2-norm sense it is not necessarily optimal in the ∞-norm sense. In fact, in general, $||H(A)^\dagger||_\infty \leq \sqrt{NM}||H(A)^\dagger||_2$ for an ODE in $\mathbb{R}^N$ and an orbit of length M.

For explicit one-step methods, $H(A)$ has the matrix form

$$H(A) = \begin{pmatrix} -A_0 & I_N & & \\ & \ddots & \ddots & \\ & & -A_{M-1} & I_N \end{pmatrix}, \tag{4.1}$$

where $H(A)$ is an $M \cdot N \times (M+1) \cdot N$ matrix, A_i is an $N \times N$ matrix, and I_N is the $N \times N$ identity matrix. The matrix A_i advances the discrete solution of the linear variational equation from t_i to t_{i+1}. Note that $H(A)H(A)^T$ is a symmetric block tridiagonal matrix. We will restrict attention to explicit one-step methods, although similar results will apply for implicit one-step methods and multistep methods. The matrix $H(A)$ has the form of a multiple shooting matrix for a linear boundary value problem, but without the N additional rows that are used to specify the boundary conditions. The next approach will be to outline strategies for adding boundary conditions and thus specifying a right inverse for $H(A)$.

Our second approach to finding a right inverse involves appending boundary conditions at multiple points to obtain a well-conditioned BVP (see [dHM2] and [Ma2]). In this way we obtain a linear multipoint BVP. Our challenge is to find boundary conditions, possibly at more than the initial and terminal times, so that a BVP has a uniformly bounded solution. When appending boundary conditions we look for switching points to dynamically change the number of stable and unstable components. In particular, if we have k stable directions initially, then up to a suitable orthogonal change of variables (see [MS]), we adjoin the boundary condition

$$\begin{pmatrix} 0 & 0 \\ 0 & I_k \end{pmatrix} u(\beta_0) = 0,$$

where I_k is the $k \times k$ identity matrix. Similarly, if at the terminal time there are l unstable directions, then we adjoin the boundary condition

$$\begin{pmatrix} I_l & 0 \\ 0 & 0 \end{pmatrix} u(\beta_{m+1}) = 0.$$

The intermediate switching points β_j for $j = 1, \ldots, m$ produce boundary conditions of the form

$$\begin{pmatrix} 0_{N-k-j} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0_{k+j-1} \end{pmatrix} u(\beta_j) = 0.$$

These are linear boundary conditions so that appending boundary conditions is equivalent to adding N rows to $H(A)$ in (4.1).

To apply Theorem 3.1, we must have estimates of the following quantities:

1. δ, the local absolute error for the original problem (2.1);
2. $h_{\max}$, the maximum stepsize;
3. L_{Df}, a bound on the Lipschitz constant of Df;
4. c^{-1}, a bound on $||H(A)^\dagger||_\infty$;
5. s, the local relative error for the quadrature formula of the linearized problem.

Our basic algorithm is as follows.

ALGORITHM.

Step 1. Integrate simultaneously

$$\dot{x} = f(x,t),$$
$$\dot{u} = Df(x(t),t)u$$

from t_j to t_{j+1} with initial data x_j and $u_j = I$ for $j = 0, \ldots, M-1$, where $h_j = t_{j+1} - t_j$ is the stepsize chosen by the integrator with given absolute and relative error tolerances to obtain estimates for δ, s, and $h_{\max}$. Thus, we obtain the x_j and A_j for all j.

Step 2. Find a bound or estimate c^{-1} for $||H(A)^\dagger||_\infty$, where $H(A)^\dagger$ is either the pseudo inverse or a right inverse formed by adjoining boundary conditions.

Step 3. Compute a posteriori bounds for L_{Df}.

Step 4. If (i) and (ii) in Theorem 3.1 are satisfied, then apply the theorem to obtain the global error $\epsilon := 2\delta(c - s - r)^{-1}$.

Remarks. (i) By integrating both the original equation and the linear variational equation simultaneously, we obtain an approximation of the linear variational equation about the interpolant defined by the numerical method.

(ii) The local and relative error tolerances provide bounds for δ and s, respectively.

(iii) The a posteriori bounds for L_{Df} may be obtained as in [SY] using Gronwall's inequality or using coarse a priori bounds as we have done for the examples in §5.

(iv) The quantity r in Theorem 3.1 may be computed in terms of δ, $h_{\max}$, L_{Df}.

We now present the details of implementations for finding a right inverse and an ∞-norm bound or estimate on this right inverse in Step 2. The first method we consider is based on finding the smallest singular value of the matrix $H(A)$ in (4.1). The second method is based on adjoining boundary conditions and solving a suitable linear inhomogeneous BVP.

Instead of directly finding the smallest singular value σ_1 of the matrix $H(A)$, we will find the smallest eigenvalue λ_1 of the symmetric block tridiagonal matrix $H(A)H(A)^T$ and then set $\sigma_1 = \sqrt{\lambda_1}$. We find λ_1 by the Lanczos process (see [GvL]). The Lanczos process is an iterative method for finding the extremal eigenvalues of a matrix. The method generates a sequence of tridiagonal matrices whose eigenvalues are progressively better estimates of the extremal eigenvalues of the original matrix that is typically large and sparse. The convergence to the extremal eigenvalues is rapid provided the relative spacing between these eigenvalues is large (see [GvL]). We have made modifications to the software package LAS2 (see [Ber]) to apply the Lanczos procedure to symmetric block tridiagonal matrices $B := H(A)H(A)^T$, where $H(A)$ is of the form (4.1). We employ the error estimation procedure that is provided as part of the software. Although there is no guarantee that we have actually found the smallest eigenvalue, we were able to confirm that for smaller examples the Lanczos process did in fact provide accurate estimates of the smallest eigenvalue.

Remarks. (i) The use of the pseudo inverse has the advantage that information about the number of stable and unstable modes is not necessary.

(ii) The use of the Lanczos method to take advantage of the sparse structure of $H(A)$ may be implemented in a memory efficient or a time efficient manner by either recalculating the A_i, $i = 1, \ldots, M$ or by storing the A_i, respectively.

(iii) As was remarked above, we obtain a supremum norm estimate of $\sqrt{NM}\sigma_1^{-1}$ as an estimate of the norm of the pseudo inverse of $H(A)$.

The boundary value problem approach is based on considering the difference equation

$$u_{n+1} = A_n u_n + g_n \tag{4.2}$$

along with the appended boundary conditions where $g = \{g_n\}_0^M$ is an arbitrary sequence with $||g||_\infty = 1$. We have that

$$\begin{aligned} ||H(A)^\dagger||_\infty &= \sup_{||g||_\infty = 1} ||H(A)^\dagger g||_\infty \\ &= \sup_{||g||_\infty = 1} \{||u||_\infty : u \quad \text{satisfies} \quad (4.2)\}. \end{aligned} \tag{4.3}$$

Our task now is to replace the problem (4.2) for an arbitrary sequence g of norm one with a problem for a fixed sequence and obtain a bound on $||H(A)^{\dagger}||_{\infty}$. This is similar to the situation when one is attempting to estimate the norm of the inverse of a matrix for condition number estimation. We will replace the arbitrary sequence g with a sequence in which every element is of absolute value one.

Since our multipoint BVP may be thought as a several-coupled two-point BVP, it suffices to consider the difference equation

$$u_{n+1} = A_n u_n + g_n,$$
$$\begin{pmatrix} 0 & 0 \\ 0 & I_k \end{pmatrix} Q_0^T u_0 = \gamma_0, \begin{pmatrix} I_{N-k} & 0 \\ 0 & 0 \end{pmatrix} Q_M^T u_M = \gamma_M,$$

where for simplicity we let M denote the number of iterates between (possible intermediate) boundary points, and we let u_0 denote the value at the left boundary point. Here Q_0 is an appropriately chosen permutation matrix (see [MS], [dHM1]) and Q_M is an orthogonal matrix to be determined below.

To solve the BVP, we decouple using an orthogonal decoupling transformation. Using the modified Gram–Schmidt method (see [GvL]), we obtain the decomposition $Q_{n+1} R_n = A_n Q_n$ for $n = 0, .., M-1$, where R_n is upper triangular with positive diagonal elements and Q_{n+1} is orthogonal. Then the decoupling transformation is given by $v_n = Q_n^T u_n$ and the decoupled equation is

$$\begin{aligned} &v_{n+1} = R_n v_n + h_n, \\ &\begin{pmatrix} 0 & 0 \\ 0 & I_k \end{pmatrix} v_0 = \gamma_0, \qquad \begin{pmatrix} I_{N-k} & 0 \\ 0 & 0 \end{pmatrix} v_M = \gamma_M, \end{aligned} \tag{4.4}$$

where $h_n = Q_{n+1}^T g_n$. We note that for $l = 0$ or $l = M$ if l denotes the time of an initial or terminal boundary point, then γ_l is the vector of all zeros. If the left boundary point of our two-point BVP is an intermediate boundary point of the multipoint BVP, then the components of the vector γ_0 are given by

$$\gamma_0^{(j)} = \begin{cases} 0, j = 1, \ldots, N-k+1, \\ v_0^{(j)}, j = N-k+2, \ldots, N. \end{cases}$$

Similarly, if the right boundary point of our two-point BVP is an intermediate boundary point of the multipoint BVP, then the components of the vector γ_M are given by

$$\gamma_M^{(j)} = \begin{cases} v_M^{(j)}, j = 1, \ldots, N-k-1, \\ 0, j = N-k, \ldots, N. \end{cases}$$

We now write the recursion in block form by setting

$$v_n = \begin{pmatrix} v_n^{(1)} \\ v_n^{(2)} \end{pmatrix},$$

where $v_n^{(2)}$ is a k-vector and

$$R_n = \begin{pmatrix} R_n^{(11)} & R_n^{(12)} \\ 0 & R_n^{(22)} \end{pmatrix}.$$

Then for any integer $0 \le J \le M$ we have that

$$\begin{aligned} v_{n+J+1}^{(2)} = &\sum_{i=n}^{n+J-1} R_{n+J}^{(22)} \cdots R_{i+1}^{(22)} h_i^{(2)} \\ &+ h_{n+J} + R_{n+J}^{(22)} \cdots R_n^{(22)} v_n^{(2)} \end{aligned} \tag{4.5a}$$

and

$$v_n^{(1)} = \sum_{i=n}^{n+J-1} [R_i^{(11)} \dots R_n^{(11)}]^{-1} \{R_i^{(12)} v_i^{(2)} + h_i^{(1)}\} + [R_{n+J}^{(11)} \dots R_n^{(11)}]^{-1} \{v_{n+J+1}^{(1)} - h_{n+J}^{(1)} - R_{n+J}^{(12)} v_{n+J}^{(2)}\}. \tag{4.5b}$$

For a matrix C and a vector b, let $|C|$ denote the corresponding matrix and $|b|$ the corresponding vector whose elements consist of the absolute value of the elements of C and b, respectively. Let $\mathbf{1} := (1, \dots, 1)^T$ denote the vector with all elements equal to one. Consider now the sequence $w = \{w_n\}_0^M$ formed as follows:

$$w_{n+J+1}^{(2)} = \sum_{i=n}^{n+J-1} |R_{n+J}^{(22)} \dots R_{i+1}^{(22)}| \mathbf{1} + \mathbf{1} + |R_{n+J}^{(22)} \dots R_n^{(22)}| w_n^{(2)} \tag{4.6a}$$

and

$$w_n^{(1)} = \sum_{i=n}^{n+J-1} |[R_i^{(11)} \dots R_n^{(11)}]^{-1}| \{|R_i^{(12)}| w_i^{(2)} + \mathbf{1}\} + |[R_{n+J}^{(11)} \dots R_n^{(11)}]^{-1}| \{w_{n+J+1}^{(1)} + \mathbf{1} + |R_{n+J}^{(12)}| w_{n+J}^{(2)}\}. \tag{4.6b}$$

It is easy to see that we have the following lemma.

LEMMA 4.1. *If γ_0, γ_M in* (4.4) *are nonnegative vectors, then for v computed in* (4.5) *and w computed in* (4.6), *we have that $||v||_\infty \geq ||w||_\infty$ for all sequences h in* (4.5) *with $||h|| = 1$.*

As a consequence of this lemma, we have that $N||v||_\infty \geq ||H(A)^\dagger||_\infty$ since the supremum norm condition number of an $N \times N$ orthogonal matrix is bounded by N.

Remarks. (i) To determine the switching points (times where there is a change in stability), we monitor the diagonal elements of the upper triangular matrices R_n. If $|R_{m-1}^{(j,j)}| > 1$ and $|R_m^{(j,j)}| < 1$ for some j, $1 \leq j \leq N$, then the mth iterate is a candidate to be a switching point. Since the number of unstable modes cannot be increased, we only decrease the number of unstable modes when the magnitude of the diagonal element of R_n has magnitude less than one over several iterates.

(ii) We compute the sequence w in (4.5) to account for roundoff errors using the methods in [Wi].

(iii) Other decoupling transformations are possible besides discrete orthogonal decoupling transformation. In particular, the Riccati transformation (see [DOR1], [DOR2], [Me1]) allows us to integrate a subset of the N^2 variables in the linear variational equation and may be better conditioned in the supremum norm. Another choice for a decoupling transformation is the continuous orthogonal transformation or continuous orthonormalization [D], [Me2].

5. Numerical examples. In this section we apply the algorithms for estimating the global error to three example problems. All computations were performed on a Silicon Graphics workstation with 64 megabytes of memory in double-precision arithmetic (machine epsilon $\approx 2.2\text{E}-16$). All computations were done using the Runge–Kutta Fehlberg integrator RKF45 of Shampine and Watts [SWD]. We have chosen RKF45 for convenience since it is a widely used automatic integrator with absolute and relative error tolerances. In Tables 1 through 6 we use T or Time to denote the final value of the independent variable t, Iterates to denote the number of timesteps that were taken, and CPU Time is the CPU time recorded in seconds. For the BVP method c^{-1} is a bound on the infinity-norm of a right inverse, and for the singular

value decomposition (SVD) method c^{-1} is an estimate of the infinity-norm of the pseudo inverse. We use δ to denote the local error tolerance and ϵ is our global error estimate.

Example 5.1. The first problem we consider is the logistic equation

$$\dot{y} = y(1-y), \qquad y(0) = \xi, 1 \gg \xi > 0,$$

which was also considered in [Be1].

We let the switching point β_1 be the time t_n such that $y(t_n) \approx \frac{1}{2}$, where $n \in \{0, \ldots, M\}$. This choice for the switching point is consistent with our theoretical results. In fact, for the exact solution $x(t)$ of the logistic equation with $x(0) = \frac{1}{2}$, the fundamental matrix solution about $x(t)$ is (K, λ)-hyperbolic for $K = 4$ and $\lambda = 1$ with $S(t) = \mathbb{R}$ for $t \in [0, +\infty)$ and $U(t) = \mathbb{R}$ for $t \in (-\infty, 0]$. When employing the boundary value method as outlined in §4, we adjoin the condition $u(\beta_1) = 0$ to the linear variational equation.

For this problem we have $L_{Df} = 2$. Tables 1 and 2 show our results for approximate orbits between the fixed points $x = 0$ and $x = 1$. For the SVD method presented in §4 we have included the 2-norm bound for $||H(A)^{\dagger}||$ in parentheses. We compute our numerical orbit with initial data $y(0) = \xi \approx 0$ and numerically integrate from $t = 0$ to $t = T$ so that $y(T) \approx 1 - \xi$. In Table 1, $\xi = 1.E-2$ and $y(T) \approx 1 - \xi$, while in Table 2, $\xi = 1.E-4$ and $y(T) \approx 1 - \xi$.

TABLE 1

Example 5.1. ($T = 9.22$, Iterates $= 188$, $\delta = 1.E-07$)			
Method	CPU time	c^{-1}	ϵ
BVP	.04	24.31	4.86E–6
SVD	.61	309.71 (22.6)	6.20E–5

TABLE 2

Example 5.1. ($T = 18.46$, Iterates $= 390$, $\delta = 1.E-07$)			
Method	CPU time	c^{-1}	ϵ
BVP	.08	24.28	4.86E–6
SVD	1.26	459.15 (23.25)	9.20E–5

For the SVD method we attained convergence in less than 100 Lanczos iterations.

Example 5.2. The next example we consider is the forced pendulum equation

$$\ddot{y} + a\dot{y} + \sin y = b\cos t, \qquad \dot{y}(0) = y(0) = 0,$$

where $a = 0.2$ and $b = 2.4$. This equation was considered in [SY]. For this equation, we found that the optimal choice was to maintain one stable and one unstable mode throughout, although the linear variational equation is not uniformly hyperbolic. There are changes in stability, but there is not a monotone decrease in the number of unstable modes. Thus, we adjoin boundary conditions to obtain a two-point boundary value problem such that in the decoupled variables v, we have the boundary conditions

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} v(\beta_0) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} v(\beta_1) = 0$$

when we integrate from $t = \beta_0 \equiv 0$ to $t = \beta_1 \equiv$ Time, where the values of Time are given in Tables 4 and 5.

For this problem we have $L_{Df} = 1$.

In our computations using the SVD method we attained convergence in the smallest eigenvalue of $H(A)H(A)^T$ to machine precision in less than 20 Lanczos iterations. For both the BVP and SVD method we were not able to compute for longer orbits due to memory

TABLE 3

Example 5.2. BVP method					
δ	Iterates	Time	CPU time	c^{-1}	ϵ
1.E–11	150,000	922	75	848,974	3.40E–05
1.E–12	300,000	1165	137	857,602	3.44E–06
1.E–12	500,000	1942	251	896,583	3.40E–06

TABLE 4

Example 5.2. SVD method					
δ	Iterates	Time	CPU time	c^{-1}	ϵ
1.E–11	150,000	922	472	808	1.62E–08
1.E–08	150,000	3677	469	802	1.61E–05
1.E-06	150,000	9229	634	799	1.60E–03

constraints, although less memory intensive implementations for both methods are possible by storing intermediate values of the orbit and then recomputing portions of the sequence $\{A_n\}_0^{M-1}$ as needed.

Although our methods serve different purposes than those considered in [SY] we now compare our results with the results obtained in [SY]. They obtain rigorous local error bounds through the error term of a fixed order, fixed stepsize Taylor series method. As such they obtain global error bounds. For the forced pendulum equation, Sauer and Yorke in [SY] were able to prove the existence of a trajectory within 1.E -9, a computer-generated orbit obtained using a seventh-order Taylor series method with a fixed step-size of $\Delta t \approx 3.\text{E} - 3$ to obtain local errors bounded by $1.\text{E} - 18$ for $0 \le t \le\approx 30,000$ on a machine with machine precision of $1.\text{E} - 28$. Our computations are performed using a standard fixed order, variable stepsize method, RKF45, in which we provide the local error tolerances and the integrator chooses the stepsize. In this way we obtain local error estimates, but not rigorous bounds on the local errors. For the BVP and the SVD methods presented in §4, our global error estimates are not rigorous bounds due to the local error estimation. Our use of the Lanczos method to estimate the smallest singular value of the matrix $H(A)$ defined in (4.1) seems to provide a reliable estimate for the norm of the pseudo inverse. For the forced pendulum equation, with the same parameter values and initial condition reported on in [SY], we were able to obtain a global error estimate of approximately $1.\text{E} - 3$ with a local error estimate of $1.\text{E} - 6$ for a trajectory of length $0 \le t \le\approx 10,000$ on a machine with machine precision approximately $1.\text{E} - 16$.

Example 5.3. The final example we consider is the space discretized Chafee–Infante equation with Neumann boundary conditions (see [Ch]). The Chafee–Infante equation is given by

$$\begin{aligned} v_t &= \xi^2 v_{xx} + f(v), \\ v_x(0) &= v_x(1) = 0, \\ v(x, t = 0) &\quad \text{given}, \end{aligned}$$

where $f(v) = v - v^3$. We consider the system of ODEs that is obtained after the above equation is discretized in its spatial variable. In particular, we consider the finite difference discretization

$$\begin{aligned} \dot{v}_i &= \left(\frac{\xi}{k}\right)^2 [v_{i+1} - 2v_i + v_{i-1}] + f(v_i), i = 1, \ldots, N, \\ v_0 &= v_1, v_{N+1} = v_N, \\ v_i(0) &\quad \text{given}, \end{aligned}$$

where $k = 1/(N-1)$.

TABLE 5

Example 5.3. BVP method					
δ	Iterates	Time	CPU time	c^{-1}	ϵ
1.E–10	5,000	6.34	451	57,698	1.15E–05
1.E–10	10,000	13.29	1490	259,811	5.20E–05
1.E–10	20,000	27.91	2220	492,733	9.86E–05

TABLE 6

Example 5.3. SVD method					
δ	Iterates	Time	CPU time	c^{-1}	ϵ
1.E–10	5,000	6.34	922	577	1.15E–07
1.E–08	5,000	15.31	3677	593	1.14E–05
1.E–06	5,000	36.51	9229	581	1.16E–03

The Chafee–Infante equation is a gradient system and its attracting set consists of the equilibrium solutions and the connections between the equilibrium solutions (see [H]). It is known (see [Ma1]) that the number of monotone pieces of the solution v, or lap number, is nonincreasing as a function of time. It is also known (see [BF]) that the dimension of the unstable subspace of an equilibrium solution is equal to the lap number of the equilibrium solution. We found that for a sufficiently fine discretization in space and sufficiently close to the attracting set, the number of unstable modes is nonincreasing along solution paths. It has recently been shown in [AD] and [LS] that for various discretizations of semilinear parabolic equations, the stable and unstable manifolds of the discretized problem converge to those of the continuous problem (see also [HLR]).

We set $N = 30$ and $\xi = 10^{-1}$ in our experiments and use the values $L_{Df} = 6$. We use the initial data $v_i(0) = \cos(3(i-1)\pi/(N-1))$ and monitor the eigenvalues of the matrix R_n to determine when we have attained the maximum number of unstable modes (during the initial transient there was an increase in the number of unstable modes). For all the experiments with the BVP method and for the SVD method with $\delta = 1.\mathrm{E}-10$, we only provide an error bound for the portion of the trajectory in which the number of unstable modes is nonincreasing. For the other examples in which the SVD method was used we include the initial transient phase. For all of our computations the number of unstable modes decreased from three to one.

In our computations with the BVP method we used $J = 0$ in (4.6(a), (4.6(b)). Somewhat better results were obtained using a larger value of J. For the SVD method, we obtained convergence to machine precision of the smallest eigenvalue of $H(A)H(A)^T$ within 20 Lanczos iterations.

6. Conclusions. In this paper we have shown that for a wide class of piecewise hyperbolic initial value ODEs, the global error in computing a discrete numerical approximation of a trajectory may be obtained as a reasonable magnification of the local error provided that we allow the true trajectory and the discrete approximation to have different initial conditions. The type of piecewise hyperbolicity we consider occurs in the case of several hyperbolic fixed points and certain space discretized parabolic partial differential equations.

From our numerical experiments it seems clear that although the SVD method was more expensive than the BVP method, we were able to obtain global error estimates with the SVD method for much smaller local error tolerances and longer time intervals. The SVD method gave better results than the BVP method but was more expensive in terms of both memory and time. It would be interesting to see if a more efficient Lanczos method could be developed specifically for the types of problems obtained when performing global error analysis. The cost of numerically integrating the linear variational equation may be decreased in the BVP

case by employing the Riccati transformation as a decoupling transformation in the case where there are very well-defined changes in the number of stable modes.

Our methods for providing global errors are not dependent on the particular integration method, although, in this paper, we restricted our attention to explicit one-step methods. In principle, any numerical integration scheme may be used, including implicit one-step methods and linear multistep methods. The amount of modification necessary depends on the particular implementation that one wishes to use. In order to use LSODE, for example, one would have to update the Nordsieck array before each step.

An interesting case for which we were not able to obtain good results is the case near a periodic orbit. This has been explored in [Be2], [E1], and [E2]. In particular, for the case in which $A_i = 1$ for all i, it is easy to see that the global error will grow linearly as a function of the length M of the orbit. This is due to the absence of hyperbolicity in solutions of systems of this type. Near a periodic orbit one does not expect to have hyperbolicity in the direction of the flow. It would be interesting to see if our methods could be applied to periodic systems to obtain global error estimates for those directions that are not in the direction of the flow.

Acknowledgments. We are grateful to Luca Dieci, Timo Eirola, Bob Russell, and the referees for helpful remarks on an earlier version of this paper.

REFERENCES

[AD] F. ALOUGES AND A. DEBUSSCHE, *On the qualitative behavior of the orbits of a parabolic partial differential equation and its discretization in the neighborhood of a hyperbolic fixed point*, Numer. Funct. Anal. Optim., 12 (1991), pp. 253–269.

[A] D. V. ANOSOV, *Geodesic Flows on Closed Riemannian Manifolds of Negative Curvature*, Trudy Math. Inst. Steklov, 90 (1967). (In Russian.)

[AMR] U. M. ASCHER, R. M. M. MATTHEIJ, AND R. D. RUSSELL, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[BF] P. W. BATES AND P. C. FIFE, *Spectral comparison Principles for the Cahn–Hilliard and phase-field equations, and time scales for coarsening*, Phys. D, 43 (1990), pp. 335–348.

[Ber] M. W. BERRY, *SVDPACK: A Fortran-77 Software Library for the Sparse Singular Value Decomposition*, preprint.

[Be1] W.-J. BEYN, *On the numerical approximation of phase portraits near stationary points*, SIAM J. Numer. Anal., 24 (1987), pp. 1095–1113.

[Be2] ———, *On Invariant Closed Curves for One-Step Methods*, Numer. Math., 51 (1987), pp. 103–122.

[Bo] R. BOWEN, *ω-limit sets for Axiom A diffeomorphisms*, J. Differential Equations, 18 (1975), pp. 333–339.

[Ch] N. CHAFEE, *Asymptotic behavior for solutions of a one-dimensional parabolic equation with homogeneous Neumann boundary conditions*, J. Differential Equeations, 18 (1975), pp. 111–134.

[CLP] S. N. CHOW, X. B. LIN, AND K. J. PALMER, *A shadowing lemma with applications to semilinear parabolic equations*, SIAM J. Math. Anal., 20 (1989), pp. 547–557.

[CP1] S. N. CHOW AND K. J. PALMER, *On the numerical computation of orbits of dynamical systems: the one-dimensional case*, Dynam. Differential Equations, 3 (1991), pp. 361–380.

[CP2] S. N. CHOW AND K. J. PALMER, *On the numerical computation of orbits of dynamical systems: the higher dimensional case*, J. Complexity, 8 (1992), pp. 398–423.

[CVV1] S. N. CHOW and E. S. VAN VLECK, *A shadowing lemma for random diffeomorphisms*, Random Comput. Dynam., 1 (1992), pp. 197–218.

[CVV2] ———, *Shadowing of Lattice Maps*, manuscript.

[Co] W. A. COPPEL, *Dichotomies in Stability Theory*, *Lecture Notes in Mathematics* 629, Springer-Verlag, New York, 1978.

[D] A. DAVEY, *An automatic orthonormalization method for solving stiff* BVPs, J. Comput. Phys., 51 (1983), pp. 343–356.

[dHM1] F. R. DE HOOG AND R. M. M. MATTHEIJ, *An algorithm for solving multi-point boundary value problems*, Computing, 38 (1987), pp. 219–234.

[dHM2] ———, *On the conditioning of multipoint and integral boundary value problems*, SIAM J. Math. Anal., 20 (1989), pp. 200–214.

[DOR1] L. DIECI, M. R. OSBORNE, AND R. D. RUSSELL, *A Riccati transformation method for solving linear* BVPs. I: *Theoretical aspects*, SIAM J. Numer. Anal., 25 (1988), pp. 1055–1073.

[DOR2] ———, *A Riccati transformation method for solving linear* BVPs. II: *Computational aspects*, SIAM J. Numer. Anal., 25 (1988), pp. 1074–1092.

[E1] T. EIROLA, *Invariant curves of one-step methods*, BIT, 28 (1988), pp. 113–122.

[E2] ———, *Aspects of backward error analysis in numerical* ODEs, J. Comput. Appl. Math., 45 (1993), pp. 65–74.

[Ge] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[GvL] G. GOLUB AND C. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1983.

[HYG1] S. HAMMEL, J. A. YORKE, AND C. GREBOGI, *Do numerical orbits of chaotic dynamical processes represent true orbits?*, J. Complexity, 3 (1987), pp. 136–145.

[HYG2] ———, *Numerical orbits of chaotic processes represent true orbits*, Bull. Amer. Math. Soc., 19 (1988), pp. 465–470.

[HLR] J. K. HALE, X.-B. LIN, AND G. RAUGEL, *Upper semicontinuity of attracts of semigroups and partial differential equations*, Math. Comput., 50 (1988), pp. 89–123.

[H] D. HENRY, *Geometric Theory of Semilinear Parabolic Equations*, Lecture Notes in Mathematics 840, Springer-Verlag, New York, 1981.

[LS] S. LARSSON AND J.-M. SANZ-SERNA, *The Behavior of Finite Element Solutions of Semilinear Parabolic Problems Near Stationary Points*, preprint.

[Ma1] H. MATANO, *Nonincrease of the lap-number of a solution for a one-dimensional semilinear parabolic equation*, J. Fac. Sci. Univ of Tokyo IA Math., 29 (1982), pp. 401–441.

[Ma2] R. M. M. MATTHEIJ, *Decoupling and stability of algorithms for boundary value problems*, SIAM Rev., 27 (1985), pp. 1–44.

[MS] R. M. M. MATTHEIJ AND G. W. M. STAARINK, *An efficient algorithm for solving general linear two-point* BVP, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 745–763.

[Me1] G. H. MEYER, *Initial Value Methods for Boundary Value Problems*, Academic Press, New York, 1973.

[Me2] ———, *Continuous orthonormalization for boundary value problems*, J. Comput. Phys., 62 (1986), pp. 248–266.

[Pa] K. J. PALMER, *Exponential dichotomies and transversal homoclinic points*, J. Differential Equations, 55 (1984), pp. 225–256.

[Pl1] V. A. PLISS, *Uniformly bounded solutions of linear systems of differential equations*, Differential Equations, 13 (1977), pp. 607–613.

[Pl2] ———, *Relationship between different conditions for structural stability*, Differential Equations, 17 (1981), pp. 545–550.

[Ro1] J. W. ROBBIN, *A structural stability theorem*, Ann. Math., 94 (1971), pp. 447–493.

[Ro2] R. C. ROBINSON, *Structural stability of C^1-diffeomorphisms*, J. Differential Equations, 22 (1976), pp. 28–73.

[SY] T. SAUER AND J. A. YORKE, *Rigorous verification of trajectories for the computer simulation of dynamical systems*, Nonlinearity, 4 (1991), pp. 961–979.

[SWD] L. F. SHAMPINE, H. A. WATTS, AND S. M. DAVENPORT, *Solving non-stiff* ODEs—*the state of the art*, SIAM Rev., 18 (1976), pp. 376–411.

[S1] H. J. STETTER, *Analysis of Discretization Methods for Ordinary Differential Equations*, Springer-Verlag, New York, 1973.

[S2] ———, *The defect correction principle and discretization methods*, Numer. Math., 29 (1978), pp. 425–443.

[Wi] J. H. WILKINSON, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 977–996, July 1994

012

NUMERICAL SOLUTION OF FIRST PASSAGE PROBLEMS IN RANDOM VIBRATIONS*

HANS PETTER LANGTANGEN†

Abstract. The reliability of dynamic systems modeled by white-noise-excited, stochastic, ordinary differential equations can be computed from deterministic backward Kolmogorov equations. This paper discusses and compares some numerical solution methods for boundary value problems involving backward Kolmogorov equations. The numerical examples concern single degree-of-freedom oscillators subjected to white noise and filtered white noise excitation. The efficiency of various methods and the sensitivity of the solution to the choice of the numerical parameters are particularly discussed.

Key words. stochastic differential equations, stochastic processes, finite element methods, random vibrations, first passage problems

AMS subject classifications. 60J70, 65C05, 65M60

1. Introduction. A frequently applied design criterion for dynamic systems subjected to uncertain input data is to require some response parameters to stay within a prescribed safe region, for a time interval, with a sufficiently high probability. In structural engineering the displacement of the system is often used as a critical response parameter. If the excitation is a stochastic process the displacement will be likewise and failure may occur when the displacement process exits the safe region. The time T to the first exit of the safe region is commonly called the first passage time of the displacement process and constitutes a measure of the lifetime of the structure. The reliability of the structure is then dependent upon the first passage time statistics. Exact analytical expressions for the statistics of T are unfortunately not available even for the simplest dynamic system of engineering interest. Hence, approximation methods are required. The purpose of this paper is to present an improved numerical method for accurately computing the distribution or the moments of T and to evaluate the behavior of this method.

Methods for calculating first passage time statistics of the single degree-of-freedom linear oscillator subjected to white noise excitation have been discussed by Crandall [8]. A particular class of methods that can yield exact statistics for T is based on Markov process theory and commonly called diffusion methods. A recent review of their application to first passage problems has been given by Roberts [17]. One of the particularly attractive features of diffusion methods is that mechanical nonlinearities and non-Gaussian excitation present, in principle, no difficulties. Diffusion methods result in linear, second order, partial differential equations, frequently in high dimensions, for the distribution or moments of T. The equations are usually defined on infinite domains.

Two- and three-dimensional versions of the partial differential equations associated with diffusion methods have been solved by various techniques. For example, Toland and Yang [20] used a random walk model, Sun and Hsu [19] worked with a cell-method, Langley [15] utilized a variational approach with Hermite polynomial expansions and Bergman, Spencer, and their coworkers [1], [2], [3], [18] employed a fairly general finite element solution method. The present article follows the latter type of numerical approach. The method and its implementation are extended to an arbitrary number of space dimensions and parts of the solution method are significantly improved with respect to computational efficiency and storage requirements. The main object of this paper is however to report the behavior of different

*Received by the editors September 16, 1991; accepted for publication (in revised form) March 12, 1993.

†Mechanics Division, Department of Mathematics, University of Oslo, P.O. Box 1053 Blindern, 0316 Oslo 3, Norway, and SINTEF SI, Department of Industrial Mathematics, P.O. Box 124 Blindern, 0314 Oslo 3, Norway (hpl@math.uio.no).

numerical strategies in some model problems. In particular, we demonstrate the influence of various numerical parameters on the accuracy. Two of the numerical examples concern filtered white noise excitation. To the author's knowledge, these first passage problems have not been solved previously in the literature by general diffusion methods.

2. Problem description. Suppose the stochastic dynamic system can be modeled by a system of first-order, ordinary, stochastic differential equations where the only stochastic excitation is of a white noise type:

$$\frac{d}{dt}X_i(t) = a_i + \sum_{j=1}^{d} B_{ij}N_j(t), \qquad i = 1, \ldots, d. \tag{1}$$

Here $\mathbf{X}(t) = (X_1(t), \ldots, X_d(t))^T$ is the response vector of the system, a_i and B_{ij} are functions of $X_1, \ldots, X_d$. Moreover, $N_i(t)$ is a white noise process defined as the generalized derivative of a normalized Wiener process [13] with $\mathrm{E}[N_i(t)] = 0$ and $\mathrm{E}[N_i(t+\tau)N_i(t)] = \delta(\tau)$, where δ is the Dirac delta function and $\mathrm{E}[\cdot]$ is the expectation operator. All the $N_i(t)$ processes are assumed to be independent. One can show that $\mathbf{X}(t)$ governed by (1) is a vector Markov process [13].

Let $\Omega_{\mathbf{x}}$ be a safe region for $\mathbf{X}(t)$. Engineering applications are frequently concerned with determining the probability that $\mathbf{X}(t) \in \Omega_{\mathbf{x}}$ for some time interval $[t_0, t]$. For example, a failure criterion may be formulated as $\mathbf{X} \notin \Omega_{\mathbf{x}}$. The time T to first exit from $\Omega_{\mathbf{x}}$, conditional on $\mathbf{X}(t_0) = \mathbf{y} \in \Omega_{\mathbf{x}}$, is commonly called the first passage time of the process $\mathbf{X}(t)$. Of particular interest is the reliability function $R(t \mid \mathbf{y}) \equiv \Pr\{T > t \mid \mathbf{X}(t_0) = \mathbf{y}\}$, with $\mathbf{y} = (y_1, \ldots, y_d)^T$. It can be shown that R is governed by the backward Kolmogorov equation [6], [13], [18]

$$\frac{\partial R}{\partial t} = \sum_{r=1}^{d}\left\{a_r\frac{\partial R}{\partial y_r} + \frac{1}{2}\sum_{s=1}^{d} C_{rs}\frac{\partial^2 R}{\partial y_r \partial y_s}\right\}, \qquad \mathbf{y} \in \Omega_{\mathbf{x}} \subset \Omega. \tag{2}$$

The coefficients C_{rs} are given as $C_{rs} = \sum_{i=1}^{d} B_{ri}B_{si}$. The functional form of a_r and B_{rs} are as defined in (1), but in equation (2), X_i must be replaced by y_i in the expressions for a_r and C_{rs}. The initial condition reads $R(t_0 \mid \mathbf{y}) = 1$.

In many practical applications the matrix B_{rs} contains mostly zeros, and the boundary conditions must then be prescribed with care to achieve a well-posed problem. Fichera [11] has studied the wellposedness of the present problem and the main results relevant for our equation (2), when the eigenvalues of C_{rs} are nonnegative, are as follows. Divide the boundary $\partial\Omega_{\mathbf{x}}$ of $\Omega_{\mathbf{x}}$ into three nonoverlapping parts, Γ_1, Γ_2, and Γ_3 defined by

$$\Gamma_1 = \left\{\mathbf{y} \in \partial\Omega_{\mathbf{x}};\ \sum_{i=1}^{d}\sum_{j=1}^{d}\chi_i C_{ij}\chi_j \neq 0\right\}, \tag{3}$$

$$\Gamma_2 = \left\{\mathbf{y} \in \partial\Omega_{\mathbf{x}};\ \sum_{i=1}^{d}\sum_{j=1}^{d}\left(a_i - \frac{\partial C_{ij}}{\partial y_j}\right)\chi_i > 0\right\}, \tag{4}$$

$$\Gamma_3 = \partial\Omega_{\mathbf{x}}\backslash(\Gamma_1 \cup \Gamma_2). \tag{5}$$

Here χ_i is the ith component of the outward unit normal to $\Omega_{\mathbf{x}}$. The solution R can be prescribed on $\Gamma_1 \cup \Gamma_2$ while no conditions should be assigned on Γ_3. The restrictions on the boundary conditions are a mathematical requirement for the problem to be well posed, but in simpler cases a physical interpretation can be given; see §5.1 for an example.

One can easily show that the moments M_k of T fulfill the set of recursive equations [6], [13], [18]

$$-kM_{k-1} = \sum_{r=1}^{d} \left\{ a_r \frac{\partial M_k}{\partial y_r} + \frac{1}{2} \sum_{s=1}^{d} C_{rs} \frac{\partial^2 M_k}{\partial y_r \partial y_s} \right\}, \quad \mathbf{y} \in \Omega_{\mathbf{x}}, \quad k = 1, 2, 3, \ldots \tag{6}$$

with

$$M_k(\mathbf{y}) \equiv -\int_{t_0}^{\infty} (t - t_0)^k \frac{\partial}{\partial t} R(t \mid \mathbf{y}) dt.$$

The boundary conditions are the same as for the reliability function R. Equation (6) is commonly referred to as the generalized Pontriagin–Vitt equation.

The applications of the above general theory to be exploited in this paper concern a single degree-of-freedom dynamic system governed by the equation of motion

$$\ddot{x} + r(x) + c(x, \dot{x}) = f(t), \tag{7}$$

where $x(t)$ is the displacement of the system, $r(x)$ represents restoring forces, $c(x, \dot{x})$ models damping forces, and $f(t)$ is a prescribed, stochastic excitation process. Engineering applications of the stochastic differential equation (7) arise in, for example, reliability analysis of structures subject to wind, current, wave, or earthquake loads. If $f(t)$ cannot be adequately modeled as a white noise process, auxiliary variables and equations are needed to filter white noise to the desired excitation process to achieve a system on the form (1). Examples concerning such auxiliary equations are given in §5.2.

No exact analytical solutions for R are known for dynamic systems of engineering interest. Although the backward Kolmogorov equation looks similar to transport equations in fluid dynamics, for which efficient numerical schemes are available, the backward equation has several features that make great demands to the numerical methods. For example, one must deal with infinite domains and singularities. Franklin and Rodemich [12] managed to derive an analytical solution for $M_1(\mathbf{y})$, in case of a white noise excited free particle, which showed that $M_1(\mathbf{y}) \notin H^1$, where H^1 is the relevant Hilbert space of functions with square integrable zeroth and first-order derivatives. Therefore one cannot expect the solution of the boundary value problems for R and M_k to lie in H^1. Nevertheless, the finite element approximations in use are confined to H^1 (see [1] for a comment on this issue).

Equation (2), and particularly the stationary version (6), have been solved by a standard Petrov–Galerkin finite element method by Bergman, Spencer, and their coworkers for a wide range of problems where $d = 2$ [3], [4], [10]. A white-noise-excited, hysteretic, single degree-of-freedom system was formulated and solved as a $d = 3$ first passage problem by Spencer [18].

The solution method to be applied in this paper is also based on a Petrov–Galerkin finite element formulation. However, we compare two different classes of weighting functions in combination with a general mesh grading strategy. Furthermore, we investigate both implicit and explicit time integration. The mathematical formulation of the numerical methods and their computer implementation are valid for any value of d. To limit the content of the paper, only numerical examples with $d \leq 3$ are presented.

The main deficiency of the solution approach used by Bergman, Spencer, and their coworkers is the long execution time and the large storage requirements associated with direct solution

of matrix systems. To improve this part of the solution procedure, this article employs efficient iterative methods. These methods increase the potential of finite element solution of backward Kolmogorov equations in higher space dimensions significantly. Bergman, Spencer, and their coworkers only considered white noise excitation of dynamic systems. In two numerical examples, we present results for oscillators excited by filtered white noise.

In the presentation and discussion of numerical results, attention is paid to the regimes of most interest in structural reliability. This includes, for example, only the portion of time evolution curves of R, where R is close to unity. Many contributions to the literature on first passage time statistics put emphasis on methods that show high accuracy as t grows large. However, such validation of methods is of engineering interest only if the large t values are associated with small failure probabilities.

3. Solution methods.

3.1. Spatial finite element discretization. An approximation to $R(t \mid \mathbf{y})$ is sought on the form

$$R(t \mid \mathbf{y}) \approx \sum_{i=1}^{n} H_i(\mathbf{y}) q_i(t).$$

Inserting this expression in the boundary value problem for R and applying the method of weighted residuals with weighting functions W_i, $i = 1, \ldots, n$, yield n ordinary differential equations for $q_i(t)$, $i = 1, \ldots, n$. Terms with second-order derivatives are integrated by parts to achieve a weak weighted residual or a Petrov–Galerkin formulation. On matrix form, the resulting time-dependent equations can be written

$$\mathbf{M}\dot{\mathbf{q}} = \mathbf{L}\mathbf{q}, \qquad \mathbf{q} = (q_1(t), \ldots, q_n(t))^T, \tag{8}$$

where matrices $\mathbf{M}$ and $\mathbf{L}$ have the form:

$$M_{ij} = \int_{\Omega_{\mathbf{x}}} W_i H_j \, dy_1 \ldots dy_d \tag{9}$$

$$L_{ij} = \int_{\Omega_{\mathbf{x}}} \left[W_i \sum_{r=1}^{d} \left(a_r \frac{\partial H_j}{\partial y_r} - \frac{1}{2} \sum_{s=1}^{d} \left(C_{rs} \frac{\partial W_i}{\partial y_r} + \frac{\partial C_{rs}}{\partial y_r} W_i \right) \frac{\partial H_j}{\partial y_s} \right) \right] dy_1 \ldots dy_d + \int_{\partial\Omega_{\mathbf{x}}} \frac{1}{2} W_i \sum_{r=1}^{d} \sum_{s=1}^{d} C_{rs} n_s \frac{\partial R}{\partial y_r} d\Gamma, \tag{10}$$

and n_s is the unit normal on the boundary $\partial\Omega_{\mathbf{x}}$. The surface integral, arising from integration by parts, vanishes in our numerical examples, since R will be prescribed or

$$\sum_{r,s} C_{rs} n_s \partial R / \partial y_r = 0.$$

Finite elements have traditionally been constructed for $d \leq 3$ where the geometry of the element is easily visualized. In this work, we have extended, in detail, the finite element concept to an arbitrary number of spatial dimensions. Our implementation and the numerical experiments presented in this paper utilize isoparametric multilinear elements. In d space dimensions, the H_i functions are defined through the tensor product of one-dimensional linear shape functions. For $d = 2$ the standard bilinear element is recovered, while for $d = 3$

one obtains the trilinear brick. Multiquadratic elements, defined as the tensor product of one-dimensional quadratic elements, have been tested, but no additional efficiency or accuracy was obtained. In fact, multiquadratic elements showed inferior behavior compared to multilinear elements.

The solution of the backward Kolmogorov equation may contain very abrupt stationary spatial variations [3], [18]. In such cases standard Galerkin methods may in such cases lead to strongly oscillating numerical solutions and are generally not applicable. To stabilize the discretization method, it is common to use upwinding techniques. In this paper Petrov–Galerkin methods, where $W_i \neq H_i$, are applied. Two choices of the weighting functions W_i are used. One choice corresponds to the streamline-upwind/Petrov–Galerkin (SUPG) method introduced by Brooks and Hughes [5]. These W_i are piecewise multilinear discontinuous functions on the form

$$W_i = H_i + \tau \sum_{r=1}^{d} a_r \frac{\partial H_i}{\partial y_r},$$

where we have used the following choice of τ [5]:

$$\tau = \frac{\sum_{i=1}^{d} \gamma_i a_i h_i}{\sum_{i=1}^{d} a_i^2}, \quad \gamma_i = \coth \nu_i + \frac{1}{\nu_i}, \quad \nu_i = -\frac{h_i a_i}{C_{ii}},$$

with h_i being the length of the element in the y_i-direction. Later we refer to these functions as the SUPG choice of W_i. All investigations of Petrov–Galerkin methods in this paper are confined to cases where C_{rs} are constant and $C_{rs} = 0$ for $r \neq s$. Such restrictions are relevant for many applications. Modifications of the weighting functions in the nonconstant, nondiagonal case is not considered herein and constitute a topic for further research.

The other choice of weighting functions coincides with those introduced by Heinrich et al. [14]. The d-dimensional form of W_i follows from a tensor product generalization of the one-dimensional forms. The latter have the following expressions on the interval $[-1, 1]$:

$$\tilde{W}_1 = \frac{1}{2}(1-\tilde{\xi})\left(1 + \frac{3}{2}\tilde{\alpha}(1+\tilde{\xi})\right), \qquad \tilde{W}_2 = \frac{1}{2}(1-\tilde{\xi})\left(1 - \frac{3}{2}\tilde{\alpha}(1-\tilde{\xi}_i)\right),$$

where $\tilde{W}_i$ is the function associated with the local node at $\tilde{\xi} = (-1)^i$, $i = 1, 2$, and $\tilde{\alpha}$ is an optimization parameter that is chosen as in [1] and [18]; that is, it equals ν_i when the one-dimensional form is applied for y_i-direction. We will refer to this second choice of W_i as quadratic weighting functions and use the abbreviation QUAD. Most of the development of Petrov–Galerkin methods has concerned the linear convection–diffusion equation for which the backward Kolmogorov equation is a special case. Experience from the literature during the last decade indicates that SUPG is superior to QUAD. Whether this is also the case in the present problem, where the singularities are much stronger than commonly encountered in hydrodynamical applications, is investigated in §6.2.

In this work Petrov–Galerkin methods are combined with mesh refinements. The refinement procedure consists of first generating a uniform mesh and then transforming the ith coordinate of a node according to $y_i = g(\xi_i; \alpha_i, \mu_i^-, \mu_i^+), i = 1, \ldots, d$, where $\mu_i^- \leq \xi_i \leq \mu_i^+$ is the coordinate value in the uniform mesh, y_i is the corresponding coordinate value in the refined mesh, and $\alpha_i > 0$ is a mesh grading parameter. If $\alpha_i < 1$, we obtain a dense mesh around $y_i = 0$ while $\alpha_i > 1$ gives a mesh denser towards the end points. Note that the refinements in the different spatial directions are independent. The particular form of the g-function to be used is

$$(11) \qquad g(\xi; \alpha, a, c) = \frac{1}{2}\operatorname{sign}\{2\xi - (c+a)\}(c-a)\left|\frac{2\xi - (a+c)}{c-a}\right|^{1/\alpha} + \frac{1}{2}(c+a),$$

where $x, \xi \in [a, c]$. Observe that (11) applies to the whole domain. A seemingly more efficient approach would be to introduce local mesh refinements only in the vicinity of the singularities but this would require sophisticated mesh generation techniques that are complicated to implement for $d \geq 3$. Since the numerical method and the computer program were developed to handle an arbitrary d, the described refinement procedure is easily implemented and it is of interest to study its behavior.

Integrals are computed by numerical quadrature where we have experimented with the trapezoidal rule in addition to standard two-point Gaussian quadrature. A possible advantage with the trapezoidal rule is the increased number of zeros in the matrices $\mathbf{M}$ and $\mathbf{L}$. The standard two-point Gaussian quadrature results in up to 3^d nonzeros per matrix row, while the trapezoidal rule may lead to only $2d + 1$ nonzeros per row, which is the same sparsity as produced by standard finite difference methods applied to (2). In higher space dimensions ($d > 3$) such increased sparsity implies significant computational savings both with respect to storage and execution times.

The trapezoidal rule does not increase the sparsity when the original SUPG formulation is applied. To obtain only $2d + 1$ nonzeros per row we have modified the SUPG method. The product of W_i and the convection term gives rise to a term which can be interpreted as an anisotropic diffusion term with diffusion tensor proportional to $a_i a_j$. Our modification consists in neglecting the off-diagonal terms in this diffusion tensor. In the implementation one uses $W_i = H_i$ in (9)–(11), but an additional term

$$\int_{\Omega_x} \left[\sum_{r=1}^{d} \tau a_r^2 \frac{\partial H_i}{\partial y_r} \frac{\partial H_j}{\partial y_r} \right] dy_1 \ldots dy_d$$

is added to the L_{ij} given in (11). The trapezoidal rule with the modified SUPG is later referred to as the dSUPG procedure. Whether dSUPG gives sufficient accuracy is commented in §6.1.

The Pontriagin–Vitt equation is solved by the same finite element method as that employed for (2). Let $M_k \approx \sum_{i=1}^{n} H_i(\mathbf{y}) \tilde{q}_i^{(k)}$. The spatial discretization yields a linear system of algebraic equations for $\tilde{\mathbf{q}}^{(k)} = (\tilde{q}_1^{(k)}, \ldots, \tilde{q}_n^{(k)})^T$:

$$\mathbf{L}\tilde{\mathbf{q}}^{(k)} = \mathbf{b}^{(k)}, \tag{12}$$

where $\mathbf{L}$ is given above and $\mathbf{b}^{(k)}$ is a vector with the ith element equal to

$$-k \int_{\Omega_x} W_i \left(\sum_{j=1}^{n} H_j \tilde{q}_j^{(k-1)} \right) dy_1 \ldots dy_d.$$

3.2. Temporal finite difference discretization. The system of ordinary differential equations for $\mathbf{q}$ is solved by lower-order finite difference schemes. The well-known θ-method gives a recursive set of matrix systems

$$\mathbf{Q}\mathbf{q}^{\ell} = \mathbf{S}\mathbf{q}^{\ell-1}, \quad \mathbf{Q} = \mathbf{M} - \Delta t(1-\theta)\mathbf{L}, \quad \mathbf{S} = \mathbf{M} + \Delta t(1+\theta)\mathbf{L}, \tag{13}$$

where $\mathbf{q}^{\ell}$ denotes $\mathbf{q}(t)$ at time level ℓ and Δt is the current timestep. The θ-scheme is unconditionally stable when $\theta \geq \frac{1}{2}$. The truncation error is of order Δt for $\theta \neq \frac{1}{2}$ and of order Δt^2 when $\theta = \frac{1}{2}$.

A second-order Runge-Kutta method has also been investigated. This scheme takes the form

$$\mathbf{q}^{*} = \mathbf{q}^{\ell-1} + \Delta t \mathbf{M}^{-1}\mathbf{L}\mathbf{q}^{\ell-1}, \tag{14}$$

$$\mathbf{q}^{\ell} = \mathbf{q}^{\ell-1} + \frac{1}{2}\Delta t(\mathbf{M}^{-1}\mathbf{L}\mathbf{q}^{\ell-1} + \mathbf{M}^{-1}\mathbf{L}\mathbf{q}^{*}). \tag{15}$$

$\mathbf{M}$ is lumped to make the method explicit. This scheme is only conditionally stable. The stability criterion depends on the eigenvalues of $\mathbf{M}^{-1}\mathbf{L}$. Proper values of Δt are however determined experimentally in this paper.

Observe that $\mathbf{M}$ and $\mathbf{L}$ are independent of time. Hence the spatial integration and assembly process are carried out only once, and $\mathbf{M}$ and $\mathbf{L}$ are stored separately.

3.3. Solution of matrix systems. When using the θ-method, or when solving the Pontriagin–Vitt equations, it becomes necessary to solve large, sparse matrix systems; see (12) or (13). In fact, the applicability of numerical solution of the backward Kolmogorov equation is to a very large extent governed by the efficiency and the storage requirements of the numerical procedure used for solving matrix systems. Previous contributions to numerical solution of the backward Kolmogorov equation have exclusively employed direct elimination methods such as, e.g., banded Gaussian elimination. Although these may be appropriate in smaller two-dimensional (2D) problems, one encounters extremely large bandwidths and hence prohibitively long CPU times and storage demands in higher space dimensions. Even inside the band the matrices are very sparse for $d \geq 3$, but direct methods replace these zeros generally by nonzero fill-in entries. On the contrary, iterative methods may only operate on the nonzeros in the matrices and hence memory or disk requirements can be significantly reduced. A particularly attractive method for solution of matrix systems involving symmetric, positive definite matrix systems is the conjugate gradient method. However, our coefficient matrices $\mathbf{L}$ and $\mathbf{Q}$ are generally nonsymmetric. There have been numerous generalizations of the conjugate gradient method to cover nonsymmetric systems during the last 15 years. Numerical simulations [16] indicate that the Orthomin(k) [9] method is a robust and efficient method. In demanding problems, Orthominres(k) [16] has shown to be attractive. These two methods are used for solving matrix systems in this paper. The parameter k is related to the amount of storage required by the algorithms [9] and indirectly also to their stability and robustness. Orthominres(k) is used in the restarted version [9] and abbreviated R-OMR(k). Orthomin(k) is abbreviated OM(k).

Unfortunately, the convergence of iterative methods of the conjugate gradient family is slow unless the matrix system is preconditioned. That is, instead of solving the original system $\mathbf{Q}\mathbf{q}^{\ell} = \mathbf{S}\mathbf{q}^{\ell-1}$, one solves the equivalent, left preconditioned system $\mathbf{U}^{-1}\mathbf{Q}\mathbf{q}^{\ell} = \mathbf{U}^{-1}\mathbf{S}\mathbf{q}^{\ell-1}$. If $\mathbf{U}$ is a good approximation to $\mathbf{Q}$ (in some sense), the iterative methods will converge much faster when applied to the preconditioned system. In this work we have employed incomplete LU (ILU) factorization preconditioning, which is suitable for nonsymmetric matrix systems [16]. The preconditioning matrix $\mathbf{U}$ is then computed as a sparse LU decomposition of the original coefficient matrix $\mathbf{Q}$ (or $\mathbf{L}$). This task is accomplished by performing Gaussian elimination on $\mathbf{Q}$ (or $\mathbf{L}$) and neglecting all fill-in entries. The sparsity pattern of $\mathbf{U}$ coincides with that of $\mathbf{Q}$ (or $\mathbf{L}$).

The work required by solving matrix systems is here reported in terms of the number of iterations required by R-OMR(k) and OM(k) in addition to the corresponding number of work units. One work unit equals one addition plus one multiplication divided by the number of unknowns (n). We remark that only the nonzero entries in the coefficient matrix need to be stored. These nonzeros are stored in a sparse matrix storage scheme [16]. The solution at the previous time level is used as start-vector in time dependent problems. When solving equation (6) for $M_k(\mathbf{y})$, $M_{k-1}(\mathbf{y})$ is used as a start-vector. The iteration is terminated when the euclidian norm of the residual-vector in the original matrix system is less than ϵ_r. A choice of $\epsilon_r = 5 \cdot 10^{-4}$ has shown to be sufficient in the examples presented herein.

The size of the matrix systems associated with finite element solution of backward Kolmogorov equations is very large when $d \geq 3$. In many problems the number of unknowns can be reduced by exploiting symmetry. For example, if C_{rs} is constant and a_r is an odd function

of each of its spatial arguments, R or M_k will be symmetric about the origin, and only half of the nodal values need to enter the system of equations. In this work we have taken advantage of symmetry wherever it is possible.

4. Monte Carlo simulation. Verification of the quality of the numerical solution of the first passage time boundary value problem is a nontrivial task. The solution may be sensitive to the location of finite boundaries, the choice of weighting functions, and mesh refinements unless the element size is sufficiently small. However, for $d \geq 3$ it is seldom possible to work with a very fine mesh, and experience shows that useful and fairly reliable results can be obtained on coarser grids if the numerical parameters are tuned properly. To determine a proper choice of values for the numerical parameters one needs some indication of the correct solution. To some extent this may be provided by direct Monte Carlo simulation (MCS).

Temporal realizations of white noise processes can be obtained as described in [7]. Integrating the system (1) numerically, with $\mathbf{x}(t_0) = \mathbf{y} \in \Omega_{\mathbf{x}}$, and measuring the time to first exit of $\mathbf{x}(t)$ from $\Omega_{\mathbf{x}}$, gives a realization of T. Taking the expectations of a large sample of such realizations leads to an estimate of $M_1(\mathbf{y})$. Usually only one $\mathbf{y}$ value, e.g., $\mathbf{y} = \mathrm{E}[\mathbf{X}]$, is sufficient to judge the quality of the solution of the backward Kolmogorov equation. Similarly, MCS may provide estimation of $R(t \mid \mathbf{y})$ for a prescribed t and $\mathbf{y}$. In this work infinite boundaries are located at finite positions. Simulations with various choices of $\mathbf{y}$, letting $\mathbf{y}$ approach the infinite boundaries, give valuable guidance to the determination of the location of the finite boundaries that are required by the solution scheme from §3.

5. Model problems.

5.1. White noise excitation. Consider a single degree-of-freedom oscillator excited by a white noise load with constant spectral density S_0. Assuming the restoring force to be a general function f of the displacement Z and the damping force to be linear, the governing equation can be written

$$\ddot{Z} + 2\zeta\omega_0\dot{Z} + f(Z) = \sqrt{2\pi S_0}N_2(t), \tag{16}$$

where ζ is the damping ratio and ω_0 is the natural frequency. Introducing the Markov process vector $(X_1, X_2)^T$, with $X_1 = Z$ and $X_2 = \dot{X}_1$, the system can be expressed on the form (1). The associated backward Kolmogorov equation has the coefficients $a_1 = y_2$, $a_2 = -2\zeta\omega_0 y_2 - f(y_1)$, $C_{22} = \pi S_0$, and $C_{rs} = 0$, $r \neq 2$, or $s \neq 2$. Let σ_{X_i} be the standard deviation of $X_i(t)$. The boundary conditions for a safe domain $|x_1| \leq \beta\sigma_{X_1}$, $|x_2| \leq \tilde{C}_2$, where $\tilde{C}_2 \to \infty$, using (3)–(5) becomes

$$R(t \mid -\beta\sigma_{X_1}, y_2) = 0, \qquad y_2 < 0, \tag{17}$$

$$R(t \mid \beta\sigma_{X_1}, y_2) = 0, \qquad y_2 > 0, \tag{18}$$

$$R(t \mid y_1, \pm\tilde{C}_2) = 0. \tag{19}$$

It is also possible to prescribe $R = 0$ at $y_2 = 0$, but this has little effect on the solution, except that some numerical noise on the boundary may be slightly amplified (see [1]). The above partial boundary conditions can be given a physical interpretation. At $y_1 = \beta\sigma_{X_1}$ the system will move out of the safe domain if the velocity is positive. Hence $R = 0$ when $y_2 > 0$. In the case where $y_2 < 0$, the system will move into the domain and no value of R can be prescribed. The same reasoning can be applied to the conditions at $x = -\beta\sigma_{X_1}$.

For a linear oscillator we have $f(x) = \omega_0^2 x$. This problem will be referred to as model problem 1. Model problem 2 consists of a nonlinear oscillator with a "bang-bang" spring that has $f(x) = \omega_0 \mathrm{sign}(x)$. First passage time statistics related to this problem have been considered by Toland and Yang [20] using a random walk method. Model problem 3 concerns

a Duffing oscillator where the spring is expressed as $f(x) = \omega_0(x_1 + \varepsilon x_1^3)$. In later sections specifications of the safe domain are given in terms of σ_{X_1}. For model problem 1 $\sigma_{X_1}^2 = \pi S_0/(2\zeta\omega_0^3)$, while for model problem 2 $\sigma_{X_1} = \pi S_0/(\sqrt{2}\zeta\omega_0^2)$.

5.2. Filtered white noise excitation. In model problems 4 and 5, we consider an oscillator with the equation of motion

$$\ddot{Z} + 2\zeta\omega_0\dot{Z} + \omega_0^2(Z + \varepsilon Z^3) = \eta_Q u(Q(t)), \tag{20}$$

where u is a deterministic function and $Q(t)$ is a Gaussian process with mean μ_Q, standard deviation σ_Q, and autocorrelation function $\mathrm{E}[Q(t)Q(t+\tau)] = \sigma_Q^2 \exp(-\epsilon_Q|\tau|)$, that is, Q has a low frequency dominated spectrum. The process $Q(t)$ can be obtained by filtering white noise according to the equation

$$\dot{Q} = -\epsilon_Q(Q - \mu_Q) + \sigma_Q\sqrt{2\epsilon_Q}N_3(t). \tag{21}$$

It is evident that $\mathrm{E}[X_1] = \omega_0^{-2}\eta_Q\mathrm{E}[u(Q)]$ when $\epsilon = 0$.

Introducing $X_1 = Z$, $X_2 = \dot{X}_1$, and $X_3 = Q$, the Markov vector $\mathbf{X} = (X_1, X_2, X_3)^T$ is governed by a system of white noise excited stochastic differential equations of the form (1). The coefficients in the backward Kolmogorov equation read

$$a_1 = y_1, \tag{22}$$

$$a_2 = \eta_Q u(y_3) - 2\zeta\omega_0 y_2 - \omega_0^2(y_1 + \varepsilon y_1^3), \tag{23}$$

$$a_3 = -\epsilon_Q(y_3 - \mu_Q), \tag{24}$$

$$C_{33} = \sigma_Q^2\epsilon_Q, \tag{25}$$

$$C_{ij} = 0, \quad i \neq 3 \quad \text{or} \quad j \neq 3. \tag{26}$$

Different choices of u give rise to different low frequency dominated load processes. To this author's knowledge the backward Kolmogorov equation associated with (22)–(26) has not been solved elsewhere in the literature.

Model problem 4 employs a Gaussian excitation process with $u(Q) = Q$. When $\varepsilon = 0$, one can find exact closed form expressions for the first- and second-order moments by using, e.g., the moment equations. The results are omitted here but the numerical value of σ_{X_1}, which is used in the specifications of $\Omega_{\mathbf{x}}$, will be given later. The linear oscillator with low frequency Gaussian excitation results in a boundary value problem that is particularly challenging to solve. The boundary value problems for R and M_k involve additional complexities compared to, e.g., the $d = 3$ problem investigated by Spencer [18]. Besides general numerical difficulties due to singularities and the shape of the solution, it is necessary to deal properly with the prescription of boundary conditions to ensure a well-posed problem. Defining the domain $\Omega_{\mathbf{x}}$ as $|x_1| \leq \beta\sigma_{X_1}$, $|x_2| < \tilde{C}_2$, and $|x_3| < \tilde{C}_3$, and using (3)–(5), one obtains

$$R(t \mid \beta\sigma_{X_1}, y_2, y_3) = 0, \quad 0 < y_2 < \tilde{C}_2, \quad |y_3| < \tilde{C}_3, \tag{27}$$

$$R(t \mid -\beta\sigma_{X_1}, y_2, y_3) = 0, \quad -\tilde{C}_2 < y_2 < 0, \quad |y_3| < \tilde{C}_3, \tag{28}$$

$$R(t \mid y_1, \tilde{C}_2, y_3) = 0, \quad |y_1| \leq \beta\sigma_{X_1}, \quad |y_3| < \tilde{C}_3, \quad a_2 > 0, \tag{29}$$

$$R(t \mid y_1, -\tilde{C}_2, y_3) = 0, \quad |y_1| \leq \beta\sigma_{X_1}, \quad |y_3| < \tilde{C}_3, \quad a_2 < 0, \tag{30}$$

$$R(t \mid y_1, y_2, \pm\tilde{C}_3) = 0, \quad |y_1| < \beta\sigma_{X_1}, \quad |y_2| < \tilde{C}_2. \tag{31}$$

The mathematical problem we want to solve corresponds to $\tilde{C}_2, \tilde{C}_3 \to \infty$, but finite values must be used in the numerical computations.

Model problem 5 is related to slow-drift oscillations of moored marine structures, where the excitation is often of low frequency and exponential nature. Transformation of a normalized ($\sigma_Q = 1$, $\mu_Q = 0$), normally distributed Q to an exponentially distributed $u(Q)$, with unit expectation and variance, can be carried out by $u(Q) = -\ln(1 - \Phi(Q))$, where Φ is the normalized, univariate, cumulative, normal distribution function. The boundary conditions become different in this problem compared to the model problem 4. The domain Ω_x is defined as $|x_1 - \mathrm{E}[X_1]| \leq \beta\sigma_{X_1}$, $|x_2| < \tilde{C}_2$, and $\tilde{C}_3 < x_3 < \tilde{C}_4$. Note that $\mathrm{E}[X_1] = \eta_Q$. Mathematically, $\tilde{C}_2, \tilde{C}_4 \to \infty$, while $\tilde{C}_3 \to -\infty$. From (3)–(5) it follows that

$$R(t \mid \eta_Q + \beta\sigma_{X_1}, y_2, y_3) = 0, \quad 0 < y_2 < \tilde{C}_2, \quad \tilde{C}_3 < y_3 < \tilde{C}_4, \tag{32}$$

$$R(t \mid \eta_Q - \beta\sigma_{X_1}, y_2, y_3) = 0, \quad -\tilde{C}_2 < y_2 < 0, \quad \tilde{C}_3 < y_3 < \tilde{C}_4, \tag{33}$$

$$R(t \mid y_1, \tilde{C}_2, y_3) = 0, \quad |y_1 - \eta_Q| \leq \beta\sigma_{X_1}, \quad \tilde{C}_3 < y_3 < \tilde{C}_4, \quad a_2 > 0, \tag{34}$$

$$R(t \mid y_1, -\tilde{C}_2, y_3) = 0, \quad |y_1 - \eta_Q| \leq \beta\sigma_{X_1}, \quad \tilde{C}_3 < y_3 < \tilde{C}_4, \quad a_2 < 0. \tag{35}$$

As in model problem 4 one can also prescribe boundary conditions for R on $y_3 = \tilde{C}_3$ and $y_3 = \tilde{C}_4$. It is clear that $R \to 0$ as $Q \to \infty$ so one may set

$$R(t \mid y_1, y_2, \tilde{C}_4) = 0, \quad |y_1 - \eta_Q| < \beta\sigma_{X_1}, \quad |y_2| < \tilde{C}_2. \tag{36}$$

As $Q \to -\infty$ the excitation vanishes and hence has no effect on the reliability. The proper condition at $y_3 = \tilde{C}_3$ is then

$$\frac{\partial}{\partial y_3} R(t \mid y_1, y_2, \tilde{C}_3) = 0, \quad |y_1 - \eta_Q| < \beta\sigma_{X_1}, \quad |y_2| < \tilde{C}_2. \tag{37}$$

In model problems 1–4 it is trivial to show that $R(t \mid \mathbf{y}) = R(t \mid -\mathbf{y})$ with a similar result also for $M_k(\mathbf{y})$. Hence it is possible to reduce the number of nodal values entering the matrix systems by 50%. In the examples presented in the next section we have taken advantage of this symmetry with respect to $\mathbf{y} = \mathbf{0}$.

6. Results. When solving first passage problems associated with oscillating systems, we are concerned with applications to structural reliability. This implies that, for example, $0.9 \leq R(t \mid \mathbf{y}) \leq 1$ is the region of the solution of most interest. Moreover, the bounds on X_1 are fairly large, e.g., $\beta \geq 3$. The results presented below are computed with $\omega_0 = 1$ and $S_0 = 1/\pi$ in model problems 1–3. In model problems 4 and 5 we have used $\varepsilon = 0$, $\eta_Q = 10$, $\sigma_Q = \epsilon_Q = 1$, and $\mu_Q = 0$. In all problems $t_0 = 0$.

The grids consist of $m_1 \times \cdots \times m_d$ multilinear elements of regular, nondistorted hypercube shape. Many of the results are presented in a compact tabular form. In the tables the column text "upw." refers to the choice of upwind weighting functions.

6.1. Integration with the trapezoidal rule. When solving partial differential equations by the finite element method in higher space dimensions, nodal point integration using the trapezoidal rule on multilinear elements give substantial savings in storage requirements, but the accuracy may be questionable. The performance of the trapezoidal rule has been investigated in the previously described model problems. It is unfortunately evident that the trapezoidal rule is not suited in these problems. Table 1 displays a comparison between some different integration rules and choices of weighting functions in model problem 2. The convergence of nodal point integration is seen to be very slow compared to Gaussian quadrature. Since upwind weighting functions are always required to obtain meaningful solutions, the inferior behavior of the trapezoidal rule may be caused by the modified SUPG weighting functions. As Table 1 shows, both modification of SUPG and nodal point integration

contribute to decrease the accuracy. Since the trapezoidal rule with dSUPG weighting gives rise to algebraic equations that are (almost) equivalent to the equations produced by a standard, first-order, upwind finite difference scheme, we may draw the conclusion that the standard finite difference methodology is generally not suitable for numerical solution of first passage problems. Nevertheless, Franklin and Rodemich [12] solved a simple Pontriagin–Vitt equation by a finite difference method with success. Their problem has also been run with the finite element methods in the present work and nodal point integration with dSUPG in fact worked better than Gauss quadrature with full SUPG for this particular equation.

TABLE 1

Nonlinear bang-bang oscillator (model problem 2) with $\zeta = 0.01$, $\beta = 1$, $\tilde{C}_2 = 40$, $\alpha_1 = 3.0$, *and* $\alpha_2 = 0.5$, *except for* $m_1 > 56$, *where* $\alpha_2 = 0.75$. MCS *estimated* $M_1(0, 0) = 170$.

m_1	m_2	upw.	integration rule	$M_1(0, 0)$
28	56	QUAD	Gauss quadrature	169
28	56	SUPG	Gauss quadrature	166
28	56	dSUPG	trapezoidal rule	65
28	56	SUPG	trapezoidal rule	106
70	140	QUAD	Gauss quadrature	170
70	140	SUPG	Gauss quadrature	167
70	140	dSUPG	trapezoidal rule	82

6.2. Sensitivity to numerical parameters. The backward Kolmogorov boundary value problem gives rise to numerical difficulties associated with infinite domains and resolution of singularities. To overcome the difficulties we may use upwind weighting functions, mesh grading, smaller elements and a proper finite location of the boundaries. Some examples are given below to show to what extent various numerical parameters influence the accuracy. One of our aims is also to recover possible problems with or shortcomings of finite element solution of the backward Kolmogorov equation. Previous contributions to the literature have mostly focused on the advantages of the finite element approach. To enable comprehensive future discussions and comparisons of different methods for obtaining first passage time statistics, it is important to have documentation on both advantages and possible limitations of the present solution strategy.

The first example concerns model problem 1. Table 2 displays a comparison of different choices of weighting functions for various grid resolutions. In this example SUPG is more accurate than QUAD on the finest grid. On coarser grids both SUPG and QUAD can lead to $R > 1$ at $t = 14$ although SUPG was more likely to produce occasions of $R > 1$. If one considers the complete $R(t \mid 0, 0)$ curve for $0 \leq t < \infty$, as has been usual in the literature, there are only very small differences between SUPG and QUAD. Generally QUAD was less sensitive than SUPG to variations in $\tilde{C}_2$, α_i and n. In time-dependent problems a uniform grid is shown to be successful [2], [3]. However, our type of mesh grading improves the results in the present example and $\alpha_1 = 2.0$, $\alpha_2 = 1/\alpha_1$ seemed to be a good choice.

Tables 3–6 concern the nonlinear oscillator in model problem 2. In these examples, SUPG was clearly inferior to QUAD with respect to accuracy, and the latter was also more robust than the former. For example, SUPG showed little sensitivity to $\tilde{C}_2$ and α_i for $\beta = 1$, while the sensitivity was very pronounced for $\beta = 4$. On coarse grids QUAD was significantly superior to SUPG.

TABLE 2
Model problem 1, $\zeta = 0.1$, $\Delta t = 0.5$, $\beta = 3$, $\tilde{C}_2 = 90$, $\alpha_1 = 2.0$, *and* $\alpha_2 = 0.5$. MCS *estimated* $R(14\,|\,0, 0) = 0.988$.

m_1	m_2	upw.	$R(14\,\|\,0, 0)$
14	28	SUPG	1.044
14	28	QUAD	0.994
28	56	SUPG	0.993
28	56	QUAD	0.983
42	84	SUPG	0.988
42	84	QUAD	0.984

More elements were needed to maintain sufficient accuracy as the displacement bounds β were increased. For smaller bounds, e.g., $\beta = 1$, the problem is generally easy to solve and there are minor differences between different numerical strategies. MCS also becomes more expensive as β increases; the execution time is typically proportional to $\exp(\sqrt{2}\beta)$. For $\beta = 3$, for example, the finite element approach for finding the complete $M_1(y_1, y_2)$ was much more efficient than MCS for calculating the single value $M_1(0, 0)$.

A trial and error process utilizing plots of R or M_k, in addition to experience and MCS, seems to be the most effective way to determine the boundary location $\tilde{C}_2$. On coarse grids the solution may show considerable sensitivity to changes in $\tilde{C}_2$, especially if $\tilde{C}_2$ is chosen larger than strictly necessary. This sensitivity decreases rapidly as the grid is refined. However, many problems in higher space dimensions must probably be run on a coarse grid. A proper tuning of parameters related to boundary locations and mesh grading may lead to fairly accurate solutions even on coarse grids. Experience with the present set of model problems reveals that comparison of M or R with MCS for a single point, e.g., $(\mathbf{y} = \mathrm{E}[\mathbf{X}])$, may be sufficient in the tuning process.

When solving the moment equation, the optimal mesh grading parameters were $\alpha_1 = 1.5$ and $\alpha_2 = 0.75$, with the exception of a few cases where $\alpha_1 = 2$ led to higher accuracy. In time-dependent problems, a uniform grid seemed to be an efficient choice for model problem 2.

TABLE 3
Model problem 2, $\zeta = 0.1$, $\beta = 1$, $\tilde{C}_2 = 24$, $\alpha_1 = 1.5$, *and* $\alpha_2 = 0.75$. MCS *estimated* $M_1(0, 0) = 26.6$.

m_1	m_2	upw.	$M_1(0, 0)$
14	28	SUPG	24.7
14	28	QUAD	26.2
56	112	SUPG	26.2
56	112	QUAD	26.3

To demonstrate that no particular numerical strategy turned out to be "best" we show results in Table 7 from model problem 3 where SUPG was significantly superior to QUAD. SUPG was also less sensitive to variations in $\tilde{C}_2$.

Model problem 4 turned out to make greater demands to the numerical solution schemes than the other model problems. Since this particular problem has not been solved in the literature before, it may be enlightening to show the typical shape of the reliability function.

TABLE 4
Model problem 2, $\zeta = 0.1$, $\beta = 4$, $\alpha_1 = 1.5$, *and* $\alpha_2 = 0.75$. MCS *estimated* $M_1(0, 0) = 1660$.

m_1	m_2	upw.	$\tilde{C}_2$	$M_1(0, 0)$
28	56	SUPG	24	940
28	56	QUAD	24	1360
28	56	SUPG	54	1170
28	56	QUAD	54	1360
56	112	SUPG	54	1210
56	112	QUAD	54	1490

TABLE 5
Model problem 2, $\zeta = 0.01$, $\beta = 3$, $\alpha_1 = 1.5$, *and* $\alpha_2 = 0.75$. MCS *estimated* $M_1(0, 0) = 1890$.

m_1	m_2	upw.	$\tilde{C}_2$	$M_1(0, 0)$
28	56	SUPG	40	1730
28	56	QUAD	40	1830
28	56	SUPG	90	1860
28	56	QUAD	90	1890
56	112	SUPG	90	1810
56	112	QUAD	90	1890

TABLE 6
Model problem 2, $\zeta = 0.1$, $\Delta t = 0.5$, $\beta = 3$, $\alpha_1 = \alpha_2 = 1.0$. MCS *estimated* $R(15 \mid 0, 0) = 0.992$.

m_1	m_2	upw.	$\tilde{C}_2$	$R(15 \mid 0, 0)$
28	56	SUPG	16	0.991
28	56	QUAD	16	0.992
28	56	SUPG	36	0.994
28	56	QUAD	36	0.992
56	112	SUPG	36	0.992
56	112	QUAD	36	0.992

TABLE 7
Model problem 3, $\zeta = 0.1$, $\varepsilon = 2$, $\beta = 1$, $\alpha_1 = 1.5$, *and* $\alpha_2 = 0.75$. MCS *estimated* $M_1(0, 0) = 87.4$.

m_1	m_2	upw.	$M_1(0, 0)$
28	56	SUPG	83.4
28	56	QUAD	65.1
56	112	SUPG	86.6
56	112	QUAD	76.7

Figures 1–3 display R as a function of a single space coordinate in three space directions and at three different points of time. The other two coordinates are kept fixed at zero. The parameters equaled $\zeta = 0.1$, $\tilde{C}_2 = 360$, $\tilde{C}_3 = 120$, $\beta = 3$, $m_1 = 14$, $m_2 = 20$, $m_3 = 26$,

$\Delta t = 1.0$, $\theta = 1.0$, $\alpha_1 = 2$, $\alpha_2 = 0.667$, and $\alpha_3 = 0.5$. Quadratic weighting functions were used. The finite element solution value for $R(25 \mid 0, 0, 0)$ equaled 0.944, while MCS (10 000 samples) gave 0.954.

SUPG was generally not competitive with QUAD in this problem. Numerical oscillations caused difficulties, and the damping inherent in the θ-scheme when $\theta = 1$ was useful. However, in other problems, $\theta = 1$ may lead to inaccurate results; see [2]. A general feature of model problem 4 is that the solution showed considerable sensitivity to numerical parameters such as m_i, α_i, θ, $\tilde{C}_2$, and $\tilde{C}_3$.

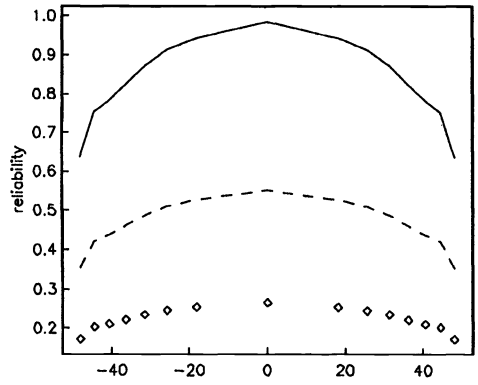


FIG. 1. *$R(t \mid y_1, 0, 0)$ as a function of y_1 for $t = 20$ (solid line), $t = 100$ (dashed line), and $t = 200$ ($\diamond$) in model problem 4.*

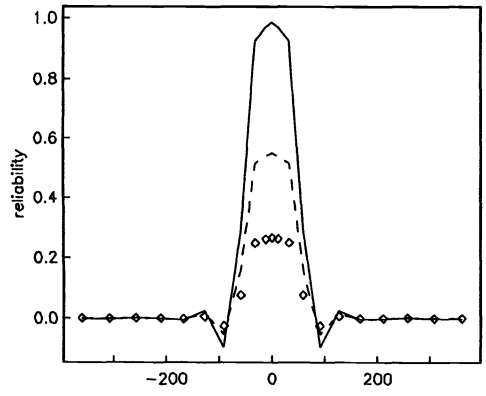


FIG. 2. *$R(t \mid 0, y_2, 0)$ as a function of y_2 for $t = 20$ (solid line), $t = 100$ (dashed line), and $t = 200$ ($\diamond$) in model problem 4.*

Model problem 5 is closely related to model problem 4, but is considerably easier to solve by the present numerical method. As plots of R in this problem do not exist in the literature, examples of the typical features of the function are presented in Figs. 4–6. It is seen that one can work with a much coarser mesh in the y_3-direction than in the other two space directions. For the particular example in Figs. 4–6, the parameters were $\zeta = 0.1$, $\tilde{C}_2 = 240$, $\tilde{C}_3 = -4$, $\tilde{C}_4 = 10$, $\beta = 3$, $m_1 = 22$, $m_2 = 36$, $m_3 = 12$, $\Delta t = 0.5$, $\theta = 0.75$, $\alpha_1 = 2$, $\alpha_2 = 0.667$, and $\alpha_3 = 1.0$. Quadratic weighting functions were used. MCS (10,000 samples) for $R(10 \mid 10, 0, 0)$ resulted in 0.971, while the finite element solution read 0.965 (recall that $\mathrm{E}[X_1] = \eta_Q = 10$).

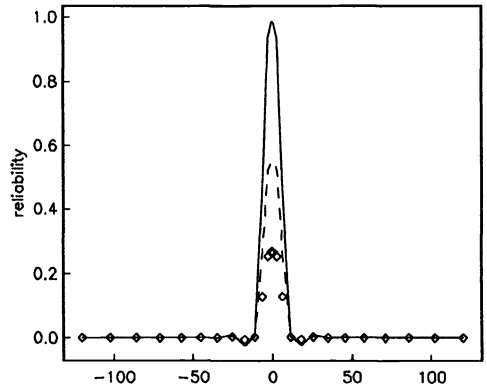


FIG. 3. $R(t \mid 0, 0, y_3)$ *as a function of* y_3 *for* $t = 20$ (*solid line*), $t = 100$ (*dashed line*), *and* $t = 200$ ($\diamond$) *in model problem* 4.

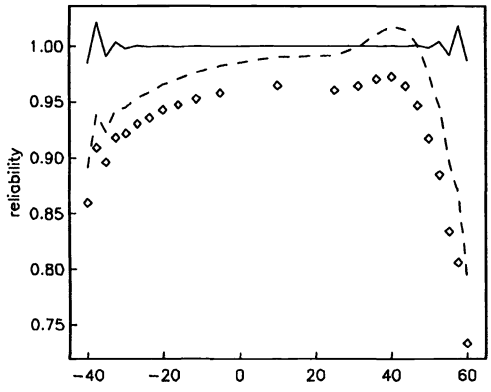


FIG. 4. $R(t \mid y_1, 0, 0)$ *as a function of* y_1 *for* $t = 1$ (*solid line*), $t = 5$ (*dashed line*), *and* $t = 10$ ($\diamond$) *in model problem* 5.

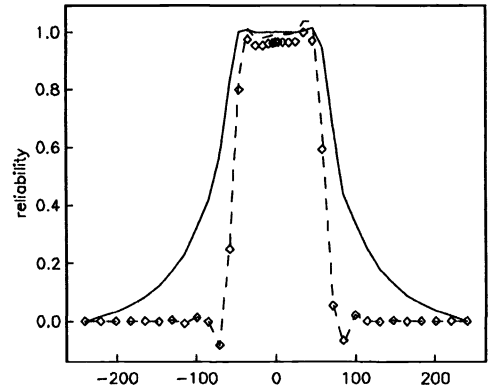


FIG. 5. $R(t \mid 10, y_2, 0)$ *as a function of* y_2 *for* $t = 1$ (*solid line*), $t = 5$ (*dashed line*), *and* $t = 10$ ($\diamond$) *in model problem* 5.

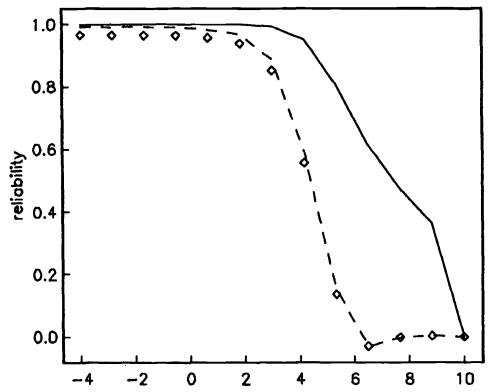


FIG. 6. $R(t \mid 10, 0, y_3)$ *as a function of* y_3 *for* $t = 1$ (*solid line*), $t = 5$ (*dashed line*), *and* $t = 10$ ($\diamond$) *in model problem* 5.

The sensitivity of the solution to the choice of various numerical parameters was significantly smaller than in model problem 4. Tables 8 and 9 show some results.

TABLE 8
Model problem 5, *other parameters are as given in the text.* MCS *estimated* $R(10 \mid 10, 0, 0) = 0.971$.

m_1	m_2	m_3	α_1	α_2	$R(10 \mid 10, 0, 0)$
16	24	12	1	1	0.961
16	24	12	3/2	2/3	0.967
16	24	12	2	1/3	0.938
24	36	18	1	1	0.934
24	36	18	3/2	2/3	0.963
24	36	18	2	1/3	0.972

TABLE 9
Model problem 5, *other parameters are as given in the text.* MCS *estimated* $R(10 \mid 10, 0, 0) = 0.971$.

m_1	m_2	m_3	$\tilde{C}_2$	upw.	$R(10 \mid 10, 0, 0)$
16	24	12	240	QUAD	0.918
16	24	12	240	SUPG	0.938
16	24	12	160	QUAD	0.887
16	24	12	160	SUPG	0.955
22	36	12	240	QUAD	0.965
24	36	18	240	QUAD	0.954
24	36	18	240	SUPG	0.972

The sensitivity of R to variations in the mesh grading parameters is displayed in Table 8. The results corresponds to the SUPG method. Similar, but slightly less accurate results, were obtained by QUAD. As in the previous model problems, QUAD showed less sensitivity (in comparison with SUPG) to perturbations in the numerical parameters as n was increased.

Table 9 displays some additional results when $\alpha_1 = 2$, $\alpha_2 = \frac{2}{3}$, and $\alpha_3 = 1$. In the present problem, SUPG is slightly superior to QUAD. A proper resolution of the solution in x_2- and x_3- direction in the interior parts of the domain requires a minimum number of elements to be used. Therefore $\tilde{C}_2$ and $\tilde{C}_3$ needs to be smaller on a coarse mesh compared to the optimal values on a finer mesh. If the mesh is too coarse, e.g., $n = 819$ in the present problem, significantly inaccurate results may occur and the solution is usually considerably sensitive to variations in numerical parameters such as $\tilde{C}_i$ and α_i. For $d > 3$ it would be very advantageous to have numerical methods that could produce qualitatively acceptable results on coarse grids ($m_i \sim 10$).

The value of Δt had little impact on the accuracy in model problem 5 as long as $\Delta t \leq 1$, reflecting that the spatial approximation properties constitute the critical parts of the numerical method.

6.3. Behavior of iterative equation solvers. Utilizing iterative methods for solving matrix systems increases the potential of finite element discretization of backward Kolmogorov equations significantly. This subsection reports the efficiency of the approach in more detail.

When solving the time-dependent backward Kolmogorov equation, the solution at the previous time level can be used as start vector for the iterative methods. In such cases only a few iterations, usually one–six, are necessary to obtain a converged solution when using preconditioning. Roughly speaking, R-OMR(k) requires approximately half as many iterations as OM(k), but is also twice as expensive per iteration. The number of iterations increases only slightly with n and Δt. The work required by the equation solvers showed negligible sensitivity to other numerical parameters. If preconditioning is not applied, the work per time level increases significantly. As an example, consider model problem 2 with $\zeta = 0.01$, $\beta = 3$, $m_2 = 2m_1 = 56$, and QUAD. R-OMR(5) and OM(1) required about 2500–3000 work units for solving the system of equations when no preconditioning was applied. With preconditioning the number of work units was reduced to about 170.

In stationary problems associated with the solution for $M_k(\mathbf{y})$, preconditioning is almost always required to avoid divergence or extremely slow convergence of the iterative solvers. Table 10 shows some selected results for the behavior of R-OMR(k) in model problem 2. It is evident that the method is sensitive to the choice of the parameter k. The optimal value of k is significantly larger in the present type of problems than in hydrodynamical convection-diffusion problems. As β is increased, k must also be increased to achieve stability and fast convergence. With a sufficiently large k, the iterative method is robust and the work increases very slightly with increasing n. It is seen that the work increases rapidly with β and n when k is small. OM(k) for small k turned out to be useless for $\beta \geq 3$. In general, R-OMR(k) was more efficient than OM(k) in the most demanding problems. The work associated with the iterative solvers also depended on the boundary locations and on the mesh grading parameters if k was not sufficiently large to ensure robustness. In cases where the numerical parameters led to low accuracy in the solution, the iterative solver usually required a large number of iterations. Thus if the convergence of the iterative solver is slow it may indicate that k is too small or that the numerical parameters like m_i, α_i, $\tilde{C}_i$, etc., are not properly tuned.

6.4. Implicit versus explicit time integration. Since storage requirements and not CPU time seem to be the main limitation for solving higher dimensional backward Kolmogorov equations, it may be preferable to use explicit time integration schemes like the second-order Runge–Kutta method. However, there are two main disadvantages with explicit schemes. The first is that the finite element integration and assembly process is usually much more costly than solving the matrix system, at least for the n-values relevant to the presently available computer generation. The explicit method should therefore avoid the assembly process at each

TABLE 10

Number of iterations for R-OMR(k) *in model problem* 2 (*Pontriagin–Vitt equation*). $\zeta = 0.1$, $\alpha_1 = 1.5$, $\alpha_2 = 0.75$, $\tilde{C}_2 = 60$ ($\beta = 1$), *and* $\tilde{C}_2 = 135$ ($\beta = 3$).

n	β	k	it.
1653	1	1	35
6441	1	1	89
1653	1	9	9
6441	1	9	23
1653	3	1	589
6441	3	1	> 900
1653	3	9	19
6441	3	9	316
1653	3	15	24
6441	3	15	28

time level. This is easily accomplished by storing the matrices **M**, **L**, **S**, and **Q** and carrying only matrix-vector products at each timestep. Nevertheless, such an approach requires about the same storage as the implicit method. The second disadvantage of an explicit scheme is that Δt is related to the size of the smallest element in the mesh due to stability requirements. In the present formulation of first passage problems, small elements are needed in the vicinity of singular points at the boundary to avoid unacceptable numerical instabilities in the spatial discretization.

The explicit Runge–Kutta method has been tested on some of our model problems and found less efficient than the implicit approach. Table 11 shows the performance of the explicit method in the same problem as covered by the implicit scheme in Table 6. It is seen that the accuracy of the Runge–Kutta method is slightly inferior to the results produced by the implicit approach. At each time level, the explicit method requires two matrix-vector products. Consideration of the typical work per timestep in the implicit method reveals that the explicit strategy can be faster than the implicit strategy if the Δt required by the explicit method is not less than $\frac{1}{12}$ of the Δt that is suitable in an implicit scheme. A central question is what happens when the mesh is refined. If $m_1 = 56$ and $m_2 = 112$ are used in the problem in Table 11, the explicit method is not stable even with $\Delta t = 0.0005$ and the implicit integration strategy is extremely faster. The fact that temporal truncation errors can usually be ignored in comparison with spatial truncation errors in the present type of problems makes it practical to employ time steps that are considerably larger than what is dictated by stability requirements of explicit schemes.

7. Conclusion and discussion. The paper has presented and evaluated general finite element solution methods for computing first passage time statistics of oscillating systems. Use of preconditioned conjugate gradient-like methods for solving matrix systems was an important part of the method for obtaining efficiency. In the problems treated herein, the time spent on solving matrix systems was usually much smaller than the time spent on the element by element spatial integration process.

The numerical examples included novel model problems, the impact of various numerical parameters on the accuracy, comparisons of explicit versus implicit time integration and verification by comparison with Monte Carlo simulations. The main conclusion is that the per-

TABLE 11

Explicit second-order Runge–Kutta time integration. Model problem 2, $\zeta = 0.1$, $\Delta t = 0.05$, $\beta = 3$, $\alpha_1 = \alpha_2 = 1.0$. MCS *estimated* $R(15\,|\,0, 0) = 0.992$.

m_1	m_2	upw.	$\tilde{C}_2$	$R(15\,\|\,0, 0)$
28	56	SUPG	16	0.990
28	56	QUAD	16	0.989
28	56	SUPG	36	0.991
28	56	QUAD	36	0.990

formance of the finite element method is very dependent on the type of problem being solved. In some problems, for example, model problems 1, 2, and 5, the method must be considered as fairly robust and easily used. Other problems recover serious sensitivity to numerical parameters. When the interest concerns conditions corresponding to low failure probabilities, the present numerical formulation is particularly attractive since only a few time steps (typically 10–40) are needed and the iterative solution of matrix systems is extremely efficient in time-dependent problems. The general indication of our comparison between implicit and explicit time integration reveals that pure explicit schemes are inferior to implicit methods with respect to efficiency. It should be emphasized that explicit-implicit approaches may be advantageous where implicit timestepping is used for the small elements around the singular points, while explicit timestepping is used for elements of larger size. The development of numerical methods with better spatial approximation properties are especially warranted. Such methods should ideally improve the damping of oscillations due to singularities at the boundaries, guarantee that $0 \leq R \leq 1$, and give qualitatively acceptable results on coarse grids as this would be an important property when $d > 3$.

A central question is whether the proposed methods can compete with MCS. Of course, the computational work associated with numerical solution of partial differential equations increases exponentially with d, while the increase is only linear in d for MCS methods. As a simple model one may consider T to be exponentially distributed. Then the standard deviation of the estimator for $\mathrm{E}[T]$ used in MCS behaves as $\tilde{m}^{-1/2}e^{-\beta^2/2}$, where $\tilde{m}$ is the number of samples of T and β represents the width of the failure bounds in standard deviation units. In the finite element method it will suffice to keep the number of elements per unit length constant as β increases, and hence the work is roughly proportional to β^d. Thus for narrow bounds MCS is effective, especially for larger d values. As the bounds increase the finite element approach becomes superior. However, for large failure bounds serious numerical instabilities may arise in the finite element method. In such cases simpler approximation formulas for the first passage time statistics are usually accurate. Another important fact is that MCS yields consistent values of the statistics, while the numerical solution of the backward Kolmogorov equation may give probabilities that exceed unity. The general conclusion must then be that if the finite element method is robust in the problem being solved, the bounds correspond to $\beta \sim 3 - 4$, the finite element scheme is more efficient than MCS.

Acknowledgments. The author thanks Professor Bill Spencer for valuable discussions.

REFERENCES

[1] L. A. BERGMAN AND J. C. HEINRICH, *Petrov–Galerkin finite element solution for the first passage time of the randomly accelerated free particle*, Comp. Methods Appl. Mech. Engrg., 27 (1981), pp. 345–362.

[2] L. A. BERGMAN AND J. C. HEINRICH, *On the reliability of the linear oscillator and systems of coupled oscillators*, Internat. J. Numer. Methods Engrg., 18 (1981), pp. 345–362.

[3] L. A. BERGMAN AND B. F. SPENCER, *Solution of the First Passage Problem for the Simple Linear and Nonlinear Oscillators by the Finite Element Method*, Report no. 461 (UILU-ENG 83-6007), Dept. of Theoretical and Applied Mechanics, University of Illinois at Urbana-Champaign, 1983.

[4] ———, *First passage time for linear systems with stochastic coefficients*, Probab. Engrg. Mech., 2 (1987), pp. 46–53.

[5] A. BROOKS AND T. J. R. HUGHES, *A streamline-upwind/Petrov–Galerkin finite element formulation for advection dominated flows with particular emphasis on the incompressible Navier–Stokes equations*, Comp. Methods Appl. Mech. Engrg., (1982), pp. 199–259.

[6] T. K. CAUGHEY, *Nonlinear theory of random vibrations*, Adv. Appl. Mech., 11 (1971), pp. 209–253.

[7] S. H. CRANDALL, K. L. CHANDIRAMANI, AND R. G. COOK, *Some first passage problems in random vibrations*, J. Appl. Mech. ASME, 33 (1966), pp. 532–538.

[8] ———, *First Crossing Probabilities of the Linear Oscillator*, J. Sound Vibra., 12 (1970), pp. 285–299.

[9] S. C. EISENSTAT, H. C. ELMAN, AND M. C. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 20 (1983), pp. 345–357.

[10] I. ELISHAKOFF AND B. F. SPENCER, *Reliability of an uncertain sliding structure*, J. Sound Vibration, 114 (1987), pp. 399–404.

[11] G. FICHERA, *On a unified theory of boundary value problems for elliptic-parabolic equations of second order*, in Boundary Problems in Differential Equations, R. E. Langer, ed., Univ. of Wisconsin Press, 1960, pp. 97–120.

[12] J. N. FRANKLIN AND E. R. RODEMICH, *Numerical analysis of an elliptic-parabolic partial differential equation*, SIAM J. Numer. Anal., 5 (1968), pp. 680–716.

[13] C. W. GARDINER, *Handbook of Stochastic Methods*, 2nd ed., Springer-Verlag, New York, 1985.

[14] J. C. HEINRICH, P. S. HUYAKORN, O. C. ZIENKIEWICZ, AND A. R. MITCHELL, *An "upwind" finite element scheme for two dimensional convective transport equations*, Internat. J. Numer. Methods Engrg., 11 (1977), pp. 131–143.

[15] R. S. LANGLEY, *A variational formulation of the* FPK *equations with application to the first passage problem in random vibration*, J. Sound Vibration, 123 (1988), pp. 213–227.

[16] H. P. LANGTANGEN, *Conjugate gradient methods and* ILU *preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns*, Internat. J. Numer. Methods Fluids, 9 (1989), pp. 213–233.

[17] J. B. ROBERTS, *First passage probabilities for randomly excited systems: Diffusion methods*, Probab. Engrg. Mech., 1 (1986), pp. 66–81.

[18] B. SPENCER, JR., *Reliability of randomly excited hystertic structures*, Lecture Notes in Engineering, No. 21, Springer-Verlag, New York, 1986.

[19] J.-Q. SUN AND C. S. HSU, *First-passage time probability of non-linear stochastic systems by generalized cell mapping method*, J. Sound Vibration, 124 (1988), pp. 233–248.

[20] R. H. TOLAND AND C. Y. YANG, *Random walk model for first passage probability*, J. Engrg. Mech. Div., ASCE, 97 (1971), pp. 791–807.

SIAM J. SCI. COMPUT.
Vol. 15, No. 4, pp. 997–1001, July 1994

013

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.

CONTOUR DYNAMICS AND THE FAST MULTIPOLE METHOD*

H. SCHMITT†

Abstract. Contour dynamics methods are used to evolve regions of constant vorticity moving under the Euler equations for an inviscid, incompressible fluid in two dimensions. If the induced velocity field is evaluated directly, contour dynamics methods require a CPU time per timestep proportional to N^2 (N being the number of nodes used to represent the contour). The fast multipole method is used to obtain an algorithm whose CPU time is $O(N)$. This results in speedup factors of 100 to 1000 without any loss of accuracy.

Key words. CFD, contour dynamics, fast multipole method, treecodes

AMS subject classifications. 65D99, 65M99, 76C05

1. Mathematical formulation.

Euler's equations in two dimensions for incompressible, inviscid flow can be written as the set of equations

$$\frac{D\omega}{Dt} = 0,$$

$$\Delta\psi = \omega,$$

$$\mathbf{u} = (u, v) = (-\partial_y, \partial_x)\psi,$$

where ω is the vorticity and ψ is the stream function. The velocity u is then given by

$$\mathbf{u}(\mathbf{x}) = \nabla^{\perp}\Delta^{-1}\omega = \iint_{\mathbf{R}^2} \omega(\boldsymbol{\xi})\nabla^{\perp}G(\mathbf{x}, \boldsymbol{\xi})d\xi d\eta,$$

where $\mathbf{x} = (x, y)$, $\boldsymbol{\xi} = (\xi, \eta)$, and $G(\mathbf{x}, \boldsymbol{\xi}) = \frac{1}{2\pi}\log|\mathbf{x}-\boldsymbol{\xi}|$. For a regular region Ω (a bounded closed region whose boundary is a closed regular curve) with vorticity $\omega \equiv 1$ inside Ω and $\omega \equiv 0$ outside Ω, the complex velocity is

$$\begin{aligned} w(z) = u - iv &= \frac{1}{2\pi i}\int_{\Omega} \frac{1}{z-\theta}d\xi d\eta \\ &= \frac{1}{4\pi}\oint_{\partial\Omega} \frac{\bar{z}-\bar{\theta}}{z-\theta}d\theta, \end{aligned} \tag{1}$$

where $z = x + iy$ and $\theta = \xi + i\eta$. Integration by parts gives

$$w(z) = -\frac{1}{4\pi}\oint_{\partial\Omega}\Big[\log(z-\theta)d\bar{\theta} + \overline{\log(z-\theta)d\theta}\Big]. \tag{2}$$

*Received by the editors April 30, 1993; accepted for publication (in revised form) December 20, 1993.
†Department of Mathematics, ETH, 8092 Zürich, Switzerland (schmitt@sam.math.ethz.ch).

In a nutshell, contour dynamics [3], [4], [10] consists of solving the infinite-dimensional system of ordinary differential equations

$$\overline{Z}_t(\sigma,t) = \frac{1}{4\pi}\int_0^1 \frac{\overline{Z}(\sigma,t)-\overline{Z}(\sigma',t)}{Z(\sigma,t)-Z(\sigma',t)} Z_\sigma(\sigma',t)d\sigma',$$

where $Z(\sigma,t) : [0,1] \to \partial\Omega(t)$ is a parametrization of $\partial\Omega(t)$.

2. The $O(n)$ algorithm. The velocity field along $\partial\Omega$ can be evaluated efficiently by the fast multipole method [2], [6]–[9], subsequently denoted by FMM. Both (1) and (2) are suitable for fast summation. We derive the necessary expressions for (1); (2) is treated similarly. Suppose that $\partial\Omega$ is represented by N nodes z_j, $j = 1,\dots,N$, and denote the part of $\partial\Omega$ between z_j and z_{j+1} by γ_j. Then the velocity is

$$w(z) = \sum_{j=1}^{N} \frac{1}{4\pi}\int_{\gamma_j} \frac{\overline{z}-\overline{\theta}}{z-\theta}d\theta =: \sum_{j=1}^{N} w_j(z).$$

Let $m_j = (z_j + z_{j+1})/2$. If $|z-m_j| > |\theta - m_j|$ for all $\theta \in \gamma_j$, $w_j(z)$ becomes

$$\begin{aligned} w_j(z) &= \frac{1}{4\pi}\int_{\gamma_j} \frac{\overline{z}-\overline{m}_j-(\overline{\theta}-\overline{m}_j)}{z-m_j-(\theta-m_j)}d\theta \\ &= \frac{1}{4\pi}\int_{\gamma_j} (\overline{z}-\overline{m}_j-(\overline{\theta}-\overline{m}_j))\sum_{n=0}^{\infty}\frac{(\theta-m_j)^n}{(z-m_j)^{n+1}}d\theta \\ &= \sum_{n=0}^{\infty}\frac{1}{4\pi}\int_{\gamma_j}(\theta-m_j)^n d\theta \cdot \frac{\overline{z}-\overline{m}_j}{(z-m_j)^{n+1}} \\ &\quad - \sum_{n=0}^{\infty}\frac{1}{4\pi}\int_{\gamma_j}(\overline{\theta}-\overline{m}_j)(\theta-m_j)^n d\theta \cdot \frac{1}{(z-m_j)^{n+1}}. \end{aligned} \tag{3}$$

For polynomial interpolation between the nodes, the integrals $\int(\theta-m_j)^n d\theta$ and $\int(\overline{\theta}-\overline{m}_j)(\theta-m_j)^n d\theta$ can be evaluated analytically without much cost. The far field expansions (3) can be summed by the FMM. As for the local evaluation of the terms w_j, let γ_j be given by $q_j(\sigma) = z_j + t_j\sigma + n_j p_j(\sigma)$, where $t_j = z_{j+1}-z_j$, $n_j = it_j$, and $p_j(\sigma)$ is real-valued and differentiable, $\sigma \in [0,1]$, and $p_j(0) = p_j(1) = 0$. Then,

$$\begin{aligned} w_j(z) &= \frac{1}{4\pi}\int_{\gamma_j}\frac{\overline{z}-\overline{\theta}}{z-\theta}d\theta \\ &= \frac{1}{4\pi}\Bigg[-((\overline{z}-\overline{z}_{j+1})\log(z-z_{j+1}) - (\overline{z}-\overline{z}_j)\log(z-z_j)) \\ &\quad + \frac{\overline{t}_j}{t_j}\Bigg[(z-z_{j+1})\log(z-z_{j+1}) - (z-z_j)\log(z-z_j) \\ &\quad - (z_{j+1}-z_j) + 2\int_0^1 (z-(z_j+t_j\sigma))\frac{q_j'(\sigma)}{z-q_j(\sigma)}d\sigma\Bigg]\Bigg]. \end{aligned} \tag{4}$$

In the case of linear interpolation (i.e., $p_j(\sigma) \equiv 0$), the last integral is simply $z_{j+1}-z_j$, and in the case of cubic interpolation, $z - q_j(\sigma)$ can be factored by Cardano's formulas and the integral can again be evaluated exactly.

3. Numerical test and accuracy. We use the algorithm to simulate the evolution of two circular patches of constant vorticity $\omega = 1$ whose radii are $r = \frac{1}{10}$ and that are separated by $\frac{1}{2}r$. The interpolation is linear. When the integrals (3) are evaluated, a segment γ_j should be ideally contained in a leaf node of the (adaptive) FMM. In practice, we can simply take the maximal distance δ between two adjacent nodes z_j and z_{j+1} small in comparison to the length Δx of the smallest box in the FMM, and we consider γ_j to be in a box if the midpoint of z_j and z_{j+1} is in the box; we set $\delta = 2^{-11}$ and $\Delta x = 2^{-9}$. Initially, each circle is parametrized by 2000 points. We use a fourth-order Runge–Kutta method for the time integration with the timestep $\Delta t = 0.1$. After each timestep, the nodes are adjusted to resolve the curvature and to keep $\delta \leq 2^{-11}$, see [1], [3]. Figure 1 shows the evolution for the times $t = 0, 10, 20, 30, 40,$ and 50. At the end there are over 30,000 points on each contour. The computation takes about two weeks on a Sparc Station 2; on the average it takes about 10 minutes to evaluate the velocity field along the contour. If we had evaluated the velocity directly, the computation would have taken 150 times as long (this number has been estimated by calculating the velocity field directly at selected times).

The accuracy of the algorithm is determined mainly by the contour dynamics method, i.e., the number of nodes used to resolve the contours, the kind of interpolation between the nodes, and the timestep Δt; see, for example, [4]. The FMM does not introduce any new errors. As a rule of thumb one needs approximately three terms in all the multipole expansions to guarantee one digit of accuracy; in practice the FMM is often considerably more accurate; see [6]. In our calculations we use 50 terms, which guarantee us twelve digits of accuracy. As for our particular calculation, we mention that the area enclosed by the two contours, which is an invariant, changes by less than one part in 10^6.

4. Generalizations. The algorithm can be applied to vorticity distributions of the form $\omega = \sum \omega_i I_i$, I_i being indicator functions; the cost is $O(N)$, where N is the total number of nodes on all the contours. The same algorithm can be used if the kernel in the above integrals is not as simple and if the integrals cannot be evaluated analytically anymore; in the case of 2π-periodic vorticity layers [1], (1) and (2) become

$$w(z) = \frac{1}{8\pi} \int_{\gamma_1 - \gamma_2} (\bar{z} - \bar{\theta}) \cot \frac{z - \theta}{2} d\theta, \tag{1'}$$

and

$$w(z) = -\frac{1}{4\pi} \int_{\gamma_1 - \gamma_2} \left[\log \sin \frac{z - \theta}{2} d\bar{\theta} + \overline{\log \sin \frac{z - \theta}{2} d\theta} \right]. \tag{2'}$$

To apply the FMM to (1′) we use the Laurent series of $\cot z$; the far field expansion corresponding to (3) then has a negative as well as a positive part, and to get the formula corresponding to (4) we integrate the Laurent series of $\cot z$ term by term and the integral can again be evaluated to any desired precision.

Remark. If z is away from the region Ω, the second term on the right-hand side of (2) (or (2′)) vanishes; one can then drop that term and the evaluation of the velocity field for points $z \notin \Omega$ is thus essentially cut in half (and so is the required storage space). The advantage of (1) and (2) (or (1′) and (2′)) is that we have to pay no attention to the topology of the curves and even in the case of nested contours with different vorticities we can use the scheme without any complications.

Acknowledgements. The author would like to thank Thomas Buttke and Leslie Greengard for interesting conversations and one of the referees for pointing out [5].

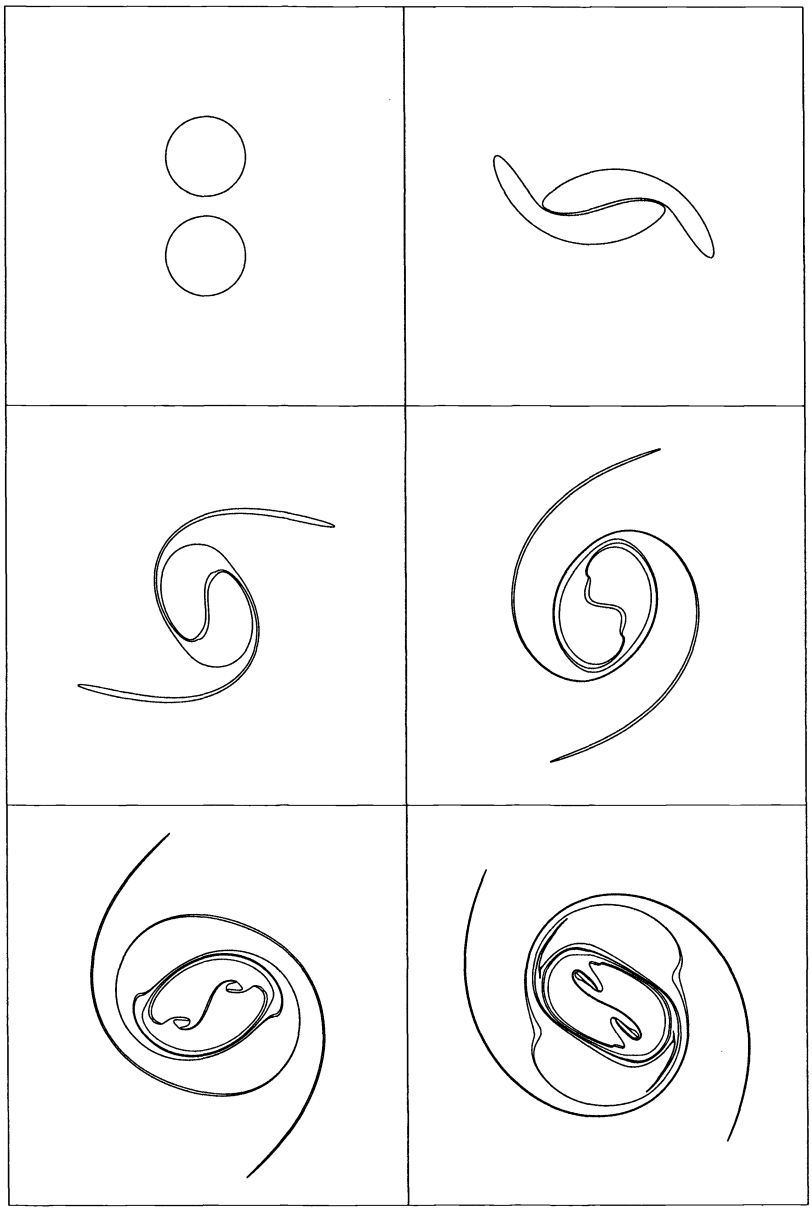


FIG. 1.

REFERENCES

[1] G. R. BAKER AND M. J. SHELLEY, *On the connection between thin vortex layers and vortex sheets*, J. Fluid Mech., 215 (1990), pp. 161–194.

[2] 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.

[3] D. G. DRITSCHEL, *Contour surgery*, J. Comput. Phys., 77 (1988), pp. 240–266.

[4] ———, *Contour dynamics and contour surgery*, Comput. Phys. Rep., 10 (1989), pp. 77–146.

[5] ———, *A fast contour dynamics method for many-vortex calculations in two-dimensional flows*, Phys. Fluids A, 5 (1993), pp. 173–186.

[6] L. F. GREENGARD, *The Rapid Evaluation of Potential Fields in Particle Systems*, MIT Press, Cambridge, MA, 1988.

[7] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[8] J. K. SALMON AND M. S. WARREN, *Skeletons from the treecode closet*, J. Comput. Phys., 1994, to appear.

[9] M. S. WARREN AND J. K. SALMON, *Astrophysical N-body simulations using hierarchical tree data structures*, in Proc. Supercomputing, IEEE Computer Society Press, Los Alamitos, CA, 1992.

[10] N. ZABUSKY, M. HUGHES, AND K. ROBERTS, *Contour dynamics for the Euler equations in two Dimensions*, J. Comput. Phys., 30 (1979), pp. 96–106.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1003–1025, September 1994

SMOOTHING SPLINE SCORE ESTIMATION*

PIN T. NG†

Abstract. A new characterization and interpretation of the Cox [*Ann. Instit. Statist. Math.*, 37 (1985), pp. 271–288] smoothing spline score estimator is provided, which makes it possible to construct an efficient algorithm for computing this score estimator. On choosing the smoothing parameter, the author proposes adaptive information criteria that outperform conventional data-driven choice criteria based on the assumption of Gaussian innovation. A small Monte-Carlo experiment is performed to investigate the finite sample properties of the smoothing spline score estimator as compared to adaptive kernel and weighted kernel score estimators. It is demonstrated that the smoothing spline score estimator is more robust to distributional variation and that all forms of the adaptive information criteria for choosing the smoothing parameter outperform conventional data driven smoothing parameter choice methods based on the Gaussian innovation assumption.

Key words. score, splines, robust model selection, banded matrices

AMS subject classifications. primary: 62G05; secondary: 62J05, 62P20, 65D07, 65D10, 65F50

1. Introduction.

The score function of a probability density function f_0, defined as $\psi_0 = -(\log f_0)' = -(f_0'/f_0)$, is essential to many aspects of nonparametric and robust statistics [36], [18], [21]. For data exploration purposes, we can locate modes or antimodes, as well as assess the tail behavior of the underlying density through examining the estimated score function. Accurate estimate of the score function is crucial to the construction of the asymptotically robust adaptive estimators of the linear model [3], [27], [32]. In an empirical study of the law of demand, Härdle, Hildenbrand, and Jerison [16] used an estimated score function to construct a matrix that was closely related to the mean income effect matrix. The log-density estimators as investigated in [38] and [31] can also be recovered from the score estimator through integration.

Most existing score function estimators are constructed by computing the negative logarithmic derivative of some ad hoc kernel based density estimators. Stone [40] proposed the negative logarithmic derivative of a standard Gaussian kernel estimate as did Manski [27]. Using different window widths for the numerator and denominator of the score function, Cox and Martin [8] suggested a bikernel score estimator that attains a faster convergence rate than conventional single kernel estimators. Csörgő and Révész [10] discussed a nearest-neighbor approach.

Using a totally different approach, Cox [7] proposed a direct method to estimate the score function by minimizing a penalized mean squared error rule. Using a reproducing kernel Hilbert space argument, he proved the existence of a unique solution to the objective function, the consistency of the estimator, and its achievement of an optimal rate of convergence among other score function estimators under some mild regularity conditions. The construction and implementation of the estimator are, however, not explored in detail in Cox [7]. In §2 of this paper, we show that this approach is closely related to nonparametric curve fitting by penalized likelihood and the resulting estimator, given a natural choice of the smoothness penalty, is a variant of the cubic smoothing spline that has a nice nonparametric regression interpretation. This new characterization to the problem using nonparametric regression analogy is appealing because the form of the estimator is computationally tractable. We provide, in §2, an efficient algorithm for computing this score estimator taking advantage of the banded structure of the relevant linear algebra.

*Received by the editors June 17, 1991; accepted for publication (in revised form) May 12, 1993.
†Economics Department, University of Houston, Houston, Texas 77204-5882 (pin@uh.edu).

All score estimation methods share the problem of having to chose a degree-of-smoothness parameter, say λ, which determines the trade-off between data fidelity and the smoothness of fit. In contrast to kernel-based methods, for which maximum likelihood methods of choosing λ break down completely, we suggest, in §3, various versions of adaptive information choice criteria based on an estimated likelihood for choosing the smoothing parameter of the smoothing spline score estimator.

There is always the question: "How well does an asymptotically optimal estimator perform in finite sample situations?" This question is studied through a small Monte-Carlo simulation in §4. Our results show that the performance of a kernel-based score function estimator depends very much on the type of kernel used. In practice, of course, we have little a priori information on the unknown density function and hence little guidance on the type of kernel to use. In this respect, we find the smoothing spline score estimator to be more robust to distributional variation. Our simulations also show that all forms of adaptive information criteria for choosing the smoothing parameter outperform conventional data driven smoothing parameter choice methods based on the Gaussian innovation assumption.

2. The smoothing spline score estimator. Cox [7] observed that if $f_0 = F_0'$, and $\psi_0 = -f_0'/f_0$, then under very mild regularity conditions,

$$\int \psi_0 \zeta \, dF_0 = -\int f_0'(x)\zeta(x)\, dx = \int \zeta' \, dF_0$$

and consequently

$$\int (\psi - \psi_0)^2 \, dF_0 = \int (\psi^2 - 2\psi') \, dF_0 + \int \psi_0^2 \, dF_0. \tag{1}$$

Since the second term on the right-hand side of (1) is independent of ψ, minimizing the integrated mean-squared error may focus exclusively on the first term. Of course we do not observe F_0, so Cox [7] considered the (penalized) empirical analogue,

$$L[\psi] = \int (\psi^2 - 2\psi') \, dF_n + \lambda \int (\psi''(x))^2 \, dx. \tag{2}$$

Minimizing this expression yields a balance between "fidelity-to-data" represented by the mean-squared error term and smoothness of $\hat{\psi}$. Obviously, the tradeoff between these two objectives is controlled by the parameter λ. Cox shows that as $\lambda \to \infty$, so smoothness becomes the paramount objective; $\hat{\psi}$ converges to a linear function corresponding to the ψ-function of a Gaussian random variable with mean and standard deviation of the sample. Before further exploring the solution to minimizing $L[\psi]$, it is helpful to review the closely related problem of estimating a scalar, Gaussian, nonparametric regression problem with the smoothing spline.

2.1. Nonparametric regression. Consider the model

$$y_i = g(x_i) + u_i, \qquad i = 1, 2, \ldots, n,$$

where x_i is a known sequence of scalars in [0, 1], g is an unknown, but presumably smooth response function, and u_i is a sequence of independently and identically distributed (i.i.d.) Gaussian random variables. An attractive approach to the nonparametric estimation of g is to minimize

$$L[g] = n^{-1} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int (g''(x))^2 \, dx \tag{3}$$

over g in the Sobolev space $G_2[0, 1] = \{g \mid g,\ g' \text{ absolutely continuous and } g'' \in L_2[0, 1]\}$. See Reinsch [33], DeBoor [11], and Wahba [43] for further details and Kimeldorf and Wahba [20] and Wahba [42] for an elegant Bayesian motivation for this approach.

We may rewrite $L[g]$ slightly, employing the so-called Dirac delta function $\delta_\alpha(x) = \delta_0(x - \alpha)$, which assigns mass one to the point α (see, e.g., Lighthill [25]), as

$$L[g] = n^{-1}\sum_{i=1}^{n}\int (y_i - g(x))^2\delta_{x_i}(x)\,dx + \lambda\int (g''(x))^2\,dx.$$

The Euler–Lagrange condition for this minimization problem is simply

$$-n^{-1}\sum_{i=1}^{n}(y_i - g(x))\delta_{x_i}(x) + \lambda g^{(4)}(x) = 0. \tag{4}$$

Since $\delta_{x_i}(x) = 0$ almost everywhere, we may conclude immediately that any solution $\hat{g}$ is piecewise cubic since $g^{(4)}$ vanishes almost everywhere. Integrating (4) from $-\infty$ to $x_i\pm$ yields

$$-n^{-1}\sum_{j \le i}(y_j - g(x_j)) + \lambda\, g^{(3)}(x_i+) = 0$$

and

$$-n^{-1}\sum_{j < i}(y_j - g(x_j)) + \lambda\, g^{(3)}(x_i-) = 0,$$

and subtracting we have

$$g^{(3)}(x_i+) - g^{(3)}(x_i-) = (y_i - g(x_i))/n\lambda.$$

We may fully characterize the solution as, taking the form,

$$\hat{g}(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \tag{5}$$

for $x \in [x_i,\ x_{i+1}]$, $i = 1, \ldots, n - 1$, with

$$\hat{g}^{(k)}(x_i+) - \hat{g}^{(k)}(x_i-) = \begin{cases} 0 & \text{if } k = 0, 1, 2, \\ (yi - g(x_i))/n\lambda & \text{if } k = 3, \end{cases} \tag{6}$$

in which $\hat{g}^{(k)}(x_i\pm) = \lim_{h\to 0}\hat{g}^{(k)}(x_i \pm h)$. Note that there are $4n$ such (linear) constraints and $4(n - 1)$ unknowns in (5) plus the 4 unknowns a_0, b_0, a_n, b_n characterizing the estimate $\hat{g}$ outside the interval $[x_{(1)},\ x_{(n)}]$. (Clearly $c_0 = d_0 = c_n = d_n = 0$. Since were they not, the penalty could be reduced without disturbing the first term of the objective function.) Thus (6) provides $4n$ equations in $4n$ unknowns.

Alternatively, one may use the first $3n$ equations in (6) to express the objective function (3) in n-unknowns and solve explicitly. This is the approach adopted by Reinsch [33] and we briefly sketch it here.

Suppose $0 \le x_1 < x_2 < \cdots < x_n \le 1$, and let $h_i = x_{i+1} - x_i$. Then (6) with $k = 0, 1, 2$ implies, for $i = 1, \ldots, n - 1$,

$$\begin{aligned} a_{i+1} - a_i &= b_ih_i + c_ih_i^2 + d_ih_i^3, \\ b_{i+1} - b_i &= 2c_ih_i + 3d_ih_i^2, \\ c_{i+1} - c_i &= 3d_ih_i. \end{aligned}$$

Substituting to eliminate the b's and d's yields

$$\frac{a_{i+1}-a_i}{h_i}-\frac{a_i-a_{i-1}}{h_i}=c_{i+1}\left(\frac{h_i}{3}\right)+\frac{2}{3}c_i(h_i+h_{i-1})+c_{i-1}\left(\frac{h_{i-1}}{3}\right),$$

which can be written as

$$Q'\mathbf{a}=R\mathbf{c}, \tag{7}$$

where
$\mathbf{a}=(a_1,\ldots,a_n)'$;
$\mathbf{c}=(c_2,\ldots,c_{n-1})'$;
$R=$ an $(n-2)^2$ tridiagonal matrix with entries:
$r_{i,i}=2(h_i+h_{i+1})/3$, $r_{i,i+1}=r_{i+1,i}=h_{i+1}/3$; for $i=1,\ldots,n-3$,
$r_{n-2,n-2}=2(h_{n-2}+h_{n-1})/3$;
$Q=$ an $n\times(n-2)$ banded matrix:
$q_{i,i}=1/h_i$, $q_{i+1,i}=-(1/h_i+1/h_{i+1})$, $q_{i+2,i}=1/h_{i+1}$, for $i=1,\ldots,n-2$.

It is easily seen (e.g., DeBoor [11]) that the penalty may be expressed as

$$\int (g''(x))^2\,dx=\mathbf{c}'R\mathbf{c},$$

so the original problem may be reformulated as minimizing

$$\begin{aligned} n\,L[g] &= (\mathbf{y}-\mathbf{a})'(\mathbf{y}-\mathbf{a})+n\lambda\mathbf{c}'R\mathbf{c}\\ &= (\mathbf{y}-\mathbf{a})'(\mathbf{y}-\mathbf{a})+n\lambda\mathbf{a}'QR^{-1}Q'\mathbf{a}, \end{aligned}$$

which has the solution

$$\hat{\mathbf{a}}=(I+n\lambda QR^{-1}Q')^{-1}\mathbf{y}. \tag{8}$$

2.2. Score estimator. Now using a similar approach, the objective function (2) of the score function estimation problem may be rewritten as

$$L[\psi]=n^{-1}\sum_{i=1}^{n}\int(\psi^2(x)-2\psi'(x))\delta_{x_i}(x)\,dx+\lambda\int(\psi''(x))^2\,dx.$$

The new Euler–Lagrange condition is

$$n^{-1}\sum_{i=1}^{n}(\psi(x)\delta_{x_i}(x)+\delta_{x_i}'(x))+\lambda\,\psi^{(4)}(x)=0.$$

Again the solution is clearly piecewise cubic with the form

$$\hat{\psi}(x)=a_i+b_i(x-x_i)+c_i(x-x_i)^2+d_i(x-x_i)^3$$

for $x\in[x_i,\,x_{i+1}]$, $i=1,\ldots,n-1$. Integrating as before we may compute the jump in the third derivative as

$$\psi^{(3)}(x_i+)-\psi^{(3)}(x_i-)=-\psi(x_i)/n\lambda.$$

But the $\delta'(x)$ term introduces a new complication. Integrating once from $-\infty$ to z gives

$$n^{-1}\sum_{j:\,x_j\le z}\psi(x_j)+n^{-1}\sum_{j=1}^{n}\delta_{x_j}(z)+\lambda\psi^{(3)}(z)=0,$$

and integrating again from $-\infty$ to $x_i\pm$ yields

$$K + n^{-1}i + \lambda\psi^{(2)}(x_i+) = 0,$$

$$K + n^{-1}(i - 1) + \lambda\psi^{(2)}(x_i-) = 0,$$

so, analogous to (6), we have

$$\hat{\psi}^{(k)}(x_i+) - \hat{\psi}^{(k)}(x_i-) = \begin{cases} 0 & \text{if } k = 0, 1, \\ -\frac{1}{n\lambda} & \text{if } k = 2, \\ -\frac{\psi(x_i)}{n\lambda} & \text{if } k = 3. \end{cases} \tag{9}$$

The jump in the second derivative is the consequence of the δ' term. We now have, using the prior constraints for $k = 0, 1, 2$ as before, a modified version of (7),

$$Q'\mathbf{a} = R\mathbf{c} + \mathbf{r}/2n\lambda, \tag{10}$$

along with

$$\mathbf{b} = A\mathbf{a} - C\mathbf{c} + \mathbf{s}/n\lambda, \tag{11}$$

$$\mathbf{d} = D\mathbf{c} + \mathbf{t}/6n\lambda, \tag{12}$$

where
$\mathbf{b} = (b_1, \ldots, b_n)'$;
$\mathbf{d} = (d_1, \ldots, d_{n-1})'$;
A = an $n \times n$ banded matrix:
$a_{i,i} = -1/h_i$, $a_{i,i+1} = 1/h_i$ for $i = 1, \ldots, n-1$,
$a_{n,n} = 1/h_{n-1}$, $a_{n,n-1} = -1/h_{n-1}$;
C = an $n \times (n-2)$ banded matrix:
$c_{i,i} = h_i/3$, $c_{i,i-1} = 2h_i/3$ for $i = 2, \ldots, n-2$,
$c_{1,1} = h_1/3$, $c_{n-1,n-2} = 2h_{n-1}/3$, $c_{n,n-2} = -h_{n-1}/3$;
D = an $(n-1) \times (n-2)$ banded matrix:
$d_{i,i} = -1/3h_i$, $d_{i,i+1} = 1/3h_i$ for $i = 1, \ldots, n-2$;
$\mathbf{r}$ = an $(n-2)$-vector independent of parameters:
$r_1 = (h_1 + h_2)/3$, $r_i = (2h_i + h_{i+1})/3$ for $i = 2, \ldots, n-2$;
$\mathbf{s}$ = an n-vector independent of parameters:
$s_1 = -h_1/6$, $s_i = h_i/6$, for $i = 2, \ldots, n-1$, $s_n = -h_{n-1}/3$;
$\mathbf{t}$ = an $(n-1)$-vector independent of parameters:
$t_1 = 2/h_1$, $t_i = 1/h_i$ for $i = 2, \ldots, n-1$.
The jump in the second derivative also necessitates a modification of the penalty, which now becomes

$$\int (\psi''(x))^2\, dx = \mathbf{c}'R\mathbf{c} + r'\mathbf{c}/n\lambda + K,$$

where K will henceforth denote a generic constant independent of the parameters $(\mathbf{a}', \mathbf{b}', \mathbf{c}', \mathbf{d}')$ of $\hat{\psi}$.

We may now write the objective function as

$$\begin{aligned} nL[\psi] &= (\mathbf{a}'\mathbf{a} - 2\mathbf{1}'\mathbf{b}) + n\lambda\mathbf{c}'R\mathbf{c} + \mathbf{r}'\mathbf{c} + K \\ &= \mathbf{a}'(I + n\lambda QR^{-1}Q')\mathbf{a} - 2\mathbf{a}'(A - CR^{-1}Q')'\mathbf{1} + K, \end{aligned}$$

where **1** is an n-vector of 1's. Minimizing with respect to **a** we have

$$\hat{\mathbf{a}} = (I + n\lambda QR^{-1}Q')^{-1}(A - CR^{-1}Q')'\mathbf{1}, \tag{13}$$

and using (10)–(12) we can obtain the solution for **b**, **c**, and **d**.

At first glance the situation with the Cox score estimator seems quite different from that in nonparametric regression. In nonparametric regression we directly observe y_i, which may be regarded as $g(x_i)$ plus noise. The estimator $\hat{g}$ constitutes a projection of observations vector into a subspace of $G_2[0, 1]$. We would like to answer the following question: "What quantity, if any, plays the role of the observations **y** in the score estimation problem?"

Comparing (13) with our prior nonparametric regression expression (8), we see that the term $(A - CR^{-1}Q')'\mathbf{1}$ appears to "play the role" of the observations **y** in nonparametric regression. We will refer to this vector as the pseudo-y or pseudo-observations of the sample $\mathbf{x} = (x_1, \ldots, x_n)'$.

Figure 1 is a plot of the pseudo-y against a grid of 50 equally spaced x_i's, which corresponds to the 50 equally spaced population quantiles of a uniform random variable. The pseudo-y tracks closely the true ψ function of the uniform random variable, which is the x-axis with positive spike at the right boundary and negative spike at the left, except a few end points. In Fig. 2, we generate the grid of **x** to be equally spaced population quantiles of a Cauchy, and plot the pseudo-y against it; the pseudo-observations again trace the true ψ function of a random Cauchy closely except for the few end points. The pseudo-y is perturbed violently, however, around the true ψ function if **x** is generated randomly. Figure 3 shows the situation of the pseudo-y that corresponds to the random sample **x**, which is the equally spaced sample quantiles, of a uniform density and we can see that the pseudo-y scatters wildly around the true ψ of the uniform. Hence, the pseudo-y is a useful piece of information if **x** reflects the true underlying density. The points near the two tails of the plots that deviate from the true ψ function are caused by the larger finite sample approximation error of the pseudo-y to the true ψ in the tails. If we generate a finer grid of equally spaced population quantiles of a uniform random variable by increasing the number of the quantiles, say to 100, the points of relatively huge deviations will approach the two ends of the support of the uniform distribution. This phenomenon is illustrated in Fig. 4. Discrepancies will, hence, only occur at the end points of the support asymptotically.

The above phenomenon provides us with a heuristic explanation of how the smoothing spline score estimator works and why it is consistent. We can write the ψ estimation model similar to the nonparametric regression case as

$$y_i = \psi_0(x_i) + \epsilon_i,$$

where ψ_0 is the true function to be estimated, y_i is the "observed" dependent variable derived from the sample **x**, and ϵ_i is some "noise." The smoothing spline score estimator projects the pseudo-y into the subspace of the Sobolev space, $H_2[a, b] = \{\psi : \psi, \psi' \text{ absolutely continuous and } \psi'' \in L_2[a, b]\}$. Given the pseudo-observations, the smoothing spline ψ estimator fits a curve through the pseudo-y that minimizes the mean-squared error between the pseudo-observations and the estimated ψ while subjecting it to a certain degree-of-roughness penalty. We know that the sample quantile is a strongly consistent estimator for the population quantiles (see Serfling [36]) and, hence, the pseudo-y will approach the true score function ψ_0 as the sample size approaches infinity. As a result, the optimal choice of λ tends to zero asymptotically.

2.3. Computationally efficient expressions. Cox [7] establishes properties of the smoothing spline ψ function estimator. His characterization does not, however, facilitate

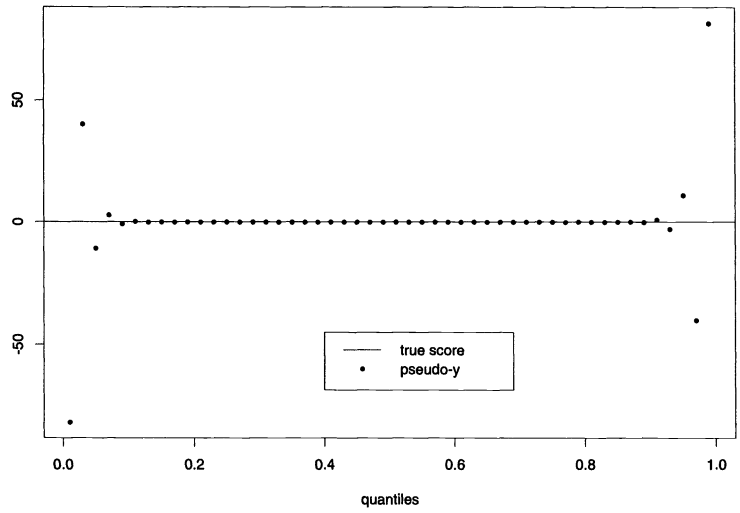


FIG. 1. *Pseudo-y of* 50 *equally spaced population quantiles of a uniform random variate.*

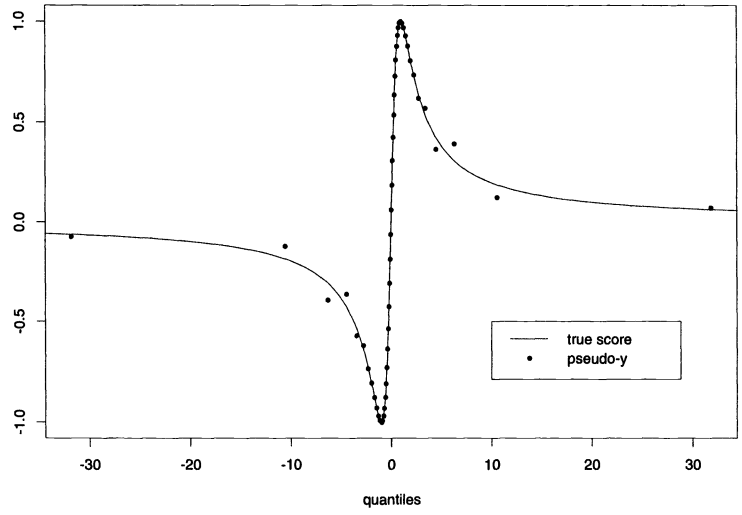


FIG. 2. *Pseudo-y of* 50 *equally spaced population quantiles of a Cauchy random variate.*

construction and implementation of the estimator. Our alternative characterization given in §2, however, allows us to implement an efficient computation of the smoothing spline $\hat{\psi}$.

If we substitute (10) into (13), writing $\mathbf{y} = (A - CR^{-1}Q')'\mathbf{1}$, we get

$$(I + n\lambda Q'QR^{-1})Q'\hat{\mathbf{a}} = Q'\mathbf{y},$$

$$\hat{\mathbf{c}} = (R + n\lambda Q'Q)^{-1}Q'\mathbf{y} - R^{-1}\mathbf{r}/2n\lambda. \tag{14}$$

Similarly, the $\hat{\mathbf{a}}$ can alternatively be expressed as:

$$(I + n\lambda Q'R^{-1}Q')\hat{\mathbf{a}} = \mathbf{y},$$

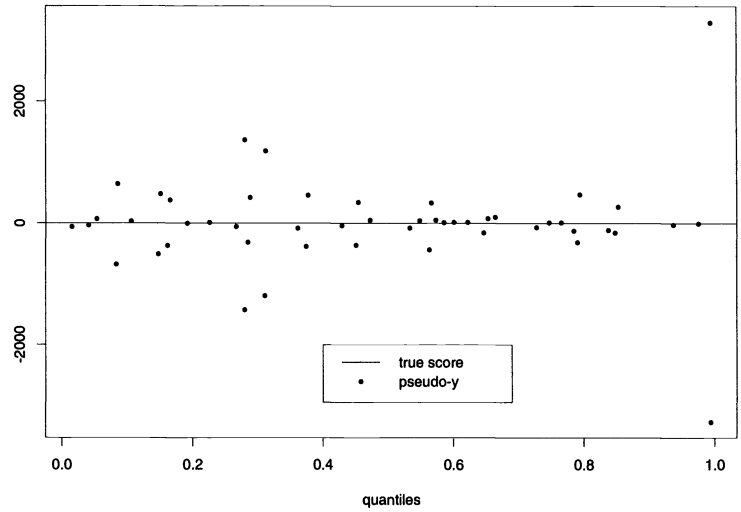


FIG. 3. *Pseudo-y of* 50 *random observations from a uniform distribution.*

$$\hat{\mathbf{a}} + n\lambda Q' R^{-1}(R\hat{\mathbf{c}} + \mathbf{r}/2n\lambda) = \mathbf{y},$$

$$\hat{\mathbf{a}} = \mathbf{y} - n\lambda Q'\hat{\mathbf{c}} - n\lambda Q' R^{-1}\mathbf{r}/2n\lambda.$$

Using (14) and simplifying yields

$$\begin{aligned} \hat{\mathbf{a}} &= (I - Q'(\mu R + Q'Q)^{-1}Q')\mathbf{y} \\ &= H(\mu)\mathbf{y}, \end{aligned} \tag{15}$$

where $\mu = \frac{1}{n\lambda}$.

Notice that the banded structures of the linear systems to be solved in (14) and (15) are preferable to (13). More efficient algorithms can be achieved through employing subroutines written specifically for banded matrices. The PORT library [13] written at Bell Laboratories and the International Mathematical and Statistical Libraries [19] both have collections of such routines, which can be readily called by programs written in FORTRAN. With $\hat{\mathbf{a}}$ and $\hat{\mathbf{c}}$ given by (15) and (14), respectively, efficient computations of $\hat{\mathbf{b}}$ and $\hat{\mathbf{d}}$ can be obtained using (11) and (12), respectively (see Ng [30] for FORTRAN source).

Note that the $H(\mu)$ matrix in (15) resembles the "hat" matrix in the theory of linear regression. The trace of the matrix, $\mathrm{Tr}(H(\mu))$, varies from 2 when $\mu = 0$ to n as $\mu \to \infty$. Hence, $\mathrm{Tr}(H(\mu))$ can be interpreted loosely as the "effective dimensionality" of $\hat{\psi}$ for the particular choice of μ or λ. Buja, Hastie, and Tibshirani [4] contains a thorough discussion on the interpretation of the "hat" matrix in the smoothing spline context. The role of this "effective dimensionality" will be explored in more detail in the next section when the choice of the smoothing parameter is considered.

3. The choice of the smoothing parameter. Any kernel based estimator faces the problem of having to choose a bandwidth parameter; the smoothing spline $\hat{\psi}$ has the penalty parameter λ to choose. Recalling from the previous section that the score estimation problem can be reduced to a nonparametric regression setting, it is natural to consider the various data-driven choice criteria commonly used in nonparametric curve fitting.

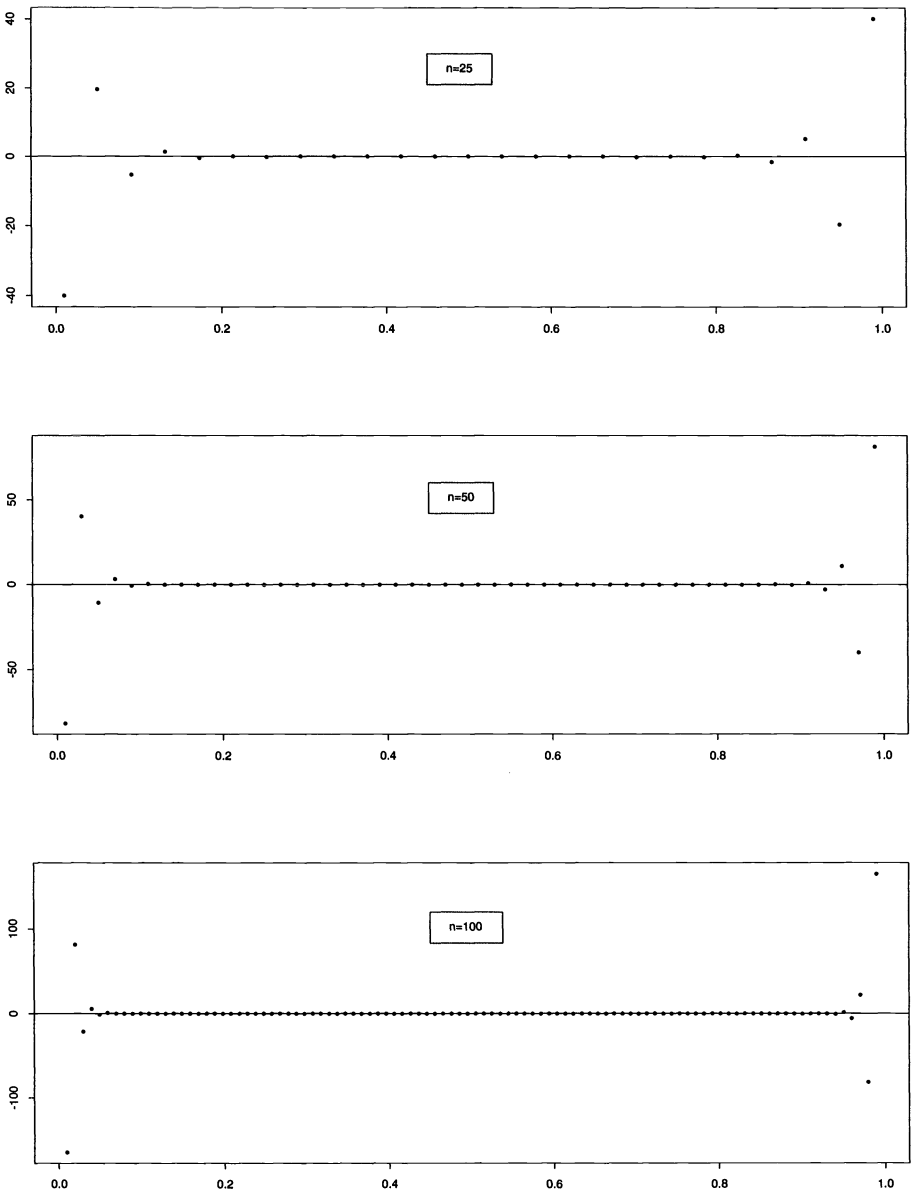


FIG. 4. *Pseudo-y of various sizes of equally spaced population quantiles of a uniform random variate.*

Conventional model selection criteria for the linear regression model involve minimizing various versions of

$$-L_j + C_j(n), \tag{16}$$

where L_j is the log likelihood function for the jth model evaluated at the maximum likelihood estimated parameters, and $C_j(n)$ is a penalty term, usually a function of the sample size n and the dimension k_j of the jth model. Some commonly used penalty terms are $C_j(n) = k_j$ for the Akaike information criterion [1], $C_j(n) = k_j/2 \log n$ for the Schwarz Bayesian information criterion [35], $C_j(n) = k_j \log\log n$ for the Hannan and Quinn criterion [15],

$C_j(n) = \frac{n}{2}\log(n + k_j)/(n - k_j)$ for the Akaike finite prediction error (FPE) criterion [1], $C_j(n) = \frac{n}{2}\log(n+2k_j)$ for the Shibata criterion [37], and $C_j(n) = -n\log(1 - k_j/n)$ is the generalized cross-validation criterion first proposed by Craven and Wahba [9]. All these variations of (16) are some form of estimate for the risk or prediction risk of the estimator. Defining the sample mean-squared error of the classical linear regression model as

$$\text{MSE} = n^{-1}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

along with i.i.d. Gaussian errors assumption, (16) can be written as

$$\frac{n}{2}\log \text{MSE} + C_j(n). \tag{17}$$

A natural extension of model selection for linear regression to λ selection for nonparametric curve fitting is to substitute $\text{Tr}(H(\lambda))$, the "effective dimensionality" determined by λ, of the response function, $\hat{g}(x_i)$, for k_j into (17). A further extension to the choice of λ for the ψ function estimation problem is to substitute pseudo-y for the dependent observations, $\hat{\psi}$ for $\hat{g}$, and $\text{Tr}(H(\lambda))$ for k_j into (17).

We have experimented with the Mallow C_p, generalized cross-validation (GCV) and various information criteria as well. None of them, however, performs satisfactorily. All these criteria almost always lead to oversmoothing, i.e., picking too large a λ (see Ng [29]). As we notice from Fig. 3, the pseudo-y fluctuates wildly around the true ψ, with extremely low signal-to-noise ratio. This strongly suggests that the errors are generated from some heavy-tailed distribution instead of the implicitly assumed Gaussian one. It is well known that model selection criteria based on the assumption of Gaussian innovations perform badly under moderate departures from the Gaussian condition in the innovation process (see Machado [26]). Scrutinizing the expression $(A - CR^{-1}Q')'\mathbf{1}$ for the pseudo-y, we notice that y_i depends on the h_i's in a subtle way, and hence y_i and y_{i+1} are obviously dependent through their neighboring h_i's and so are u_i and u_{i+1}. This violates yet another standard assumption on the u_i's in conventional linear or nonparametric regression settings.

3.1. Adaptive information criteria. The evidence of extreme outliers and dependency of u_i's in the ψ function estimation problem suggests some form of robust model selection criterion should be adopted. Martin [28] and Ronchetti [34] both suggested a robust version of the Akaike information criterion by using the least favorable distribution for the errors, under which (16) becomes

$$\sum_{i=1}^{n}\rho(\hat{\epsilon}_i) + C_j(n),$$

where $\rho(\epsilon) = \epsilon^2/2$ if $|\epsilon| \le k$ and $\rho(\epsilon) = k|\epsilon| - \frac{k^2}{2}$ if $|\epsilon| > k$ (i.e., Huber's ρ [18], which associates with a distribution that is Gaussian in the middle and Laplace in the tails). Machado [26] also suggested a similar robustified version of the Schwarz information criterion, derived from a Bayesian model selection framework, using any conventional ρ function that defines an M-estimator.

Alternatively, since one can always recover the estimated density from the estimated ψ by exponentiating the negative integral of $\hat{\psi}$, it is possible, by using the estimated likelihood, to implement an information criterion that will adapt to the underlying data generating process.

To recover $\hat{f}$ from $\hat{\psi}$, let

$$\log \hat{f}(x) = -\int \hat{\psi}(x)\,dx = -I\hat{\psi}_i(x) - IC_i \quad \text{for } x \in [x_i, x_{i+1}],$$

where

$$I\hat{\psi}_i(x) = a_i(x - x_i) + \frac{b_i}{2}(x - x_i)^2 + \frac{c_i}{3}(x - x_i)^3 + \frac{d_i}{4}(x - x_i)^4$$

are the integrals of $\hat{\psi}$, and IC_i are the integrating constants in each interval $[x_i, x_{i+1}]$. We now have,

$$\begin{aligned}
&\text{for } x \in [-\infty, x_1],\\
&\log \hat{f}(x) = -\left[a_0(x - x_1) + \frac{b_0}{2}(x - x_1)^2 + \frac{c_0}{3}(x - x_1)^3 + \frac{d_0}{4}(x - x_1)^4 + IC_0\right],\\
&\log \hat{f}(x_1) = -IC_0,\\
&\qquad\vdots\\
&\text{for } x \in [x_i, x_{i+1}] \ i = 1, 2, \ldots, n-1,\\
&\log \hat{f}(x) = -\left[a_i(x - x_i) + \frac{b_i}{2}(x - x_i)^2 + \frac{c_i}{3}(x - x_i)^3 + \frac{d_i}{4}(x - x_i)^4 + IC_i\right],\\
&\log \hat{f}(x_i) = -IC_i,\\
&\log \hat{f}(x_{i+1}) = -\left[a_i h_i + \frac{b_i}{2}h_i^2 + \frac{c_i}{3}h_i^3 + \frac{d_i}{4}h_i^4 + IC_i\right],\\
&\qquad\vdots\\
&\text{for } x \in [x_n, \infty],\\
&\log \hat{f}(x) = -\left[a_n(x - x_n) + \frac{b_n}{2}(x - x_n)^3 + \frac{d_n}{4}(x - x_n)^4 + IC_n\right],\\
&\log \hat{f}(x_n) = -IC_n.
\end{aligned}$$

We have $(n+1)$ integrating constants, including those of the two tail portions. There are n continuity constraints on $\log \hat{f}(x)$ at the n observed quantiles of F_n, which give

$$\begin{gathered}
IC_1 = IC_0,\\
\vdots\\
IC_{i+1} = a_i h_i + \cdots + \tfrac{d_i}{4}h_i^4 + IC_i = \textstyle\sum_{j=1}^{i}\left(a_j h_j + \cdots + \tfrac{d_j}{4}h_j^4\right) + IC_0,\\
\vdots\\
IC_n = a_{n-1}h_{n-1} + \cdots + \tfrac{d_{n-1}}{4}h_{n-1}^4 + IC_{n-1} = \textstyle\sum_{j=1}^{n-1}\left(a_j h_j + \cdots + \tfrac{d_j}{4}h_j^4\right) + IC_0.
\end{gathered}$$

Imposing $\int \hat{f}(x)\,dx = 1$, we determine the integrating constant IC_0 from

$$\begin{aligned}
1 &= \int_{-\infty}^{+\infty} \hat{f}(x)\,dx\\
&= \int_{-\infty}^{x_1} \exp\left(-\left[a_0(x - x_1) + \cdots + \frac{d_0}{4}(x - x_1)^4 + IC_0\right]\right) dx + \cdots\\
&\quad + \int_{x_i}^{x_{i+1}} \exp\left(-\left[a_i(x - x_i) + \cdots + \frac{d_i}{4}(x - x_i)^4\right.\right.
\end{aligned}$$

$$(18)\qquad + \sum_{j=1}^{i-1}\left(a_j h_j + \cdots + \frac{d_j}{4}h_j^4\right) + IC_0\Bigg]\Bigg)\,dx + \cdots$$

$$+ \int_{x_n}^{\infty} \exp\left(-\left[a_n(x-x_n) + \cdots + \frac{d_n}{4}(x-x_n)^4\right.\right.$$

$$\left.\left. + \sum_{j=1}^{n-1}\left(a_j h_j + \cdots + \frac{d_j}{4}h_j^4\right) + IC_0\right]\right)\,dx.$$

Letting I_0 be the first term, I_n be the last term of (18), and factoring out $\exp(-IC_0)$ gives us

$$[1 - I_0 - I_n]\,\exp(IC_0) = \int_{x_1}^{x_2} \exp\left(-I\hat{\psi}_1(x)\right)\,dx + \cdots$$

$$(19)\qquad + \exp\left(-\sum_{j=1}^{i-1}\left(a_j h_j + \cdots + \frac{d_j}{4}h_j^4\right)\right)\int_{x_i}^{x_{i+1}} \exp\left(-I\hat{\psi}_i(x)\right)\,dx + \cdots$$

$$+ \exp\left(-\sum_{j=1}^{n-2}\left(a_j h_j + \cdots + \frac{d_j}{4}h_j^4\right)\right)\int_{x_{n-1}}^{x_n} \exp(-I\hat{\psi}_{n-1}(x))\,dx.$$

With IC_0 obtained from (19), the rest of the IC_i can be calculated through recursion and we have the estimated density as

$$(20)\qquad \hat{f}_{\lambda_j}(x) = \exp(-I\hat{\psi}_i(x) - IC_i) \quad \text{for } x \in [x_i, x_{i+1}] \quad \text{and} \quad i = 0, \ldots, n$$

with $x_0 = -\infty$ and $x_{n+1} = \infty$. The subscript of $\hat{f}$ in (20) merely reminds that the estimated density is a function of λ through the coefficients of $\hat{\psi}$. The adaptive information criterion to be minimized takes the form

$$(21)\qquad -\sum_{i=1}^{n} \log \hat{f}_{\lambda_j}(x_i) + C_{\lambda_j}(n) = \sum_{i=1}^{n} IC_i + C_{\lambda_j}(n).$$

Note that $C_{\lambda_j}(n)$ is a function of n as well as λ, which determines the "effective dimension" of the chosen jth model through the $H(\mu)$ matrix in (15).

In terms of calculating IC_0 from (19), there is unfortunately no closed form for the expression $\int_{x_i}^{x_{i+1}} \exp(-I\hat{\psi}_i(x))\,dx$ so we must resort to numerical integration.

3.2. The Dirac delta disaster. When applying traditional maximum likelihood methods to density estimation, the likelihood function can typically be made arbitrarily large as the estimated density approaches the (Dirac) delta function with point mass at each observation. This can be accomplished by letting the window width tend to zero when the density is estimated by kernel methods. Therefore, when likelihood methods are used to pick the smoothing parameter of a kernel based estimate, the likelihood function must be cross validated as

$$CV(h) = n^{-1}\sum_{i=1}^{n} \log \hat{f}_{-i}(x_i),$$

where $\hat{f}_{-i}(x_i)$ is the estimated density evaluated at x_i using all data except x_i. We might, therefore, be tempted to think that the smoothing parameter λ of the smoothing spline score estimator chosen by the adaptive information criteria using the estimated likelihood will also tend to zero and produce an $\hat{f}$ tending to a sum of Dirac delta functions with point mass at the observed quantiles. It turns out this is not true.

The inherent "smoothness" of the smoothing spline score estimator given by the condition in (9) guarantees that the estimator is continuous with a continuous first derivative. Even for $\lambda = 0$, we will have a "rough" interpolating cubic spline that passes through every single point of the pseudo-y but satisfies these continuity conditions. Thus we never encounter the Dirac delta disaster.

Our experience with the adaptive information criteria using the Akaike, Schwarz, and Shibata penalty shows that they all perform better than conventional information criteria using Gaussian innovation assumption. Conventional data-driven information choice criteria lead to choosing, most of the time, too large a λ. This is equivalent to over-smoothing the score estimate. Monte-Carlo results are reported in the next section.

4. Comparative performance of various score estimators. In this section, we try to answer the question: "How well do various score function estimators perform in finite sample situations?" The criterion we used in comparing performance is the integrated squared error (ISE) of an estimate, $\hat{\psi}$, expressed as

$$ISE(\hat{\psi}, \psi_0) = \int_a^b (\hat{\psi}(x) - \psi_0(x))^2 f_0(x)\, dx. \tag{22}$$

The integral in (22) will be computed through numerical integration.

Since we are looking solely at continuous random variables having support on the whole real line or the positive half line, we must pick the lower and upper bounds for the numerical integration. It is obvious that estimates may have unbounded error in the tail where density is low; we, therefore, pick the lower and upper bounds to be the first and last of the 10%-trimmed ordered observations, respectively. The purpose of the trimming is to ensure that the squared error function will be bounded from above within these limits so that numerical integrations computed will be reliable. This also irons out some wrinkles in the ISE loss function and ensures that the numerical optimization routine searching for its minimum will converge properly to a global minimum.

4.1. Estimators studied. Three estimators for the ψ function are considered in our Monte-Carlo simulations. The first is the adaptive kernel estimator based on the adaptive kernel estimate of a density function introduced in Silverman [39]. No data-driven cross-validation choice of window width is required. The second estimator considered is the fixed-window kernel estimator with weight function suggested by Cox and Martin [8]. They show it to have a faster convergence rate than conventional kernel estimate if the weight function is chosen properly to localize the integrated squared error loss function. It is, however, more expensive than the adaptive kernel estimator for we have to use cross-validation to pick both window widths. (Details on the construction of the weighted kernel score estimator is given in the Appendix.) Finally, the primary focus of the simulation is to evaluate the performance of the smoothing spline score estimator.

4.2. Measurement of efficiency. Let λ_0 be the *optimal* smoothing parameter of the smoothing spline estimator that minimizes (22) over the parameter space $\Lambda \subset R_+$, let λ^*_{ind} be the adaptive information criterion choice of $\lambda \in \Lambda$ using a specific penalty for $C_{\lambda_j}(n)$ in (21),

with $ind = sch$ the λ chosen using Schwarz's penalty, $ind = aki$ the λ chosen with Akaike's penalty, and $ind = shi$ the λ chosen using Shibata's penalty,[1] then the relative efficiencies of the various score estimators are measured by

$$reff(\hat{\psi}_{IND}) = \frac{ISE(\hat{\psi}_{spl}(\lambda_0), \psi_0)}{ISE(\hat{\psi}_{IND}, \psi_0)},$$

where $\hat{\psi}_{spl}(\lambda_0)$ is the smoothing spline score estimator with the optimal smoothing parameter λ_0. IND is the $\hat{\psi}$ indicator with $IND = spl(\lambda^*_{ind})$ referring to the smoothing spline score estimator, using λ^*_{ind}, $IND = ake$ referring to the adaptive kernel score estimator, and $IND = wke$ referring to the weighted kernel score estimator. The relative efficiency $reff(\hat{\psi}_{spl}(\lambda^*_{ind}))$ measures how well the adaptive information criterion using the ind penalty performs in choosing the smoothing parameter for the smoothing spline estimator while $reff(\hat{\psi}_{ake})$ and $reff(\hat{\psi}_{wke})$ measure how well the adaptive kernel estimator and the weighted single kernel estimator perform relative to the smoothing spline estimator with the optimal smoothing parameter λ_0. A value of $reff(\hat{\psi}_{spl}(\lambda^*_{ind})) = 1$ means that the adaptive information criterion using ind penalty has picked the optimal λ_0 while either $reff(\hat{\psi}_{ake}) > 1.0$ or $reff(\hat{\psi}_{wke}) > 1.0$ means that the adaptive kernel or the weighted kernel estimator performs better than the smoothing spline estimator with the *optimal* λ for a particular realization of random sample.

4.3. Monte-Carlo design. Each Monte-Carlo trial consisted of 100 replications. The sample sizes were $n = 25$, $n = 50$, $n = 100$, and $n = 250$. The distributions were: (1) the standard Gaussian, N(0,1); (2) Cauchy with location parameter 0 and scale parameter 1, C(0,1); (3) lognormal with scale and shape parameters 1, L(0,1); (4) mixture of Gaussians, 0.5*N(−3,1)+0.5*N(3,1), NMIX(−3,1,3,1,.5); (5) beta with scale parameter and shape parameter 7, B(7,7); and (6) beta mixture in the form of .6*B(12,7)+.4*B(3,11), BMIX(12,7,3,11,.6). The random number generator was the Marsaglia "Super-Duper" generator implemented in the PORT3 library (Fox [13]).

4.4. Monte-Carlo results. The results of the Monte-Carlo simulation are reported in Tables 1–4. Computations were conducted using FORTRAN algorithms written for the S environment (Becker, Chambers, and Wilks [2]). The search for the optimal λ_0 in (22) was performed by the numerical optimization routine MNF (Gay [14]) available in the PORT3 library [13]. Occasionally the ISE function would have multiple minima and the optimization routine would then stop at a local minimum instead of the global minimum. This would give rise to $reff(\hat{\psi}_{spl}(\lambda^*_{ind})) > 1.0$ even though, in most cases, the ISE was a smooth function with unique global minimum. It is also possible that in any single realization of the 100 Monte-Carlo replications, the adaptive kernel estimator or the weighted single kernel estimator can outperform the optimal cubic spline estimator and gives rise to $reff(\hat{\psi}_{ake}) > 1.0$ or $reff(\hat{\psi}_{wke}) > 1.0$. This, of course, does not imply they are superior to $\hat{\psi}_{spl}(\lambda_0)$. Thus we will wish to look at some summary statistics of the 100 relative efficiencies for each estimator.

Since sample variability of the relative efficiencies are quite large in simulations involving relatively small sample sizes, we decide to concentrate on robust summary statistics. The medians along with the interquartile ranges (in parentheses) of the relative efficiencies of various estimators are reported in Tables 1–4. For the sake of completeness, we also report the sample means and sample standard deviations of various estimators.

[1] A referee has suggested that leave-one-out estimators of both the ISE and truncated ISE be incorporated into the simulation as alternative schemes of choosing the smoothing parameter λ. However, these are extremely expensive to implement. Nevertheless, we agree that with the rapidly changing computing capabilities they may soon be viable alternatives and should be a valuable future research topic.

TABLE 1

Relative efficiencies of various ψ estimators; $n = 25$. The numbers in each entry for every estimator under every distribution are: (i) *median*, (ii) (*1st quartile*, *3rd quartile*), (iii) *mean, and* (iv) *standard deviation.*

Distributions	Estimators				
	ake	wke	$spl(\lambda^*_{sch})$	$spl(\lambda^*_{aki})$	$spl(\lambda^*_{shi})$
N(0,1)	.24 (.14 , .64) .98 2.45	.20 (.07 , .49) .67 2.16	1 (.99 , 1.00) .93 .21	1 (.98 , 1.00) .86 .32	.99 (.62 , 1.00) .80 .37
C(0,1)	.48 (.27 , .68) .64 .73	.52 (.33 , .68) .64 .75	.41 (.31 , .71) .50 .31	.55 (.24 , .80) .55 .32	.63 (.23 , .84) .57 .35
L(0,1)	1.37 (1.07 , 1.70) 1.37 .55	1.36 (1.09 , 1.79) 1.47 .64	.91 (.69 , .96) .81 .22	.82 (.56 , .92) .72 .27	.69 (.11 , .86) .55 .37
NMIX(-3,1, 3,1,.5)	.32 (.25 , .38) .35 .14	.83 (.62 , 1.04) .98 .67	.89 (.70 , .96) .82 .18	.87 (.66 , .96) .79 .22	.83 (.56 , .94) .72 .29
B(7,7)	.67 (.15 , 1.46) 1.44 2.17	.33 (.09 , .79) 3.33 18.52	1.00 (1.00 , 1.00) .99 .01	1.00 (1.00 , 1.00) .95 .17	1.00 (.99 , 1.00) .92 .22
BMIX(12,7, 3,11,.6)	.34 (.20 , .44) .51 .51	.60 (.33 , 1.03) .94 .94	.72 (.23 , .94) .62 .33	.89 (.55 , .99) .78 .26	.91 (.53, .99) .89 .66

TABLE 2

Relative efficiencies of various ψ estimators; $n = 50$. The numbers in each entry for every estimator under every distribution are: (i) *median*, (ii) (*1st quartile*, *3rd quartile*), (iii) *mean, and* (iv) *standard deviation.*

Distributions	Estimators				
	ake	wke	$spl(\lambda^*_{sch})$	$spl(\lambda^*_{aki})$	$spl(\lambda^*_{shi})$
N(0,1)	.16 (.07 , .39) .45 .74	.13 (.07 , .31) .23 .24	1.00 (.99 , 1.00) .99 0.00	1.00 (1.00 , 1.00) .98 .13	.99 (.99 , 1.00) .94 .22
C(0,1)	.36 (.22 , .48) .38 .21	.47 (.28 , .61) .47 .23	.41 (.20 , .63) .44 .28	.62 (.26 , .87) .57 .33	.56 (.17 , .85) .52 .34
L(0,1)	1.17 (.60 , 1.64) 1.21 .67	1.06 (.63 , 2.1) 1.36 .96	.73 (.39 , .91) .67 .29	.63 (.37 , .87) .62 .30	.55 (.34 , .82) .54 .31
NMIX(-3,1, 3,1,.5)	.28 (.17 , .32) .26 .09	.64 (.44 , .90) .75 .44	.77 (.64 , .97) .77 .21	.84 (.62 , .96) .77 .25	.85 (.61 , .96) .74 .30
B(7,7)	.47 (.16 , 1.23) 1.10 1.58	.24 (.09 , .58) .85 1.89	1.00 (1.00 , 1.00) .99 .05	1.00 (1.00 , 1.00) .94 .20	1.00 (.99 , 1.00) .93 .23
BMIX(12,7, 3,11,.6)	.30 (.20, .35) .31 .17	.64 (.43 , .78) .67 .40	.67 (.38 , .87) .64 .28	.77 (.53 , .92) .71 .26	.82 (.54 , .95) .72 .27

TABLE 3

Relative efficiencies of various ψ estimators; $n = 100$. The numbers in each entry for every estimator under every distribution are: (i) *median*, (ii) (*1st quartile*, *3rd quartile*), (iii) *mean, and* (iv) *standard deviation.*

Distributions	Estimators				
	ake	wke	$spl(\lambda^*_{sch})$	$spl(\lambda^*_{aki})$	$spl(\lambda^*_{shi})$
N(0,1)	.14 (.04 , .20) .57 2.35	.09 (.03 , .18) .41 1.90	1.00 (.99 , 1.00) .99 0.05	1.00 (1.00 , 1.00) .94 .21	1.00 (1.00 , 1.00) .94 .21
C(0,1)	.29 (.19 , .46) .33 .21	.43 (.27 , .58) .47 .30	.34 (.17 , .52) .38 .25	.61 (.39 , .86) .62 .28	.61 (.31 , .90) .60 .31
L(0,1)	1.08 (.68 , 1.54) 1.17 .55	1.05 (.54 , 1.75) 1.24 .82	.60 (.39 , .79) .61 .26	.58 (.37 , .81) .60 .25	.59 (.33 , .84) .58 .28
NMIX(-3,1, 3,1,.5)	.20 (.15 , .26) .21 .08	.81 (.49 , .99) .84 .41	.84 (.74 , .94) .82 .15	.89 (.76 , .97) .82 .21	.89 (.68 , .96) .78 .26
B(7,7)	.32 (.11 , .64) .55 .78	.22 (.05 , .44) .34 .38	1.00 (.99 , 1.00) .99 .03	1.00 (1.00 , 1.00) .93 .23	1.00 (1.00 , 1.00) .92 .24
BMIX(12,7, 3,11,.6)	.20 (.15, .28) .23 .09	.62 (.43 , .72) .65 .33	.71 (.54 , .88) .72 .21	.87 (.67 , .97) .81 .20	.84 (.64 , .95) .77 .23

TABLE 4

Relative efficiencies of various ψ estimators; $n = 250$. The numbers in each entry for every estimator under every distribution are: (i) *median*, (ii) (*1st quartile*, *3rd quartile*), (iii) *mean, and* (iv) *standard deviation.*

Distributions	Estimators				
	ake	wke	$spl(\lambda^*_{sch})$	$spl(\lambda^*_{aki})$	$spl(\lambda^*_{shi})$
N(0,1)	.06 (.03 , .15) .13 .19	.10 (.01 , .19) .17 .23	1.00 (1.00 , 1.00) .97 .02	1.00 (1.00 , 1.00) .91 .28	1.00 (1.00 , 1.00) .91 .28
C(0,1)	.23 (.17 , .30) .24 .12	.37 (.24 , .51) .40 .17	.25 (.19 , .36) .30 .19	.64 (.45 , .85) .65 .25	.69 (.48 , .85) .69 .24
L(0,1)	.85 (.61 , 1.11) .98 .54	.96 (.56 , 1.41) 1.09 .63	.49 (.32 , .62) .51 .24	.66 (.45 , .81) .64 .22	.66 (.44 , .85) .65 .24
NMIX(-3,1, 3,1,.5)	.14 (.12 , .16) .14 .04	.71 (.50 , .95) .77 .39	.77 (.68 , .83) .76 .13	.88 (.72 , .97) .79 .25	.89 (.72 , .97) .79 .25
B(7,7)	.35 (.13 , .62) .64 .91	.19 (.06 , .41) .34 .43	1.00 (1.00 , 1.00) .98 .13	1.00 (1.00 , 1.00) .96 .19	1.00 (1.00 , 1.00) .96 .19
BMIX(12,7, 3,11,.6)	.13 (.10, .16) .14 .05	.48 (.34 , .63) .56 .35	.74 (.53 , .90) .72 .20	.83 (.65 , .95) .78 .22	.82 (.64 , .96) .77 .23

From Tables 1–4, we can see that the adaptive information choice criteria using either the Akaike or Shibata penalty performs better than that using the Schwarz penalty except in the case of extremely small sample size of 25. Since the Akaike penalty and the Shibata penalty are asymptotically equivalent, we will, henceforth, use $\hat{\psi}_{spl}(\lambda^*_{aki})$ to represent the class of smoothing spline score estimators in our relatively small sample simulations.

Let us first investigate the performances of various ψ estimators for the sample size $n = 100$. From Table 3, we can see that the clear winner is the smoothing spline score estimator $\psi_{spl}(\lambda^*_{aki})$. For the normal distribution, the smoothing spline estimator is the best. This is no surprise because the two tails of the $\hat{\psi}_{spl}(\lambda^*_{aki})$ are linear and the normal ψ function is also linear with slope equal to the reciprocal of the population standard deviation. In effect, the Cox estimator shrinks toward the Gaussian $\psi(x) = x$ function. Moreover, with λ^*_{aki} chosen to be as large as possible, the cubic spline estimator, $\hat{\psi}_{spl}(\lambda^*_{aki})$ becomes a linear function which minimizes the sum of squared distances between $\hat{\psi}_{spl}(x_i)$ and the "observations" the pseudo-y.

The poor performance of $\hat{\psi}_{ake}$ is due to the Cauchy kernel used, which always tends to bend $\hat{\psi}_{ake}$ back towards the horizontal axis. Figure 5 shows a typical result of the three estimators. The adaptive information criterion using the Akaike penalty is 100% efficient in picking the optimal smoothing parameter λ_0.

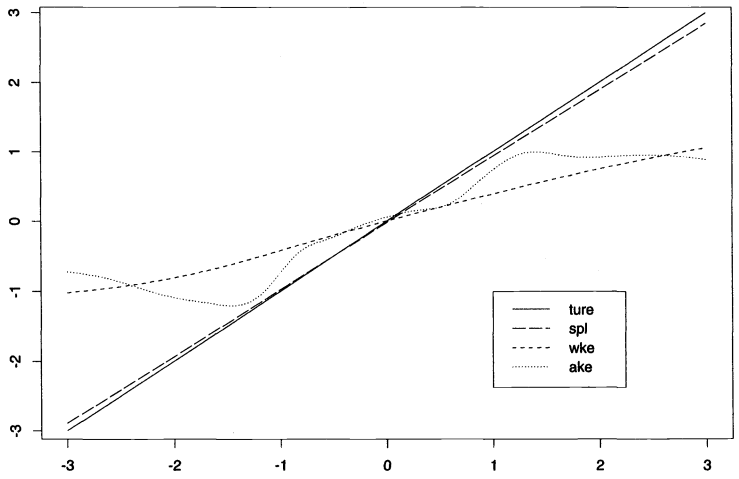


FIG. 5. *ψ_0 and $\hat{\psi}_{IND}$ of one Gaussian random sample of* 50 *observations.*

For the Cauchy distribution, the smoothing spline estimator, $\hat{\psi}_{spl}(\lambda^*_{aki})$ has the highest efficiency of 61%. The weighted kernel estimator, $\hat{\psi}_{wke}$, has the second highest efficiency of 43%. The adaptive kernel estimator has the lowest efficiency of 29%. The corresponding estimators from a particular realization of Cauchy random sample are plotted in Fig. 6.

What is astonishing is the lognormal case. The cubic spline estimator is the worst among the three. Even with the correct λ, it still only has about 95% the efficiency of $\hat{\psi}_{wke}$ and 93% of $\hat{\psi}_{akj}$. As it is illustrated by Fig. 7, in order for the cubic spline estimator to pick up the sharp curvature of ψ_0 in the lower portion of the lognormal distribution, the estimate needs to be quite rough and this will produce too rough a fit for the upper tail where observations are scarce and the pseudo-observations are highly contaminated.

Under the normal mixture distribution, the finalist is $\hat{\psi}_{spl}(\lambda^*_{aki})$. The adaptive kernel estimator is incapable of picking up the sharp turns of the ψ_0 as can be seen from Fig. 8.

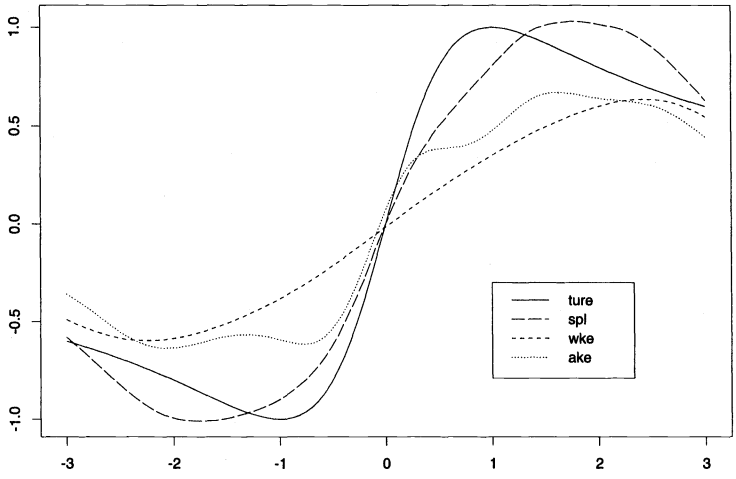


FIG. 6. ψ_0 *and* $\hat{\psi}_{IND}$ *of one Cauchy random sample of* 50 *observations.*

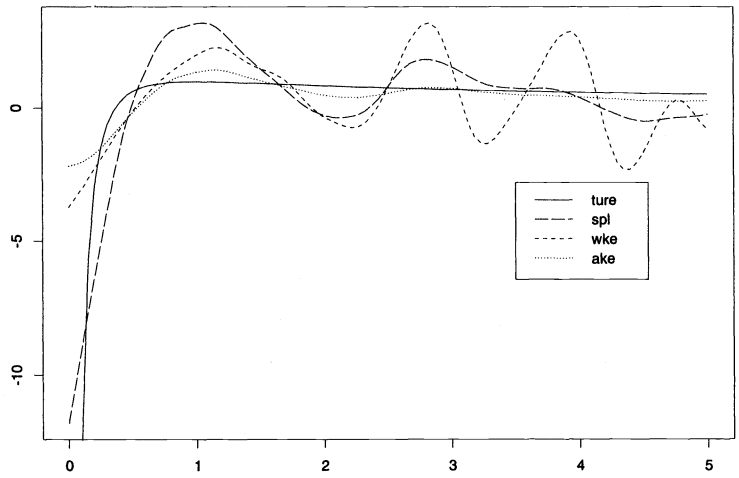


FIG. 7. ψ_0 *and* $\hat{\psi}_{IND}$ *of one lognormal random sample of* 50 *observations.*

Experimenting with smaller pilot window width for $\hat{\psi}_{ake}$ still does not help much. Again the Cauchy kernel used for $\hat{\psi}_{akj}$ contributes to its bad performance because it always tends to mimic the tails of a Cauchy ψ_0. Note also that the adaptive information criterion is quite successful in picking the optimal λ_0, close to 90% efficient, for this normal mixture distribution. We notice from Fig. 8 that $\hat{\psi}_{spl}(\lambda^*_{aki})$ is not quite able to pick up the sharp turn around the region $x \in [-1, 1]$. This, however, occurs at a low density section of the bimodal distribution and hence does not have a detrimental effect on ISE.

The obvious winner for the Beta distribution is also $\hat{\psi}_{spl}(\lambda^*_{aki})$ and the adaptive information criterion does an excellent job in picking the optimal λ_0, too. It is obvious from Fig. 9 that $\hat{\psi}_{wke}$ is fooled into thinking the density is a Gaussian type while $\hat{\psi}_{ake}$ has the unpleasant Cauchy tails again.

The clear choice for the Beta mixture is again $\hat{\psi}_{spl}(\lambda^*_{aki})$ having 87% efficiency. $\hat{\psi}_{wke}$ has about 62% efficiency while $\hat{\psi}_{akj}$ is much lower at 20%. The various score estimators for a particular realization are given in Fig. 10.

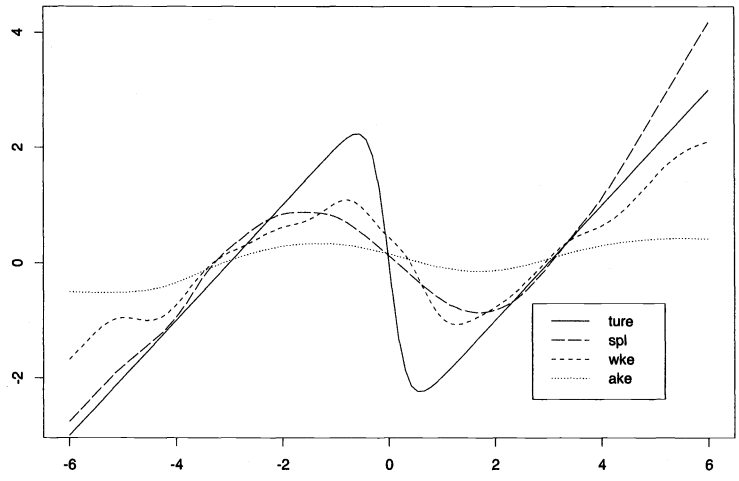


FIG. 8. ψ_0 *and* $\hat{\psi}_{IND}$ *of one normal mixture random sample of* 50 *observations.*

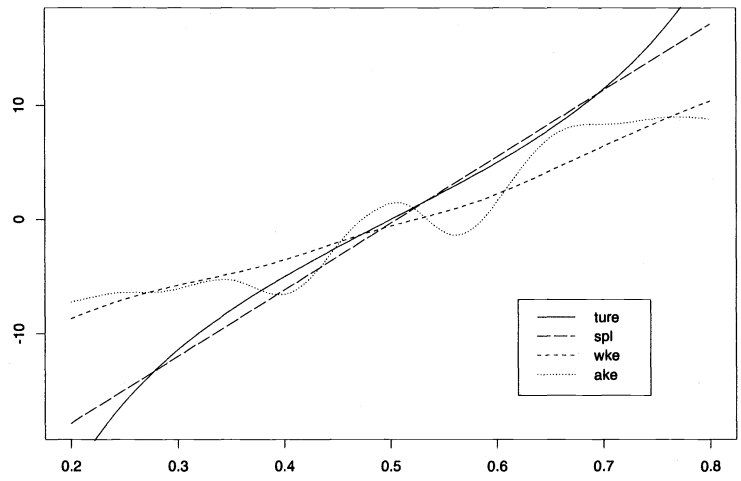


FIG. 9. ψ_0 *and* $\hat{\psi}_{IND}$ *of one beta random sample of* 50 *observations.*

Scrutinizing Tables 1, 2, and 4 reveals similar results. It is obvious from our Monte-Carlo simulations that good performance of either kernel estimator depends very much on using the correct kernel that reflects the underlying true distribution in addition to choosing the correct window width. This agrees with the view that kernel estimators are simply ad hoc estimators, which place mass around each observation. The right choice of kernel becomes even more important for observations in the tail where density is low since few observations will appear in the tail to help smooth things out. This sensitivity to correct kernel choice is further amplified in ψ function estimation where higher derivatives of the density are involved. The human eye may not be able to distinguish between the probability density functions of a normal tail and a Cauchy tail but the difference between the slopes of the tails of a normal ψ and a Cauchy ψ is too obvious to be overlooked.

Our results suggest that an estimator like $\hat{\psi}_{spl}$ that finds its theoretical justification from an explicit statistical decision criterion, in this case minimizing the integrated squared error,

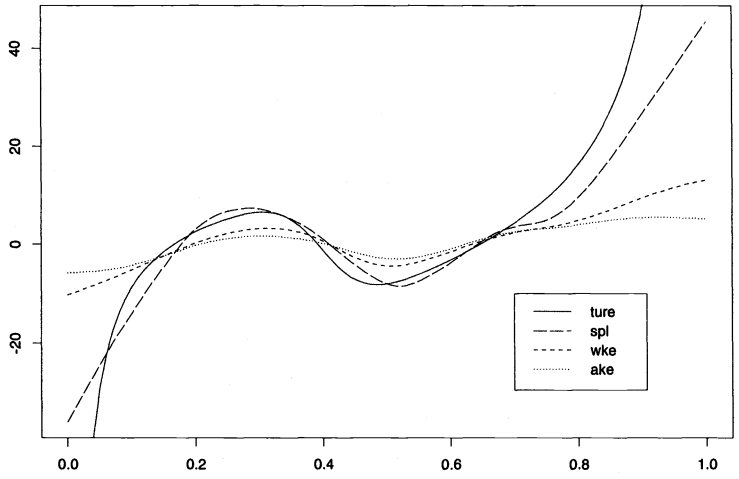


FIG. 10. ψ_0 *and* $\hat{\psi}_{IND}$ *of one beta mixture random sample of* 50 *observations.*

is somewhat more robust than some of the more ad hoc estimators, like the kernel estimator, to distribution variations. Thus when the underlying distribution of the error is unknown, which is the situation in most practical applications, an estimator that evolves as a solution to some well-defined objective functions, for example the penalized likelihood or the integrated squared error, may be a better nonparametric estimation technique.

5. Conclusions. In the spirit of nonparametric regression curve fitting, we have derived a "pseudo observation" interpretation for the smoothing spline score estimator of Cox [7] and illustrated an efficient way to compute this score estimator.

Various conventional data-driven smoothing parameter choice criteria based on the assumption of Gaussian innovation are extended to choosing the smoothing parameter of the smoothing spline score estimator but none of them yield a satisfactory result in the experiment carried out in §4. The adaptive information criteria introduced in §3 provide very encouraging results, especially when the Akaike penalty is used. Even though the number of replication is relatively small, our simulations in §4 seem to suggest that the smoothing spline score estimator using adaptive information criterion to choose the smoothing parameter performs much better than conventional kernel-based score estimators in finite sample situations for most distributions except the lognormal density even though they are all consistent asymptotically.

From the Monte-Carlo simulation results in §4, we have also learned that the accuracy of any version of the kernel score function estimators depends very much on the type of kernel being used. In practice, we have no a priori information on the unknown density function and hence have little guidance on deciding the type of kernel to use. In this sense, the smoothing spline score estimator is more robust to distributional variation.

Some versions of robustified smoothing splines as studied in Cox [6], Eubank [12], Koenker and Ng [24], and Utreras [41] may be better able to deal with the highly contaminated pseudo observations and improve the quality of the smoothing spline score estimator. This will be a topic for future research.

Appendix. Constructions of various score estimators.

A.1. Adaptive kernel estimator (Silverman [39]). Let

$$\hat{f}^{(v)}(x) = \sum_{i=1}^{n} p_i r_i^{v+1} K^{(v)}(r_i(x - X_i))$$

be the estimates of the density and its various derivatives, where $r_i = (h\lambda_i)^{-1}$ is the local bandwidth with local bandwidth factor λ_i determined by, $\lambda_i = (\tilde{f}(X_i)/g)^{-\alpha}$ in which $\tilde{f}$ is a pilot estimate of the density based on some fixed and scale dependent bandwidth, say h, α is the sensitivity parameter that controls the responsiveness of the local bandwidth to the pilot density, and $\log(g) = \sum_{i=1}^{n} p_i \log \tilde{f}(X_i)$. The adaptive kernel estimate of the ψ function is given by, $\hat{\psi}(x) = -(\hat{f}'(x))/(\hat{f}(x))$. The α used in the Monte-Carlo simulation is $\frac{1}{2}$. The kernel $K(\cdot)$ adopted is the Cauchy kernel, $K(x) = (\pi(1 + x^2))^{-1}$. The Cauchy kernel allows better estimates of the score functions in the tail segments of thick-tailed distributions. For the initial window width, h, we follow the rule, $h = \kappa \min(s_1, s_2)/n^{1/5}$, where s_1 and s_2 are the standard deviations and $\frac{1}{1.34}$(interquartile range) of the sample respectively, with $\kappa = 1.2$.

Readers are referred to Ng [29] for the rationale behind the choices of p_i, α, h, $K(\cdot)$, and κ.

A.2. The weighted kernel estimator (Cox and Martin [8]). Cox and Martin [8] suggested a broader class of kernel estimator of ψ that takes the following form:

$$\hat{\psi}(x) = \frac{\hat{g}(x)}{\hat{f}(x)},$$

where

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} h_0^{-1} K_0\left(h_0^{-1}(x - X_i)\right),$$

$$\hat{g}(x) = \frac{1}{n}\sum_{i=1}^{n} h_1^{-2} K_1\left(h_1^{-1}(x - X_i)\right),$$

and $K_1 = -H_1'$. The kernels K_0 and H_1 are both symmetric about 0 and satisfying some very mild regularity conditions.

On selecting the bandwidths for $\hat{f}$ and $\hat{g}$, they suggest minimizing one of the following weighted cross-validation scores.

$$CV_1 = -\sum_{i=1}^{n}(w(X_i)\log \hat{f}_{-i}(X_i))/\sum_{i=1}^{n} w(X_i) + \log\int \hat{f}(t)w(t)\,dt,$$

$$CV_2 = \int \hat{f}^2(t)w(t)\,dt - \frac{2}{n}\sum_{i=1}^{n} \hat{f}_{-i}(X_i)\,w(X_i),$$

$$CV_3 = \int \hat{g}^2(t)w(t)\,dt - \frac{2}{n}\sum_{i=1}^{n}[\hat{g}'_{-i}(X_i)\,w(X_i) + \hat{g}_{-i}(X_i)\,w'(X_i)],$$

where

$$w(x) = w_0\left(\frac{x - x_{(\alpha)}}{x_{(\beta)} - x_{(\alpha)}}\right).$$

In this simulation we use a logistic kernel with fixed window width chosen to be the largest of the window widths picked by CV_1, CV_2, and CV_3. The weight function takes the form,

$$w_0(q) = \begin{cases} 0 & \text{if } x < x_{(.05)} \text{ or } x > x_{(.95)}, \\ (3 - 2q)q^2 & \text{if } x_{(.05)} < x < x_{(.1)} \text{ or } x_{(.9)} < x < x_{(.95)}, \\ 1 & \text{if } x_{(.1)} < x < x_{(.9)} \end{cases}$$

with

$$q = \begin{cases} \frac{x - x_{(.05)}}{x_{(.1)} - x_{(.05)}} & \text{if } x_{(.05)} < x < x_{(.1)}, \\ \frac{x - x_{(.9)}}{x_{(.95)} - x_{(.9)}} & \text{if } x_{(.9)} < x < x_{(.95)}. \end{cases}$$

See Cox and Martin [8] and Ng [30] for the rationale behind these choices of kernel, w_0, and cross validation scores.

A.3. The smoothing spline estimator. The smoothing spline estimator for ψ used in the simulation is the $\hat{\psi}_{spl}(\lambda^*_{aki})$ introduced in §2.2 that uses the adaptive information criterion of §3.1 to pick the smoothing parameter λ.

Acknowledgments. This article evolves from the author's Ph.D. dissertation, under the supervision of Roger Koenker. The author would like to thank Dennis Cox for incisive criticism and for making some parts of his source codes available for use in the simulation in §4. Any errors that might have occurred during the adaptation is of course my responsibility. The author would like to express his gratitude to Dennis Cox and Douglas Martin for making an early draft of their paper available for reference. The author is also indebted to Anil Bera and Robin Sickles for helpful discussions. Finally, the comments and suggestions of the associate editor and a referee are gratefully acknowledged.

REFERENCES

[1] H. AKAIKE, *A new look at the statistical model identification*, IEEE Trans. Automat. Control, AC-19 (1974), pp. 716–723.

[2] R. A. BECKER, J. M. CHAMBERS, AND A. R. WILKS, *The New S Language: A Programming Environment for Data Analysis and Graphics*, Wadsworth and Brooks/Cole Advanced Book & Software, Pacific Grove, CA, 1984.

[3] P. J. BICKEL, *The* 1980 *Wald memorial lectures: On adaptive estimation*, Ann. Statist., 10 (1982), pp. 647–671.

[4] A. BUJA, T. HASTIE, AND R. TIBSHIRANI, *Linear smoothers and additive models*, Ann. Statist., 17 (1989), pp. 453–510.

[5] P. K. CLARK, *A subordinated stochastic process model with finite variance for speculative prices*, Econometrica, 41 (1973), pp. 135–155.

[6] D. D. COX, *Asymptotics for M-type smoothing splines*, Ann. Statist., 11 (1983), pp. 530–551.

[7] ———, *A penalty method for nonparametric estimation of the logarithmic derivative of a density function*, Ann. Instit. Statist. Math., 37 (1985), pp. 271–288.

[8] D. D. COX AND D. R. MARTIN, *Estimation of Score Functions*, Tech. Report, University of Washington, 1988.

[9] P. CRAVEN AND G. WAHBA, *Smoothing noisy data with spline functions, estimating the correct degree of smoothing by the method of generalized cross-validation*, Numer. Math., 31 (1979), pp. 377–408.

[10] M. CSÖRGŐ AND P. RÉVÉSZ, *An N.N.-estimator for the score function*, in Seminarberichte Nr.49, Proc. First Easter Conference on Model Theory, Sektion Mathematik, 1983, pp. 62–82.

[11] C. DEBOOR, *A Practical Guide to Splines*, in Applied Mathematical Sciences Vol. 27, Springer-Verlag, New York, 1978.

[12] R. EUBANK, *Spline Smoothing and Nonparametric Regression*, Dekker, New York, 1988.

[13] P.A. FOX, *The Port Mathematical Subroutine Library*, AT&T Bell Laboratories, Murray Hill, NJ, 1984.

[14] D.M. GAY, *Subroutines for unconstrained minimization using a model/trust-region approach*, ACM Trans. Math. Software, 9 (1983), pp. 503–524.

[15] E. J. HANNAN AND B. G. QUINN, *The determination of the order of an autoregression*, J. Roy. Statist. Soc., Ser. B, 41 (1978), pp. 190–195.

[16] W. HÄRDLE, W. HILDENBRAND, AND M. JERISON, *Empirical Evidence on the Law of Demand*, Projektbereich A, Discussion paper No. A-193, 1988.

[17] D. HSIEH AND C. MANSKI, *Monte Carlo evidence on adaptive maximum likelihood estimation of a regression*, Ann. Statist., 15 (1987), pp. 541–551.

[18] P. J. HUBER, *Robust Statistics*, Wiley, New York, 1981.

[19] IMSL, *Library Reference Manual*, IMSL Inc., Houston, TX, 1982.

[20] G. KIMELDORF AND G. WAHBA, *Some results on Tchebycheffian spline functions*, J. Math. Anal. Appl., 33 (1971), pp. 82–95.
[21] R. KOENKER, *Robust methods in econometrics*, Econometric Rev., 1 (1982), pp. 213–255.
[22] ———, *Discussion of 'The trimmed mean in the linear model' by A. H. Welsh*, Ann. Statist., 15 (1987), pp. 39–44.
[23] R. KOENKER AND V. D'OREY, *Computing regression quantiles*, Appl. Statist., 36 (1987), pp. 383–393.
[24] R. KOENKER AND P. NG, *Quantile smoothing splines*, Proc. International Symposium on Nonparametric Statistics and Related Topics, North-Holland, New York, 1992.
[25] M. J. LIGHTHILL, *Introduction to Fourier Analysis and Generalised Functions*, Cambridge University Press, New York, 1975.
[26] J. MACHADO, *Model Selection: Consistency and Robustness Properties of the Schwarz Information Criterion for Generalized M-Estimation*, Ph.D. dissertation, Dept. Economics, University of Illinois, Champaign-Urbana, 1989.
[27] C. MANSKI, *Adaptive estimation of non-linear regression models*, Econometric Rev., 3 (1984), pp. 145–194.
[28] D.R. MARTIN, *Robust estimation of autoregressive models*, in Directions in Time Series, 228–262, D. R. Brillinger and G. C. Tiao, eds., Institute of Mathematical Statistics, Hayward, CA, 1980.
[29] P. NG, *Estimation of the score function with application to adaptive estimators*, working paper, University of Houston, 1989.
[30] ———, *Computing smoothing spline score estimator*, working paper, University of Houston, 1989.
[31] F. O'SULLIVAN, *Fast computation of fully automated log-density and log-hazard estimators*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 363–379.
[32] S. PORTNOY AND R. KOENKER, *Adaptive L-estimation of linear models*, Ann. Statist., 17 (1989), pp. 362–381.
[33] C. REINSCH, *Smoothing by spline functions*, Numer. Math., 10 (1967), pp. 177–183.
[34] E. RONCHETTI, *Robust model selection in regression*, Statist. Probab. Lett., 3 (1985), pp. 21–23.
[35] G. SCHWARZ, *Estimating the dimension of a model*, Ann. Statist., 6 (1978), pp. 461–464.
[36] R. SERFLING, *Approximation Theorems of Mathematical Statistics*, Wiley, New York, 1980.
[37] R. SHIBATA, *An optimal selection of regression variables*, Biometrika, 68 (1981), pp. 45–54.
[38] B. W. SILVERMAN, *On the estimation of a probability density function by the maximum penalized likelihood method*, Ann. Statist., 10 (1982), pp. 795–810.
[39] ———, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, London, New York, 1986.
[40] C. STONE, *Adaptive maximum likelihood estimators of a location parameter*, Ann. Statist., 3 (1975), pp. 267–284.
[41] F. I. UTRERAS, *On computing robust splines and applications*, SIAM, J. Sci. Statist. Comput., 2 (1981), pp. 153–163.
[42] G. WAHBA, *Improper priors, spline smoothing and the problem of guarding against model errors in regression*, J. Roy. Statist. Soc., Ser. B, 40 (1978), pp. 364–372.
[43] ———, *Spline Models for Observational Data*, CBMS 59, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1026–1037, September 1994

002

ALGEBRAIC MULTILEVEL PRECONDITIONING OF ANISOTROPIC ELLIPTIC PROBLEMS*

SVETOZAR D. MARGENOV† AND PANAYOT S. VASSILEVSKI‡

Abstract. In this paper the recently proposed algebraic multilevel iteration method for iterative solution of elliptic boundary value problems with anisotropy and discontinuous coefficients is studied. Based on a special approximation of the blocks corresponding to the new nodes at every discretization level, an optimal order preconditioner with respect to the arithmetic cost independent of both the discontinuity and the anisotropy of the coefficients is constructed. The advantages of the proposed algorithms are illustrated by numerical tests.

Key words. multilevel preconditioner, elliptic problems, anisotropy, finite elements, optimal order preconditioner

AMS subject classifications. 65F10, 65N20, 65N30

Introduction. The recently proposed algebraic multilevel iteration (AMLI) methods (Axelsson and Vassilevski [4], [5], Vassilevski [16], see also [13], [2], [15]) are some of the most efficient techniques for the numerical solution of second order elliptic boundary value problems. An important feature of these methods is that they converge with an optimal rate independently of the regularity of the elliptic problem. It was proved in Axelsson and Vassilevski [4] and [5] (see also Vassilevski [16]) that the AMLI methods are optimal with respect to both convergence rate and arithmetic operations per iteration for problems with discontinuous coefficients. In the present paper we consider a more complicated case when the problem can be also anisotropic. There are some recent papers (Axelsson [1], Axelsson and Eijkhout [2], Donato and Chan [7], and Hackbusch [12]) dealing with anisotropic problems, but the algorithms presented there depend in general on the ratio of the anisotropy. The behavior of the related constants (denoted commonly by γ) in the strengthened CBS (Cauchy–Bunyakowski–Schwarz) inequality was numerically studied in Vassilevski and Etova [17] for a wide range of piecewise polynomial finite element spaces.

In this paper we present an optimal order multilevel iterative method for second order elliptic boundary value problems with strong anisotropy and discontinuous coefficients. The proposed preconditioner is a special case of the AMLI methods from [5] with approximate blocks $A_{11}^{(k)}$. The AMLI method is based on polynomial inner iterations (inner between the levels of discretization) and it is a straightforward multilevel algebraic generalization of the two-by-two block factorization methods used in Bank and Dupont [6] and Axelsson and Gustafsson [3]. The blocks $A_{11}^{(k)}$ of the matrix at discretization level k correspond to the new nodes added at refinement level k. The purpose of this paper is to construct spectrally equivalent approximations of $A_{11}^{(k)}$ that are also insensitive with respect to possible anisotropy of the coefficients of the bilinear form. The construction we consider is based on special ordering of the unknowns (related to the block $A_{11}^{(k)}$) in the direction of strong anisotropy. Based on such an ordering we break the connections in the corresponding finite element mesh and the approximation is then defined by assembling the element matrices using only the remaining connections in the mesh. This gives rise to a block structure of the thus constructed approximation $B_{11}^{(k)}$ to $A_{11}^{(k)}$ that is then easy to factor. Strategies of ordering unknowns in

*Received by the editors June 24, 1992; accepted for publication (in revised form) June 8, 1993. This research was conducted while both authors were on leave from the Center of Informatics and Computer Technology, "Acad. G. Bontchev" str., Bl. 25 A, 1113 Sofia, Bulgaria. This research was supported in part by National Science Foundation grant INT-8914472 and by Bulgarian Ministry of Education and Science grant MM75-91.

†Center of Informatics and Computer Technology, "Acad. G. Bontchev" str., Bl.25A, 1113 Sofia, Bulgaria (margenov@bgearn.bitnet).

‡Department of Mathematics, University of California at Los Angeles, 405 Hilgard Avenue, Los Angeles, California 90024-1555 (panayot@math.ucla.edu).

direction of strong anisotropy have been explored in Duff and Meurant [8] and Eijkhout [9] in regard to their influence on the convergence rate of incomplete factorization preconditioners. Our approach is based on local, i.e., element-by-element construction (the elements are from the finite element triangulation of the domain at level $k-1$) and, more importantly, our construction allows us to derive estimates for the spectral equivalence relations between $A_{11}^{(k)}$ and $B_{11}^{(k)}$ and this we view as the main contribution of the present paper. We also show, by numerical experiments, that in practice one can also use the well-known (block-)ILU approximations of the blocks $A_{11}^{(k)}$ in combination with the adaptive hybrid multilevel iteration method from Vassilevski [16]. However, this can only be seen as one more demonstration of the robustness of the (block-)ILU type preconditioners.

The formulation of the considered problem is given in §1. In §2 the main results about the AMLI methods from Axelsson and Vassilevski [4], [5] are reviewed and in §3 the optimal bounds for the relative condition numbers of the multilevel preconditioner with the proposed approximate blocks $B_{11}^{(k)}$ of $A_{11}^{(k)}$ with respect to the corresponding stiffness matrix are proved. These results are shown under the assumption that the coefficients of the second order elliptic operator (of divergence type) are constants on the triangles from the initial (coarsest) triangulation. Various numerical tests illustrating the main theoretical results obtained in the present paper are given in §4. At the end some additional numerical results are presented when approximations based on block-ILU factorization of $A_{11}^{(k)}$ are used, showing the robustness of the ILU-like approximations in the multilevel methods as well.

1. The problem. Consider the elliptic equation

$$
\begin{aligned}
-(a(x,y)u_x)_x - (b(x,y)u_y)_y &= f(x,y), \quad \text{for } (x,y) \in \Omega, \\
u &= 0 \quad \text{on } \Gamma_D, \\
au_x n_1 + bu_y n_2 &= 0 \quad \text{on } \Gamma_N,
\end{aligned} \tag{1.1}
$$

where the domain Ω is partitioned using triangles $T \in \mathcal{T}_1$ defined on an intentionally coarse mesh ω_1; $n = (n_1, n_2)$ is the outward unit vector normal to the boundary Γ_N. $(.)_x$, $(.)_y$ denote partial derivatives.

We assume that $a(x, y)$ and $b(x, y)$ are constants in each element $T_i \in \mathcal{T}_1$. The coefficients $a_i = a|_{T_i}$ and $b_i = b|_{T_i}$ may be discontinuous across the elements.

The corresponding Galerkin variational formulation of (1.1) follows.

Given $f \in L^2(\Omega)$ find a function $u \in H_0^1(\Omega) = \{v \in H^1(\Omega) : \ v = 0 \text{ on } \Gamma_D\}$, satisfying

$$
a(u, v) = (f, v) \quad \text{for all } v \in H_0^1(\Omega), \tag{1.2}
$$

where

$$
a(u, v) = \int_\Omega \left(au_x v_x + bu_y v_y\right) dx \ dy.
$$

For the approximate solution of (1.2), the standard finite element method is used. Let $V_1 \subset H_0^1(\Omega)$ denote the set of piecewise linear basis functions corresponding to the triangulation $\mathcal{T}_1$. The standard computational procedure leads to the linear system of equations

$$
A^{(1)}\mathbf{u}^{(1)} = \mathbf{f}^{(1)}, \tag{1.3}
$$

where $A^{(1)}$ is the corresponding stiffness matrix.

Now in order to obtain a sufficiently accurate solution of problem (1.2) a uniform refinement procedure is used to construct a sequence of meshes $\omega_1 \subset \omega_2 \subset \ldots \subset \omega_\ell$ corresponding

to the triangulations $\mathcal{T}_1 \subset \mathcal{T}_2 \subset \ldots \subset \mathcal{T}_\ell$. Associated with the triangulations $\{\mathcal{T}_k\}$ are the finite element spaces $V_1 \subset V_2 \subset \ldots \subset V_\ell$ and the corresponding finite element stiffness matrices computed using standard nodal basis test functions; i.e., we have $A^{(1)}, A^{(2)}, \ldots, A^{(\ell)}$.

The solution of the system of equations corresponding to the finest triangulation $\mathcal{T}_\ell$ (which is the problem we want to solve, since it best approximates the solution of the differential problem)

$$A\mathbf{u} = \mathbf{f}, \tag{1.4}$$

$A = A^{(\ell)}$, $\mathbf{u} = \mathbf{u}^{(\ell)}$, and $\mathbf{f} = \mathbf{f}^{(\ell)}$, is computed by the preconditioned conjugate gradient (PCG) method. If M is a preconditioning matrix, the convergence properties of the PCG method are given by the estimate

$$\mathbf{r}^{(i)^T} A^{-1}\mathbf{r}^{(i)} \leq \left(\frac{2q^i}{1+q^{2i}}\right)^2 \mathbf{r}^{(0)^T} A^{-1}\mathbf{r}^{(0)}, \tag{1.5}$$

where

$$q = \frac{\varkappa^{\frac{1}{2}} - 1}{\varkappa^{\frac{1}{2}} + 1}, \quad \text{and} \quad \varkappa = \operatorname{Cond}\left(M^{-1}A\right).$$

Here $\mathbf{r}^{(i)} = \mathbf{f} - A\mathbf{x}^{(i)}$ is the ith residual. Estimate (1.5) shows that the number of iterations needed to reduce the A^{-1}-norm of the initial residual by a factor ϵ is $O\left(\varkappa^{1/2} \log \frac{1}{\epsilon}\right)$.The main goal of this paper is to construct a matrix M such that the action of M^{-1} on any given vector $\mathbf{v}$ can be computed in an amount of arithmetic operations proportional to the number of the unknowns. We also require that the condition number $\varkappa$ is bounded uniformly with respect to the number of degrees of freedom and to the material coefficients as well, i.e., that $\varkappa$ remains bounded when $\frac{a}{b} \to 0$ or when $\frac{b}{a} \to 0$. Our construction also allows certain discontinuity of the coefficients.

2. Algebraic multilevel preconditioning methods. First we consider a two-level preconditioning algorithm for solving the system

$$A^{(k+1)}\mathbf{u}^{(k+1)} = \mathbf{f}^{(k+1)}. \tag{2.1}$$

To define the preconditioning matrix $C^{(k+1)}$, we partition the nodes $\mathcal{N}^{(k+1)}$ of $\mathcal{T}_{k+1}$ into two subsets: $\mathcal{N}^{(k+1)} \setminus \mathcal{N}^{(k)}$ and $\mathcal{N}^{(k)}$. Corresponding to this partitioning, $A^{(k+1)}$ takes the following two-by-two block structure

$$A^{(k+1)} = \begin{pmatrix} A_{11}^{(k+1)} & A_{12}^{(k+1)} \\ A_{21}^{(k+1)} & A_{22}^{(k+1)} \end{pmatrix}, \tag{2.2}$$

where the first pivot block $A_{11}^{(k+1)}$ corresponds to the nodes of $\mathcal{N}^{(k+1)} \setminus \mathcal{N}^{(k)}$ and the second diagonal block $A_{22}^{(k+1)}$ corresponds to the nodes of $\mathcal{N}^{(k)}$. Following the construction proposed in Bank and Dupont [6] and Axelsson and Gustafsson [3], we define the two-level preconditioner by the relation

$$C^{(k+1)} = \begin{pmatrix} A_{11}^{(k+1)} & 0 \\ A_{21}^{(k+1)} & A^{(k)} \end{pmatrix} \begin{pmatrix} I & A_{11}^{(k+1)^{-1}} A_{12}^{(k+1)} \\ 0 & I \end{pmatrix}. \tag{2.3}$$

The following theorem is well known (Bank and Dupont [6], Axelsson and Gustafsson [3]).

THEOREM 2.1. *The matrices $A^{(k+1)}$ and $C^{(k+1)}$ are spectrally equivalent and the following spectral equivalence relations hold*

$$(1-\gamma^2)\,\xi^T C^{(k+1)}\xi \le \xi^T A^{(k+1)}\xi \le \xi^T C^{(k+1)}\xi \quad \text{for all } \xi, \tag{2.4}$$

where γ is the constant in the strengthened CBS *inequality (e.g.,* [6], [3], *and* [10]).

We now define the algebraic multilevel preconditioner $M = M^{(\ell)}$ for the system (1.4) (see Axelsson and Vassilevski [4], [5]) by the following recurrence:

$$M^{(1)} = A^{(1)} \quad \text{for } k = 1, 2, \ldots, \ell - 1,$$

$$M^{(k+1)} = \begin{pmatrix} A_{11}^{(k+1)} & 0 \\ A_{21}^{(k+1)} & \tilde{A}^{(k)} \end{pmatrix} \begin{pmatrix} I & A_{11}^{(k+1)^{-1}} A_{12}^{(k+1)} \\ 0 & I \end{pmatrix}, \tag{2.5}$$

where

$$\tilde{A}^{(k)^{-1}} = \left[I - p_\beta \left(M^{(k)^{-1}} A^{(k)} \right) \right] A^{(k)^{-1}}.$$

Here $p_\beta(t)$ is a polynomial of degree β such that $p_\beta(0) = 1$ and $0 \le p_\beta(t) < 1$ in $(0, 1]$. We use

$$p_\beta(t) = \frac{1 + T_\beta\left(\frac{1+\alpha-2t}{1-\alpha}\right)}{1 + T_\beta\left(\frac{1+\alpha}{1-\alpha}\right)},$$

where $\alpha \in (0, 1)$ is properly chosen and T_β is the Chebyshev polynomial of degree β.

The basic result of the theory of the algebraic multilevel preconditioning methods (cf. Axelsson and Vassilevski [4], [5]) is given in the next theorem.

THEOREM 2.2. *There exists an $\alpha \in (0, 1)$ if $\beta > (1-\gamma^2)^{-1/2}$ (the degree of the scaled and shifted Chebyshev polynomial) such that the preconditioning matrix M and the stiffness matrix A are spectrally equivalent; i.e., the relative condition number $\varkappa\left(M^{-1}A\right)$ is bounded by a constant independent of the number of the unknowns N. More precisely, we have*

$$\varkappa\left(M^{-1}A\right) \le \frac{1}{\alpha}. \tag{2.6}$$

It follows directly from the structure of the preconditioning matrix M and from the last theorem that if $\gamma^2 < \frac{3}{4}$ and $\beta = 2$ or 3, in the case of uniform refinement by subdividing every triangle into four congruent ones, then the resulting algebraic multilevel method has an optimal arithmetic cost of $O(N \log \frac{1}{\epsilon})$ operations where $\epsilon > 0$ is the desired accuracy in the conjugate gradient method. However, the constant in $O(N \log \frac{1}{\epsilon})$ depends on the condition number of the blocks $\{A_{11}^{(k+1)}\}$. The realization of the PCG method with the above defined preconditioner M needs solutions of systems with the blocks $A_{11}^{(k+1)}$, $k = 1, \ldots, \ell - 1$ at every iteration. These matrices are symmetric and positive definite with condition number $\varkappa(A_{11}^{(k+1)})$ bounded independently of N_k, the number of nodes at level k. However, $\varkappa(A_{11}^{(k+1)})$ tends to infinity with the ratio of anisotropy ($b(x)/a(x) \to 0$ or $a(x)/b(x) \to 0$). To overcome the difficulties arising in problems with strong anisotropy we use a modified multilevel preconditioned algorithm approximating the blocks $A_{11}^{(k+1)}$ (see Axelsson and Vassilevski [5]). Let $B_{11}^{(k+1)}$ be a properly chosen symmetric and positive definite approximation to $A_{11}^{(k+1)}$ that satisfies

$$b = \text{ const } \ge \varkappa\left(B_{11}^{(k+1)^{-1}} A_{11}^{(k+1)}\right) - 1$$

for some positive constant b independent of the level number k.

The main result from [5] says that for $\beta > (1-\gamma^2)^{-1/2}$ the relative condition number $\varkappa$ of the corresponding multilevel preconditioning matrix M (with approximate blocks $\{B_{11}^{(k+1)}\}$) with respect to A is bounded uniformly by $\frac{1}{\alpha}$, where for $\beta = 2$,

$$\alpha = \frac{4\mu - 1}{1 + 2b + \left(4\mu - 1 + (1+2b)^2\right)^{\frac{1}{2}}}, \qquad \mu = 1 - \gamma^2,$$

and for $\beta = 3$, $\alpha \in (0, 1)$ is the smallest positive root of the cubic equation

$$bt^3 + (6b + 9 - \mu)t^2 + (9b + 6 - 6\mu)t + 1 - 9\mu = 0, \qquad \mu = 1 - \gamma^2.$$

Such an α exists if $\mu > \frac{1}{9}$.

Remark 2.1. The theory of the strengthened CBS inequality (e.g., Bank and Dupont [6], Maitre and Musy [14], see also Eijkhout and Vassilevski [10]) says that the constant γ can be estimated by a local analysis with respect to the elements from the initial (coarsest) triangulation. Applying this approach one can obtain the following estimates uniformly with respect to the ratio of anisotropy:

$\gamma^2 < \frac{3}{4}$ uniformly with respect to the shape of the triangles of the initial triangulation $\mathcal{T}_1$;

$\gamma^2 \leq \frac{1}{2}$ if the initial triangulation $\mathcal{T}_1$ consists of right triangles with legs parallel to the coordinate axes.

In §3 we propose a multilevel preconditioner with approximate blocks that is of optimal order independent of the anisotropy of the problem.

3. Optimal order approximations of $A_{11}^{(k+1)}$. It is well known, e.g., Bank and Dupont [6], Axelsson and Gustafsson [3] that the block $A_{11}^{(k+1)}$ in the multilevel decomposition of the stiffness matrix $A^{(k+1)}$ is positive definite and the condition number $\varkappa(A_{11}^{(k+1)})$ of $A_{11}^{(k+1)}$ is independent of the size of the problem at the $(k+1)$th discretization level. However, $\varkappa(A_{11}^{(k+1)})$ tends to infinity with $\frac{b}{a} \to 0$ and with $\frac{a}{b} \to 0$. In this section we consider an optimal order method with respect to both the number of discretization levels and the anisotropy based on a specific approximate block factorization of $A_{11}^{(k+1)}$.

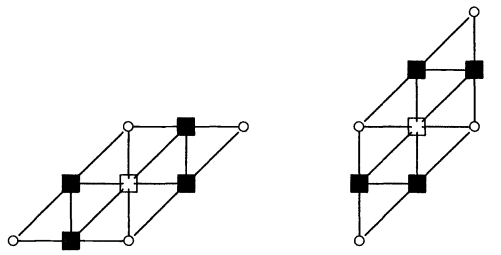
In what follows we assume that the initial triangulation $\mathcal{T}_1$ consists of right isosceles triangles with legs parallel to the coordinate axes. Recall that we have also assumed that the coefficients $a(x, y)$ and $b(x, y)$ are piecewise constants with respect to the elements of the initial triangulation $\mathcal{T}_1$. For notational convenience from now on we omit the superscripts whenever appropriate, i.e., we write $A_{11} = A_{11}^{(k+1)}$, etc.

In order to define the approximation to A_{11} we partition the nodes of $\mathcal{N}^{(k+1)} \setminus \mathcal{N}^{(k)}$ into two groups, marked, respectively, by "□" and "■" (see Fig. 1; the nodes from $\mathcal{N}^{(k)}$ are marked by "○"). The first new group of nodes consists of those belonging to either of the following two subsets:

- the centers of parallelograms Q_y consisting of two neighboring coarse-grid triangles (i.e., triangles from $\mathcal{T}_k$), which have a common cathetus oriented along the y-axis if $\frac{b}{a} \geq 1$ in Q_y (see Fig. 1(a));
- the centers of parallelograms Q_x consisting of two neighboring coarse-grid triangles (i.e., triangles from $\mathcal{T}_k$), which now have a common cathetus oriented along the x-axis if $\frac{b}{a} \leq 1$ in Q_x (see Fig. 1(b)).

The second group contains the remaining nodes from $\mathcal{N}^{(k+1)} \setminus \mathcal{N}^{(k)}$.

With respect to this partitioning of the nodes of $\mathcal{N}^{(k+1)} \setminus N^{(k)}$, A_{11} admits the following



(a) $Q_y : \frac{b}{a} \geq 1$ *in* Q_y. (b) $Q_x : \frac{a}{b} \geq 1$ *in* Q_x.

FIG. 1. *Block partitioning of the nodes in* $\mathcal{N}^{(k+1)} \setminus \mathcal{N}^{(k)}$.

two-by-two block-factored form

$$A_{11} = \begin{pmatrix} D & F \\ F^T & E \end{pmatrix} = \begin{pmatrix} D & 0 \\ F^T & S \end{pmatrix} \begin{pmatrix} I & D^{-1}F \\ 0 & I \end{pmatrix}, \tag{3.1}$$

where $S = E - F^T D^{-1} F$. Noting that D is a diagonal matrix it follows then that the Schur complement S can be assembled from the corresponding macroelement Schur complements S_Q (see Fig. 1) (corresponding to the macroelement stiffness matrices $A_{11;Q}$) obtained when the center nodes of every parallelogram Q are eliminated. This procedure is sometimes called "static condensation."

Let $\{Q\}$ be a partitioning of Ω consisting of parallelograms Q_x (when $a \geq b$ in Q_x) and Q_y (when $b \geq a$ in Q_y) with vertices from $\mathcal{N}^{(k)}$. This partitioning is determined by the initial triangulation $\mathcal{T}_1$. (Note that we have assumed that a and b are piecewise constants with respect to the elements from $\mathcal{T}_1$.) Then we can symbolically write

$$S = \bigoplus_{\{Q\}} S_Q, \tag{3.2}$$

where "$\oplus$" stands for assembling with respect to the partitioning $\{Q\}$ of Ω.

We consider the following approximate factorization of A_{11},

$$B_{11} = \begin{pmatrix} D & 0 \\ F^T & E \end{pmatrix} \begin{pmatrix} I & D^{-1}F \\ 0 & I \end{pmatrix}, \tag{3.3}$$

i.e., the symmetric block Gauss–Seidel preconditioner of A_{11}. Here the Schur complement S is simply replaced (approximated) by the block-matrix E. We recall again that in our particular case of refinement the block D is scalar diagonal. The solution of systems with the block-matrix E will be discussed in a moment.

We now study the spectral equivalence relations between A_{11} and B_{11}, which is equivalent to studying the spectral equivalence relations between the blocks E and S.

THEOREM 3.1. *The relative condition number of the approximate blocks* B_{11} *defined by* (3.3) *with respect to* A_{11} *can be estimated as follows*:

$$\varkappa\left(B_{11}^{-1} A_{11}\right) \leq 2. \tag{3.4}$$

Proof. Following the definition of the approximation B_{11} we have that a local analysis with respect to the parallelograms Q of the relative condition number $\varkappa\left(B_{11}^{-1} A_{11}\right)$ can be applied, i.e.,

$$\varkappa(B_{11}^{-1} A_{11}) \leq \max_Q \varkappa(B_{11;Q}^{-1} A_{11;Q}) = \max_Q \varkappa(E_Q^{-1} S_Q). \tag{3.5}$$

Let us consider the case when $Q = Q_y$; namely, assume that $\frac{b}{a} \geq 1$ in Q_y (see Fig. 1(a)). Denote by $\{(a_i, b_i),\ i = 1, 2\}$ the values of the material constants related to the triangles T_1, T_2 from $\mathcal{T}_k$, where $\bar{Q} = \bar{T}_1 \cup \bar{T}_2$.

The matrices S_Q and E_Q can be explicitly computed; namely, we have

$$S_Q = 2\begin{bmatrix} a_1 + b_1 - a_1^2/R & -a_1a_2/R & -b_1 & 0 \\ -a_1a_2/R & a_2 + b_2 - a_2^2/R & 0 & -b_2 \\ -b_1 & 0 & a_1 + b_1 & 0 \\ 0 & -b_2 & 0 & a_2 + b_2 \end{bmatrix},$$

where $R = a_1 + a_2 + b_1 + b_2$ and

$$E_Q = 2\begin{bmatrix} a_1 + b_1 & 0 & -b_1 & 0 \\ 0 & a_2 + b_2 & 0 & -b_2 \\ -b_1 & 0 & a_1 + b_1 & 0 \\ 0 & -b_1 & 0 & a_2 + b_2 \end{bmatrix}.$$

Since $S_Q = E_Q - F_Q^T D_Q^{-1} F_Q$ we have

$$\mathbf{y}^T E_Q \mathbf{y} \geq \mathbf{y}^T S_Q \mathbf{y}. \tag{3.6}$$

Consider now the matrix $\Phi_Q \equiv S_Q - \frac{1}{2}E_Q$,

$$\Phi_Q = \begin{bmatrix} a_1 + b_1 - 2a_1^2/R & -2a_1a_2/R & -b_1 & 0 \\ -2a_1a_2/R & a_2 + b_2 - 2a_2^2/R & 0 & -b_2 \\ -b_1 & 0 & a_1 + b_1 & 0 \\ 0 & -b_2 & 0 & a_2 + b_2 \end{bmatrix}.$$

The matrix Φ_Q has nonpositive off-diagonal entries. Moreover, since $a_i \leq b_i$, $i = 1, 2$, it follows that Φ_Q is diagonally dominant; namely,

$$a_1 + b_1 - 2a_1^2/R - 2a_1a_2/R - b_1 = a_1\left[(b_1 - a_1) + (b_2 - a_2)\right]/R \geq 0,$$
$$a_2 + b_2 - 2a_2^2/R - 2a_1a_2/R - b_2 = a_2\left[(b_1 - a_1) + (b_2 - a_2)\right]/R \geq 0.$$

This implies that Φ_Q is positive definite, hence

$$\mathbf{y}^T S_Q \mathbf{y} \geq \frac{1}{2}\mathbf{y}^T E_Q \mathbf{y} \quad \text{for all } \mathbf{y} \in \mathbb{R}^4. \tag{3.7}$$

The case $Q = Q_y$ is considered precisely in the same way as above.

Combining (3.6) and (3.7) we obtain (3.4) which completes the proof. □

Remark 3.1. If $\frac{a}{b} = \epsilon \leq 1$ is a constant in the considered parallelogram Q, then the following more precise estimate than (3.7) holds:

$$\mathbf{y}^T S_Q \mathbf{y} \geq \left(1 - \frac{\epsilon}{2 + \epsilon}\right)\mathbf{y}^T E_Q \mathbf{y} \quad \text{for all } \mathbf{y} \in \mathbb{R}^4. \tag{3.8}$$

We now consider three examples to explain the structure of the block-matrix E in the factorization (3.3) of B_{11}. The model domain is $\Omega = (0, 1) \times (0, 1)$. The connections between the nodes related to B_{11} are marked by a double line.

Example 1. The anisotropy is y-oriented, i.e., $b \geq a$ in whole Ω.

Example 2. $b \geq a$ in $\{(x, y) : \ x + y \geq 1\}$ and $b \leq a$ in $\{(x, y) : \ x + y \leq 1\}$.

Example 3. $b \geq a$ in $\left\{(x, y) : \ y \geq \frac{1}{2}\right\}$ and $b \leq a$ in $\left\{(x, y) : \ y \leq \frac{1}{2}\right\}$.

In Examples 1 and 2 (after a suitable ordering), E admits a block-diagonal form with (scalar) tridiagonal blocks (see Fig. 2(a), (b)). In Fig. 2(c), Example 3, E has a more complicated structure, but one can see that solving systems with E is again based on tridiagonal solvers. Indeed, one can factor E in the subdomain $\{(x, y), 1 \geq y \geq \frac{1}{2}\}$ starting from the top (i.e., $y = 1$) towards the line of change of anisotropy $\{y = \frac{1}{2}\}$. This does not create any fill-in outside the line $\{y = \frac{1}{2}\}$. Then the reduced matrix admits a block-diagonal form with (scalar) tridiagonal blocks similarly to Example 1 but now in the subdomain $\{(x, y) : 0 \leq y \leq \frac{1}{2}\}$.

Remark 3.2. The demonstrated strategy of ordering the nodes related to the block matrix E can be summarized as "preserving the links between the mesh nodes along the dominated anisotropy." This strategy leads in a more general case to a sparse structure of E such that solving systems with E requires: solving a number of tridiagonal systems; solving one band system with a bandwidth of order equal at most to that of $A^{(1)}$, i.e., of order $O(N_1)$. We recall that N_1 is the size of the problem on the initial coarsest mesh and problems of such relatively small size can be solved directly by LU factorization.

The above considerations show that solving systems with the matrices B_{11} require a number of arithmetic operations proportional to the dimension of A_{11}. This fact and the main result from Axelsson and Vassilevski [5] reviewed in §2 imply the following theorem.

THEOREM 3.2. *The relative condition number* $\varkappa\left(M^{-1}A\right)$ *corresponding to the modified multilevel method with approximate blocks* $\{B_{11}^{(k+1)}, \ k = 1, 2, \ldots, \ell - 1\}$ *defined by* (3.3) *is bounded by a constant independent of both* N *and the problem coefficients if the polynomial degree* $\beta \in \{2, 3\}$. *The resulting* PCG *method has a total arithmetic cost proportional to* $N \log \frac{1}{\epsilon}$, *where* ϵ *is the desired accuracy and* N *is the problem size, i.e., the method is of optimal order.*

4. Numerical tests. We consider the test problem defined by (1.1) in $\Omega = (0, 1) \times (0, 1)$ with Dirichlet boundary conditions on the boundaries $\{x = 0\}$ and $\{y = 0\}$ and Neumann boundary conditions on the rest of $\partial\Omega$. The problem is discretized by the finite element method with piecewise linear functions on isosceles right triangulations $\mathcal{T}_k$, $k = 0, 1, 2, \ldots, \ell$, for $\ell = 3, 4, \ldots, 7$. The meshsize at level k was $h_k = 2^{-k}$.

The first group of numerical tests illustrate the behavior of the relative condition number of the approximation $B_{11}^{(k)}$ to the blocks $A_{11}^{(k)}$.

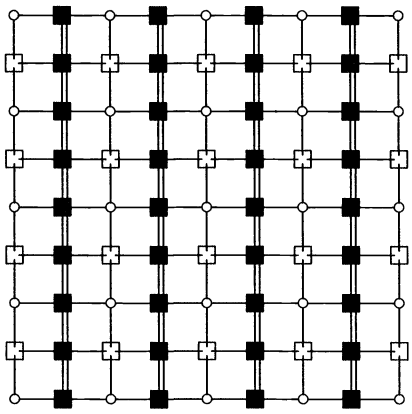
The block-matrix $B_{11}^{(k)}$ was defined by the rule corresponding to the case of y-oriented anisotropy (see Example 1, Fig. 2(a)).

The first set of experiments corresponded to constant coefficients a and b, shown in Tables 1–2. The values of a and b are denoted by (a, b).

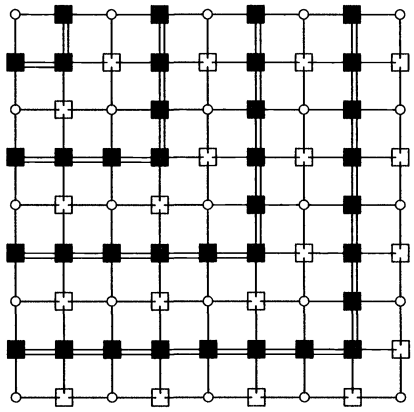
TABLE 1
$\kappa = \text{Cond}(B_{11}^{(k)^{-1}} A_{11}^{(k)})$; *smooth coefficients.*

Levels	$a = 1, b = 1$	$a = 1, b = 10$	$a = 1, b = 100$
3	1.460	1.040	1.0016
4	1.489	1.047	1.0033
5	1.496	1.049	1.0044
6	1.498	1.049	1.0048
7	1.499	1.049	1.0049

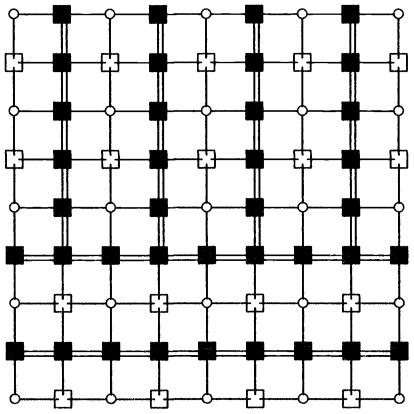
In the next set of experiments, we split the domain Ω into four subdomains Ω_1, Ω_2, Ω_3, and Ω_4 obtained from Ω using the separator lines $\{x = \frac{1}{2}\}$ and $\{y = \frac{1}{2}\}$ (see Fig. 3). We denote



(a) *Example* 1. $\left(\frac{b}{a} \geq 1\right)$ *in* Ω.



(b) *Example* 2 ($\frac{b}{a} \geq 1$ *in* $\Omega \cap \{x + y \geq 1\}$) *and* ($\frac{b}{a} \leq 1$ *in* $\Omega \cap \{x + y \leq 1\}$).



(c) *Example* 3. ($\frac{b}{a} \geq 1$ *in* $\Omega \cap \{y \geq \frac{1}{2}\}$) *and* ($\frac{b}{a} \leq 1$ *in* $\Omega \cap \{y \leq \frac{1}{2}\}$).

FIG. 2. *Substructuring of* Ω *related to the connections in* B_{11}.

in Tables 3–5 by $a_i = a|_{\Omega_i}$ and $b_i = b|_{\Omega_i}$, $i = 1, 2, 3, 4$. In this case we have, in addition to the discontinuity of the coefficients (Tables 3 and 4), mixed anisotropy shown in Table 5. In the latter case the coefficients a and b were kept constants in the subdomain $\Omega_2 \cup \Omega_4$ equal to 1, and in the rest of Ω their values (a, b) were varied.

TABLE 2
$\kappa = \mathrm{Cond}(B_{11}^{(k)^{-1}} A_{11}^{(k)})$; *smooth coefficients.*

Levels	$a = 1, b = 1$	$a = 10, b = 1$	$a = 100, b = 1$	$a = 1000, b = 1$
3	1.460	5.030	17.56	25.01
4	1.489	5.731	34.43	86.21
5	1.496	5.915	44.29	142.85
6	1.498	5.966	47.80	183.60
7	1.499	5.984	49.46	279.32

TABLE 3
$\kappa = \mathrm{Cond}\,(B_{11}^{(k)^{-1}} A_{11}^{(k)})$; *discontinuous coefficients:* $a_2 = a_4 = b_2 = b_4 = 1$.

Levels	$a_1 = a_3 = 1$ $b_1 = b_3 = 1$	$a_1 = a_3 = 1$ $b_1 = b_3 = 10$	$a_1 = a_3 = 1$ $b_1 = b_3 = 100$	$a_1 = a_3 = 1$ $b_1 = b_3 = 1000$
3	1.460	1.341	1.330	1.329
4	1.489	1.444	1.441	1.440
5	1.496	1.479	1.479	1.479
6	1.498	1.493	1.493	1.493
7	1.499	1.497	1.497	1.497

TABLE 4
$\kappa = \mathrm{Cond}\,(B_{11}^{(k)^{-1}} A_{11}^{(k)})$; *discontinuous coefficients:* $a_2 = a_4 = b_2 = b_4 = 1$.

Levels	$a_1 = a_3 = 1$ $b_1 = b_3 = 1$	$a_1 = a_3 = 10$ $b_1 = b_3 = 1$	$a_1 = a_3 = 100$ $b_1 = b_3 = 1$	$a_1 = a_3 = 1000$ $b_1 = b_3 = 1$
3	1.460	5.226	42.92	414.78
4	1.489	5.640	46.20	220.28
5	1.496	5.812	42.20	105.68
6	1.498	5.890	46.59	219.87
7	1.499	5.955	48.79	260.64

TABLE 5
$\kappa = \mathrm{Cond}\,(B_{11}^{(k)^{-1}} A_{11}^{(k)})$; *mixed anisotropy:* $a_2 = a_4 = b_2 = b_4 = 1$.

Levels	$a_1 = 1, b_1 = 1$ $a_3 = 1, b_3 = 1$	$a_1 = 1, b_1 = 10$ $a_3 = 10, b_3 = 1$	$a_1 = 1, b_1 = 100$ $a_3 = 100, b_3 = 1$	$a_1 = 1, b_1 = 1000$ $a_3 = 1000, b_3 = 1$
3	1.460	5.232	42.49	414.78
4	1.489	5.641	46.48	454.32
5	1.496	5.852	48.70	348.36
6	1.498	5.940	49.69	366.76
7	1.499	5.971	50.27	452.36

In Tables 1–5 we show the values of the condition numbers $\mathrm{Cond}(B_{11}^{(k)^{-1}} A_{11}^{(k)})$.

The numerical results are in full agreement with the theoretical estimates. One can see that the studied multilevel algorithm is optimal if $a \leq b$ (which corresponds to the construction of the approximation $B_{11}^{(k)}$ tested), independent of the size of the jump of the coefficients. If $a \geq b$, the approximation $B_{11}^{(k)}$ loses its optimal order and the corresponding relative condition numbers increased strongly with $\frac{a}{b}$.

These results show that following the so-called strategy of "preserving the links along the dominated anisotropy" is of principal importance for the proposed multilevel algorithm.

Finally we show the algebraic multilevel preconditioner from Axelsson and Vassilevski [5] in the modified version of Vassilevski [16] based on a block-incomplete LU approximation of $A_{11}^{(k)}$. Note that if we use line ordering of the nodes from $\mathcal{N}^{(k)} \setminus \mathcal{N}^{(k-1)}$ (in y-direction, for example), the block $A_{11}^{(k)}$ admits a block-tridiagonal form with (scalar) tridiagonal blocks on its main diagonal. The block-ILU approximation to $A_{11}^{(k)}$ we used was based on 5-diagonal approximations to the inverses of the approximate Schur complements required throughout the process of block-ILU factorization. In this case we tested the case of constant coefficients in Ω with values of a and b shown in Table 6. We also tested discontinuous coefficients with mixed anisotropy. The values of $a_i = a|_{\Omega_i}$ and $b_i = b|_{\Omega_i}$, $i = 1, 2, 3, 4$ are shown in Table 7 (see Fig. 3 for the partitioning of Ω, $\Omega = \cup_{i=1}^{4} \Omega_i$).

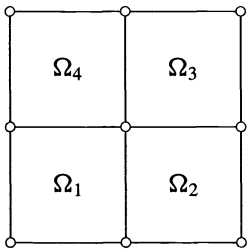


FIG. 3. *Splitting of the domain* $\Omega = (0, 1) \times (0, 1)$.

TABLE 6
$\kappa = \text{Cond}\left(M^{(k)^{-1}} A^{(k)}\right)$; $B_{11}^{(k)}$ – *ILU; smooth coefficients.*

Levels	$a = 1$ $b = 1$	$a = 1$ $b = 1000$	$a = 1000$ $b = 1$
3	2.149	2.241	2.242
4	2.272	2.330	3.328
5	2.339	2.360	2.349
6	2.370	2.366	2.346
7	2.387	2.362	2.362

TABLE 7
$\kappa = \text{Cond}\left(M^{(k)^{-1}} A^{(k)}\right)$; $B_{11}^{(k)}$ – *ILU; discontinuous coefficients.*

	Ω	$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	$\Omega_1, \Omega_2, \Omega_3, \Omega_4$	$\Omega_1, \Omega_2, \Omega_3, \Omega_4$
Levels	$a = 1$ $b = 1$	$a = 1$ $10^3, 1, 10^3, 1$	$10^3, 1, 10^3, 1$ $b = 1$	1,1,1000,1 1000, 1, 1, 1
3	2.149	2.234	2.228	2.235
4	2.272	2.293	2.294	2.300
5	2.339	2.348	2.340	2.348
6	2.370	2.345	2.345	2.348
7	2.387	2.358	2.356	2.360

The obtained numerical results show (see Tables 6–7) an optimal relative condition number of $M^{(k)^{-1}} A^{(k)}$ in this case independent of the size of the jumps of the coefficients and for large values of $\frac{a}{b}$ and $\frac{b}{a}$. This last test confirms the known robustness of the ILU-like approximations (see, e.g., Eijkhout and Vassilevski [11]) for solving more general elliptic problems including discontinuous and anisotropic coefficients now in the multilevel methods as well.

Acknowledgment. The authors are grateful to Professor Richard Ewing for the hospitality and support given through the Institute for Scientific Computation, University of Wyoming, Laramie, Wyoming.

REFERENCES

[1] O. AXELSSON, *On algebraic multilevel iteration methods for selfadjoint elliptic problems with anisotropy*, Report FSU-SCRI-90-80, Supercomputer Computations Research Institute, The Florida State University, Tallahassee, FL, 1990.

[2] O. AXELSSON AND V. EIJKHOUT, *Analysis of recursive 5-point/9-point factorization method*, in Proc. Conf. on PCG Methods, O. Axelsson and L. Kolotilina, eds., Nijmegen, The Netherlands, June, 1989; Lecture Notes in Math. 1457, Springer-Verlag, Berlin, New York, 1990, pp. 154–173.

[3] O. AXELSSON AND I. GUSTAFSSON, *Preconditioning and two-level multigrid methods of arbitrary degree of approximation*, Math. Comp., 40 (1983), pp. 219–242.

[4] O. AXELSSON AND P. S. VASSILEVSKI, *Algebraic multilevel preconditioning methods*, I, Numer. Math., 56 (1989), pp. 157–177.

[5] ———, *Algebraic multilevel preconditioning methods*, II, SIAM J. Numer. Anal., 27 (1990), pp. 1569–1590.

[6] R. BANK AND T. DUPONT, *Analysis of a Two-Level Scheme for Solving Finite Element Equations*, Report CNA-159, Center for Numerical Analysis, The University of Texas at Austin, 1980.

[7] J. DONATO AND T. F. CHAN, *Fourier analysis of incomplete factorization preconditioners for* 3-D *anisotropic problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 319–338.

[8] I. S. DUFF AND G. A. MEURANT, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.

[9] V. EIJKHOUT, *Analysis of parallel incomplete point factorizations*, Linear Algebra Appl., 154/156 (1991), pp. 723–740.

[10] V. EIJKHOUT AND P. S. VASSILEVSKI, *The role of the strengthened Cauchy-Buniakowskii-Schwarz inequality in multilevel methods*, SIAM Rev., 33 (1991), pp. 405–419.

[11] ———, *Positive definiteness aspects of vectorizable preconditioners*, Parallel Comput., 10 (1989), pp. 93–100.

[12] W. HACKBUSCH, *The frequency decomposition multi-grid method. 1. Application to anisotropic equations*, Numer. Math., 56 (1989), pp. 219–245.

[13] Y. A. KUZNETSOV, *Algebraic multigrid domain decomposition methods*, Soviet J. Numer. Anal. Math. Modelling, 4 (1989), pp. 351–380.

[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 Multigrid Methods, Proceedings, W. Hackbusch and U. Trottenberg, eds., Köln-Porz, Berlin, Germany, 1981; Lecture Notes in Math. 960, Springer-Verlag, Berlin, New York, 1982, pp. 201–216.

[15] S. D. MARGENOV, *Inverse-free multilevel methods* I, Report 4, Bulgarian Academy of Sciences, Center of Informatics and Computer Technology, Sofia, Bulgaria, 1989.

[16] P. S. VASSILEVSKI, *Hybrid V-cycle algebraic multilevel preconditioners*, Math. Comp., 58 (1992), pp. 489–512.

[17] P. S. VASSILEVSKI AND M. H. ETOVA, *Computation of constants in the strengthened Cauchy inequality*, SIAM J. Sci. Statist. Comp., 13 (1992), pp. 655–665.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1038–1058, September 1994

003

A LAGRANGIAN RANDOM CHOICE APPROACH FOR SUPERSONIC REAL GAS FLOWS*

CHING-YUEN LOH† AND MENG-SING LIOU†

Abstract. This paper presents a random choice version of the Lagrangian approach for supersonic real gas flows, reported previously by Loh and Liou [*J. Comput. Phsy.*, 104 (1993), pp. 150–161]. The exact real gas Riemann solution and the random choice method (RCM) are briefly reviewed and the derivation of the Lagrangian geometrical quantities, which represent the deformation of fluid particles in the motion, are described in detail. Extensive calculations were made to test the accuracy against the exact solution and the robustness of the Lagrangian RCM approach for real gas supersonic flows, including complex wave interactions of different types. The real gas effect is also presented by comparison with the perfect gas solution. The inherent parallelism in the Lagrangian approach lends a natural application in the massively parallel computation.

Key words. Lagrangian description, random choice method, real gas, Riemann problem, supersonic steady flow

AMS subject classifications. 65C20, 65M05, 76N10, 70D10, 76L05

1. Introduction.

Enormous progress has been made in the past decades in computational fluid dynamics in terms of discretization and solution techniques. The decade of 1980s has witnessed exhaustive exploration of upwind, monotone schemes, notably the exact Riemann solver by Gudonov [1] and various approximate Riemann solvers or upwind flux splittings [2]–[5]. In addition to the above deterministic approaches, Glimm [6] proposed the random choice method (RCM) for constructive proof of existence of solutions to nonlinear hyperbolic systems of conservation laws. Later Chorin [7] developed the RCM as a practical computational method for finding such solutions, especially to the one-dimensional (1-D) Euler equations of gas dynamics. Further improvements and applications of the method have been made by Colella [8], Sod [9], and Concus and Proskurowski [10]. In essence, the solution is advanced in time by a sequence of operations that includes the solution of Riemann problems and a sampling procedure. Recently, the idea of random choice has been extended and used in a broad way, such as in the vortex method [11] and front tracking method [12].

For one-dimensional flow the random choice method has infinite resolution with randomly off-position and is very easy to apply. For two-dimensional (2-D) steady supersonic flow, Marshall and Plohr [13] treated one of the space variables as a time-like variable and applied the random choice method in a way similar to that for one-dimensional unsteady flow. Oliver and Grönig [14] also have reported a successful application of the random choice method to 2-D shock focusing and diffraction when the shock Mach number is low. All these works using the random choice method are based on the Eulerian method of description of fluid flow. However, its very desirable properties—no diffusive errors from spatial averaging—do not seem to persist for flows in genuinely two space dimensions. Collela [8] suggested possible ways of dealing with this problem. Just as it is done in the deterministic approaches for multidimensional flows, allowing some approximations and reduction in accuracy, may still lead to satisfactory results. It seems too early to judge the ultimate fate of the RCM in solving the multidimensional flows for there is very limited finding reported in the literature, let alone in the Lagrangian framework.

Using their earlier new Lagrangian conservation formulation [16], Loh and Hui [15] recently have applied the random choice method for solving the 2-D Euler equations of ideal gas. This is made possible by the introduction of the *Lagrangian time* τ, which effectively

*Received by the editors March 3, 1993; accepted for publication July 1, 1993.

†Internal Fluid Mechanics Division, MS 5-11, NASA Lewis Research Center, Cleveland, Ohio 44135 (fsml@yinyan.lerc.nasa.gov).

reduces a 2-D steady supersonic/hypersonic flow problem to that of an 1-D unsteady flow. Compared to the Eulerian RCM methods such as [13], the Lagrangian method has the advantage of producing a smooth body surface (rather than a randomly fluctuated staircase like one), and it is an easier procedure to choose sample states. In the new Lagrangian approach, a computational cell is literally a fluid particle and the grid is automatically generated along streamlines as part of the solution.

With the renewed interest in high-speed flight, the real gas and nonequilibrium effects must be taken into account in the analysis of flow. Although there is no conceptual difficulty in the extension, however, some generalization must be made. In a recent paper [17], Loh and Liou have developed a deterministic Lagrangian approach for computing 2-D supersonic real gas flows. Encouraged by the above successful explorations, the purpose of this paper is to apply Glimm's random choice method to compute 2-D steady inviscid supersonic/hypersonic real gas flows formulated by the new Lagrangian description. In the Lagrangian random choice approach, the real gas Riemann solver, which Loh and Liou described in [17], is still a basic building block and the exact solution of the Riemann problem, instead of the approximate solution, is sought for in our study. Apart from the above real gas Riemann solution, the solution to the Lagrangian geometrical quantities play an important role and is required in the RCM approach. This solution procedure will be given in detail in this paper.

To avoid repetition, we briefly present in §2 the Lagrangian form of conservative laws, the equation of state (EOS) used for the equilibrium air and the exact Riemann solution of the Lagrangian formulation for real gases. In §3, the solution to the Lagrangian geometrical quantities are described in detail and the application of random choice method is outlined. Finally, several test problems are given in §4, followed by our concluding remarks in §5.

2. The Lagrangian conservation form and the Riemann solver for real gas.

2.1. The Lagrangian conservation forms. Based on the Lagrangian formulation of Hui and Van Roessel [18], Loh and Hui [16] used the stream function ξ and the "Lagrangian time" τ as the independent variables (Fig. 1) to compute 2-D steady supersonic flows. The Lagrangian time τ is interpreted as the physical time assigned for each fluid particle (computational cell). The conservation form (τ- conservation form) based on the transformation of this set of variables is given as follows:

$$\frac{\partial \mathbf{E}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \xi} = 0 \tag{1}$$

with

$$\mathbf{E} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{pmatrix} = \begin{pmatrix} K \\ H \\ Ku + pV \\ Kv - pU \\ U \\ V \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} 0 \\ 0 \\ -pv \\ pu \\ -u \\ -v \end{pmatrix}.$$

As usual, u, v, p, and ρ are, respectively, Cartesian velocity components, $\mathbf{V} = (u, v)^T$, pressure, and density of the fluid; and

$$\mathbf{T} = (U, V)^T = \left(\frac{\partial x}{\partial \xi}, \quad \frac{\partial y}{\partial \xi}\right)^T$$

are the geometrical quantities representing the deformation of fluid particle as it moves downstream. The quantity

$$K = \rho \begin{vmatrix} u & v \\ U & V \end{vmatrix} = \rho(uV - vU) \tag{2}$$

is the mass flux, and H is the specific total enthalpy,

$$H = \tfrac{1}{2}(u^2 + v^2) + h(p, \rho), \tag{3}$$

where h is the enthalpy. The first four equations in (1) represent the physical conservation laws of mass, energy, and momentum, respectively; they are equivalent to the Euler equations of gas dynamics. The last two equations arise from the compatibility condition between the τ-derivatives and the ξ-derivatives, representing the deformation of a fluid particle.

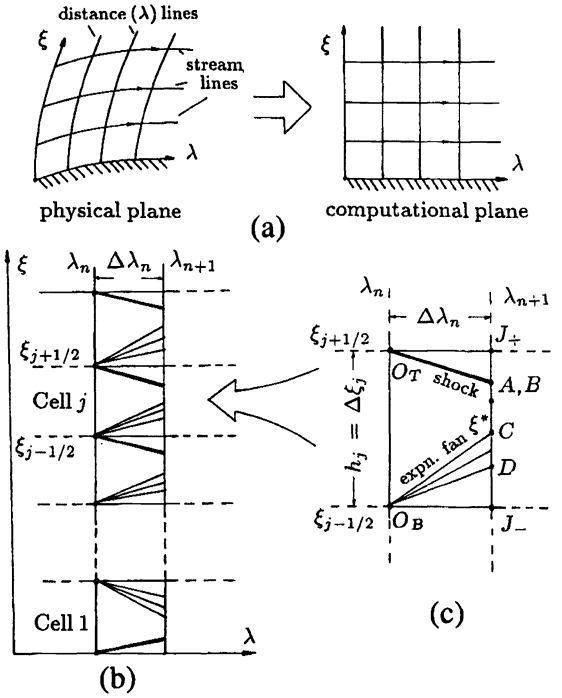


FIG. 1. (a) *Computational plane and* (b) *mesh*; (c) *wave structure in a typical cell j.*

At first glance, (1) seems to imply that more equations in the present Lagrangian formulation, six rather than four used in the Eulerian formulation, need to be integrated, and the computational effort is increased. Close examination of (1) immediately eliminates the need to integrate the first two equations for they remain constant along each ξ =constant line. Hence the present Lagrangian formulation also solves four partial differential equations (pdes) and requires the solution of geometrical compability (conservation) instead. The effort of solving for this Lagrangian coordinates is simple. While we do not have a direct comparison of computational efficiency between the Lagrangian and Eulerian formulations, we expect they are comparable. On the other hand, the former approach delivers higher accuracy, as clearly shown in [16], and saves grid generation prior to the calculation as required by the latter.

In this τ conservation form, each fluid particle marches forward with the same time step $\Delta\tau$ according to its own velocity. A drawback is that across a strong contact discontinuity

(slip line) where flow velocity may be discontinuous, two adjacent fluid particles initially in physical contact may eventually be separated from each other, rendering it difficult to apply a local Riemann solver. A simple remedy is to keep these two particles marching at the same pace by changing time step size on one side. More generally, we can let all the fluid particles march the same distance $\Delta\lambda$ instead of the same time step along their own streamlines. This idea leads to another new Lagrangian conservation form—the conservation form based on the Lagrangian distance λ. Hui and Zhao give all the details in their recent paper [19]. We present here only a sketch of the necessary steps.

First, we define the Lagrangian distance as the distance along a streamline

$$\lambda = \int_0^\tau q d\tau, \tag{4}$$

where the flow speed

$$q = (u^2 + v^2)^{1/2}.$$

The other independent variable, the stream function, remains the same as before,

$$\xi_1 = \xi.$$

Let

$$\alpha = \frac{\partial \lambda}{\partial \xi},$$

and since

$$\frac{\partial}{\partial \lambda} = \frac{1}{q}\frac{\partial}{\partial \tau},$$

a useful relation can be easily derived

$$\frac{\partial \alpha}{\partial \lambda} = \frac{1}{q}\frac{\partial q}{\partial \xi}. \tag{5}$$

Now, we can make coordinate transformation from (λ, ξ_1) to (τ, ξ) and the Jacobian J_1 is

$$J_1 = \frac{\partial(\lambda, \xi_1)}{\partial(\tau, \xi)} = \begin{pmatrix} \partial\lambda/\partial\tau & \partial\xi_1/\partial\tau \\ \partial\lambda/\partial\xi & \partial\xi_1/\partial\xi \end{pmatrix} = \begin{pmatrix} q & 0 \\ \alpha & 1 \end{pmatrix}.$$

The inverse of J_1 is

$$J_1^{-1} = \frac{\partial(\tau, \xi)}{\partial(\lambda, \xi_1)} = \begin{pmatrix} \partial\tau/\partial\lambda & \partial\xi/\partial\lambda \\ \partial\tau/\partial\xi_1 & \partial\xi/\partial\xi_1 \end{pmatrix} = \begin{pmatrix} 1/q & 0 \\ -\alpha/q & 1 \end{pmatrix}.$$

Moreover, we define a new geometrical variable vector $\mathbf{T}_1$ in place of $\mathbf{T}$,

$$\mathbf{T}_1 = (U_1, V_1)^T = \mathbf{T} - \alpha\frac{\mathbf{V}}{q}.$$

Similar to the compability equations in (1), it can be shown that

$$\frac{\partial \mathbf{T}_1}{\partial \lambda} = \frac{\partial \frac{\mathbf{V}}{q}}{\partial \xi_1}. \tag{6}$$

We note that the mass flux (2) remains unchanged during the coordinate transformation:

$$K = \rho \begin{vmatrix} u & v \\ U & V \end{vmatrix} = \rho \begin{vmatrix} u & v \\ U_1 & V_1 \end{vmatrix}.$$

After some manipulation and then dropping the subscript "1," we achieve a complete set of the new Lagrangian conservation form based on the λ-variable

$$\frac{\partial \mathbf{E}}{\partial \lambda} + \frac{\partial \mathbf{F}}{\partial \xi} = 0, \tag{7}$$

with

$$\mathbf{E} = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{pmatrix} = \begin{pmatrix} K \\ H \\ Ku + pV \\ Kv - pU \\ U \\ V \end{pmatrix}, \qquad \mathbf{F} = \begin{pmatrix} 0 \\ 0 \\ -pv/q \\ pu/q \\ -u/q \\ -v/q \end{pmatrix}.$$

Hereafter in this paper we consider only the λ-formulation.

For steady flow at supersonic speed, (1) or (7) holds for any gas. For real gases, the equation of state (EOS) is more complicated, but any pair of independent thermodynamic variables are sufficient to describe the state. In our formulation, we choose

$$h = h(p, \rho) = e(p, \rho) + \frac{p}{\rho}.$$

The internal energy e is a prescribed function of p and ρ. The one we use throughout the present paper is an EOS developed by Tannehill [20], which is based on table look-up ploynomial interpolation, with a reported maximum error of 4%. This EOS is known to yield nonsmoothness in the fitted functions; it is still chosen simply because it is widely referenced. The investigation of its range of validity is beyond the scope of this paper.

2.2. The Riemann solver for real gas. We turn now to consider the Riemann solver, which forms the basic building block in RCM. The solution procedure for the real gas Riemann problem is substantially different from the one for perfect gas, although in principle they are similar; the details have been given in [17]. For completeness, we briefly outline this procedure.

The Riemann problem for two-dimensional steady supersonic real gas flow is the initial value problem with the constant data

$$\mathbf{Q} = \begin{cases} \mathbf{Q}_T, & \xi \geq \xi_j, \\ \mathbf{Q}_B, & \xi < \xi_j \end{cases} \tag{8}$$

as initial condition at λ=0 for the flow state $\mathbf{Q} = (p, \rho, u, v)^T$ and with the EOS

$$e = e(p, \rho), \tag{9}$$

where the subscripts T and B denote top and bottom states, which are counterparts to the left and right states in one-dimensional unsteady flow. The EOS (9) is prescribed as a given function.

It should be emphasized that since a Riemann problem is a physical one and only the flow variables u, v, p, ρ are required in the RCM solution, the numerical problem is solved in terms of u, v, p, ρ rather than in terms of the conservative variables $e_1, \ldots, e_6$.

The solution of the Riemann problem is self-similar in the variable $\zeta = \lambda/\xi$ and in general consists of three types of elementary waves: the oblique shock, the slip line, and the expansion fan (Fig. 1(b)); through any state $\mathbf{Q}_0$, with p as a parameter, there are two families of state connecting to $\mathbf{Q}_0$, namely, the compression states ($p \geq p_0$) and the expansion states ($p < p_0$). As in perfect gas, these two families join smoothly at $\mathbf{Q}_0$ and can be regarded as a single family. This makes it possible to apply Newton's iterative procedure in the solution of the Riemann problem. We now briefly illustrate the details of the solution.

(i) In the $p-\theta$ plane (Fig. 2), two curves pass through the states $\mathbf{Q}_0 = \mathbf{Q}_T$ and $\mathbf{Q}_0 = \mathbf{Q}_B$ and they are defined as

$$\theta = \Phi_T(p) = \begin{cases} \theta_0 + \displaystyle\int_{p_0}^{p} \frac{\sqrt{M^2-1}}{\rho q^2}\,dp, & p \leq p_0, \\ \theta_0 + \tan^{-1}\left[\dfrac{(\rho-\rho_0)\tan\chi}{\rho + \rho_0 \tan^2\chi}\right], & p > p_0; \end{cases} \tag{10a}$$

and

$$\theta = \Phi_B(p) = \begin{cases} \theta_0 - \displaystyle\int_{p_0}^{p} \frac{\sqrt{M^2-1}}{\rho q^2}\,dp, & p \leq p_0, \\ \theta_0 - \tan^{-1}\left[\dfrac{(\rho-\rho_0)\tan\chi}{\rho + \rho_0 \tan^2\chi}\right], & p > p_0. \end{cases} \tag{10b}$$

In the compression state, an implicit relation between p and ρ, namely, the Rankine–Hugoniot relation, exists

$$\frac{\rho_0}{\rho} = 1 - 2\rho_0 \frac{e(p,\rho) - e(p_0,\rho_0)}{p + p_0}. \tag{11}$$

This determines ρ as a function of p. The shock angle χ is evaluated through

$$\chi = \sin^{-1}\left[\sqrt{\frac{(p-p_0)\rho}{(\rho-\rho_0)\rho_0}}/q_0\right]. \tag{12}$$

In the expansion state, with initial condition $\rho = \rho_0$ and by Runge–Kutta integration, ρ is evaluated through an ordinary differential equation (ode) as a function of p:

$$\frac{d\rho}{dp} = \frac{1}{a^2} = \frac{e_p}{p/\rho^2 - e_\rho}, \tag{13}$$

where

$$a^2 = \left(\frac{dp}{d\rho}\right)_s = \frac{p/\rho^2 - e_\rho}{e_p}, \quad e_p = \frac{\partial e}{\partial p}, \quad e_\rho = \frac{\partial e}{\partial \rho},$$

$$q^2 = 2\left(H - e(p,\rho) - \frac{p}{\rho}\right), \qquad M^2 = \frac{q^2}{a^2} = \frac{2(H - e(p,\rho) - p/\rho)e_p}{p/\rho^2 - e_\rho}.$$

These curves Φ_T, Φ_B are sketched in Fig. 2.

(ii) The modified Newton procedure, which is a hybrid of bisection and Newton's iteration method, as decribed in [17], is then employed to find the intersect (p^*, θ^*) of the two

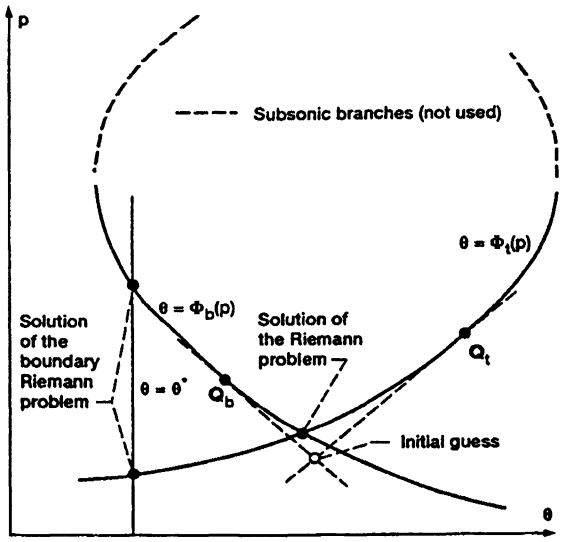


FIG. 2. *Solution of real gas Riemann problem.*

curves. The bisection procedure in the method is required to overcome any possible trouble in the Newton procedure caused by the inflection points in the table look-up interpolated EOS function. The object function to be driven to zero in this iterative procedure is

$$f(p) = \Phi_T(p) - \Phi_B(p) \tag{14}$$

and the intersect of the tangent lines passing through $\mathbf{Q}_T$ and $\mathbf{Q}_B$ (Fig. 2) is used as an initial guess to the solution. In practice we use numerical derivative to replace the analytical ones. Usually it takes two to four iterations to converge to a tolerance less than 10^{-6}.

(iii) With the slip line values (p^*, θ^*) obtained, we calculate ρ on either side across the slip line, using the appropriate equation ((11) for the compression state and (13) for the expansion state) and then calculate q using (3), namely,

$$q^2 = 2[H - h(p, \rho)].$$

Velocity components are easily obtained by

$$u = q\cos\theta^* \quad \text{and} \quad v = q\sin\theta^*. \tag{15}$$

At the solid boundary, the flow inclination condition is imposed and one of the curves, say, $\theta = \Phi_B(p)$ degenerates to a a straight line $\theta = \theta^* = \text{const.}$, i.e., parallel to the p axis (Fig. 2). In [16], this particular problem is termed "boundary Riemann problem." The solution of a boundary Riemann problem is similar to the above procedure with the exception of using a different object function

$$f(p) = \Phi_T(p) - \theta^*$$

or

$$f(p) = \Phi_B(p) - \theta^*.$$

3. Solution to the Lagrangian geometrical quantities $\mathbf{T}=(U, V)^T$ and application of the random choice method.

3.1. Solution to the Lagrangian geometrical quantities. Associated with the Riemann solution, the geometrical quantities U and V are to be solved as part of the solution prior to the application of the RCM. Indeed, the Riemann solution combined with the geometrical quantities U and V may be considered as a complete Lagrangian Riemann solution. In this section, we first detail the computation of $\mathbf{T} = (U, V)^T$ and then proceed to the application of the RCM. Here, only the formulations for the "upright" case, i.e., the case with the slipline at the bottom, are presented; formulations for the "upside down" case follow in a similar way.

The basis for evaluating $\mathbf{T} = (U, V)^T$ is the compatibility equations in (7):

$$\frac{\partial \mathbf{T}}{\partial \lambda} = \frac{\partial \frac{\mathbf{V}}{q}}{\partial \xi}.$$

As in the Riemann solution procedure, we consider the solution for the elementary waves separately.

• Across an oblique shock.

We use subscript "0" to denote the given uniform flow state upstream of the shock and "s" the flow state downstream of the shock. Thus $\mathbf{Q}_0 = (p_0, \rho_0, u_0, v_0)^T$, $\mathbf{T}_0 = (U, V)^T$, $\theta_0 = \tan^{-1}(v_0/u_0)$, $\beta = \tan^{-1}(V/U)$, $\beta_0 = \tan^{-1}(V_0/U_0)$, and $T_0 = (U_0^2 + V_0^2)^{1/2}$. The shock angle χ is obtained from (12). Let the streamline ξ intersect the initial λ line at A and then the oblique shock at B. From the triangle OAB (Fig. 3), for any point (λ_s, ξ_s) immediately downstream of the shock, we have

$$\zeta_s = \frac{\lambda_s}{\xi_s} = \frac{T_0 \sin(\beta_0 - \chi - \theta_0)}{\sin \chi}. \tag{16}$$

By applying the divergence theorem to the quadrilateral OABC (Fig. 3), where BC is another λ line, we have

$$\mathbf{T}_s = \mathbf{T}_0 + \zeta_s \left(\frac{\mathbf{V}_0}{q_0} - \frac{\mathbf{V}_s}{q_s} \right). \tag{17}$$

In summary, the complete Riemann solution in the compression state is

$$(\mathbf{Q}, \mathbf{T}) = \begin{cases} (\mathbf{Q}_0, \mathbf{T}_0), & \zeta < \zeta_s, \\ (\mathbf{Q}_s, \mathbf{T}_s), & \zeta \geq \zeta_s \end{cases} \tag{18}$$

where $\mathbf{Q}_0$, $\mathbf{Q}_s$ are, respectively, the flow states upstream and downstream of the shock from the local Riemann solver.

• Across an expansion fan.

In this case, we need the relations between the self-similarity variable $\zeta = \lambda/\xi$, ϕ and eventually pressure p. The local polar coordinate (r, ϕ) is adopted in the analysis; see Figs. 4 and 5.

Here we use subscript "0" to denote quantities upstream of or at the forward Mach line location and the subscript "1" to denote quantities downstream of or at the rearward Mach line location. In the uniform flow region (Region I in Fig. 5) upstream of the forward Mach line, $\zeta \leq \zeta_0$, obviously,

$$\mathbf{T} = \mathbf{T}_0,$$

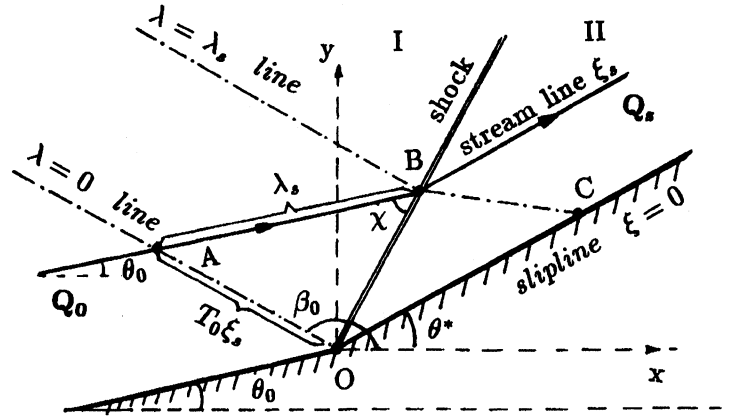


FIG. 3. *Computing U, V in a compression state.*

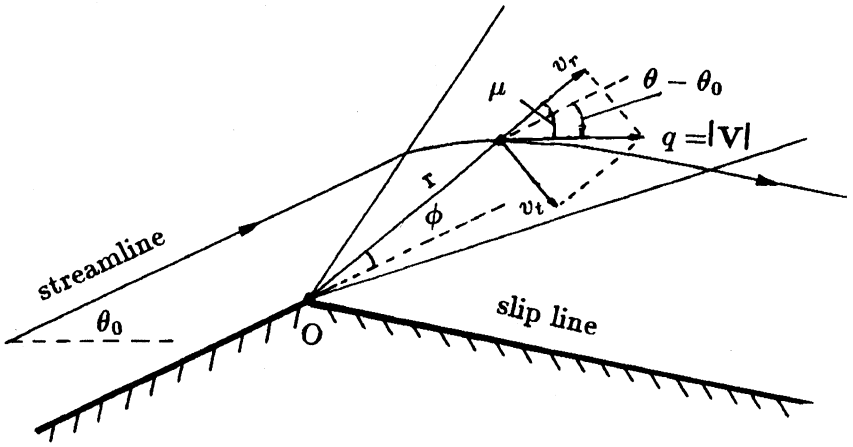


FIG. 4. *Showing* $\phi = \mu + \theta - \theta_0$.

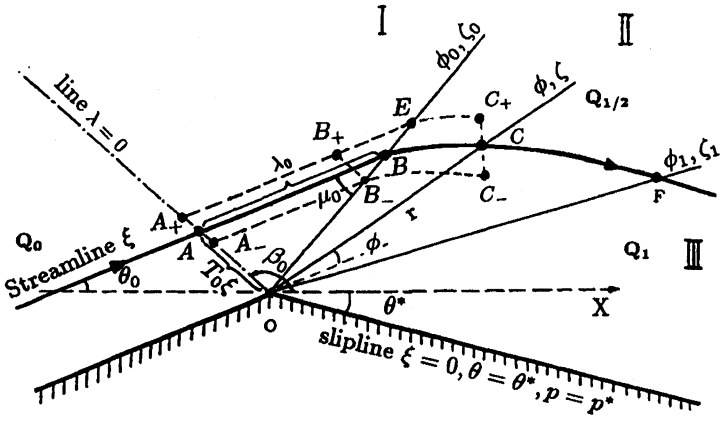


FIG. 5. *Computing U, V in an expansion state.*

where, from the triangle OAB in Fig. 5,

$$\zeta_0 = \frac{\lambda_0}{\xi_0} = T_0 M_0 \sin(\beta_0 - \mu_0 - \theta_0). \tag{19}$$

In the expansion fan region (Region II in Fig. 5), $\zeta_0 \leq \zeta \leq \zeta_1$, the Euler equations in polar coordinates can be reduced to the following ode system:

$$\begin{gathered}
\left(v_r + \frac{dv_t}{d\phi}\right) + \frac{v_t}{\rho}\frac{d\rho}{d\phi} = 0, \\
\frac{dv_r}{d\phi} = v_t, \\
v_t\left(v_r + \frac{dv_t}{d\phi}\right) + \frac{1}{\rho}\frac{dp}{d\phi} = 0, \\
e = e(p, \rho), \\
H = \text{ const.},
\end{gathered} \tag{20}$$

and

$$\frac{dp}{d\rho} = a^2,$$

where v_r and v_t are, respectively, the radial and transversal velocities, as illustrated in Fig. 4. From the first, third, and sixth equations of (20),

$$\frac{dp}{d\phi} = {v_t}^2 \frac{d\rho}{d\phi},$$

$${v_t}^2 = \frac{dp}{d\rho} = a^2, \qquad {v_r}^2 = q^2 - a^2. \tag{21}$$

The polar angle is

$$\phi = \mu + \theta - \theta_0,$$

where θ is the flow inclination, θ_0 the initial flow inclination, and μ the local Mach angle.

After a small algebraic manipulation, we obtain the following equations:

$$d\theta = \frac{(M^2 - 1)^{1/2}}{\rho q^2} dp,$$

$$d\mu = d \sin^{-1} \frac{1}{M} = -\frac{1}{M(M^2 - 1)^{1/2}} dM,$$

and

$$\begin{aligned}
d\phi = d\mu + d\theta &= \frac{(M^2 - 1)}{\rho q^2} dp - \frac{1}{M(M^2 - 1)^{1/2}} dM \\
&= \left[\frac{(M^2 - 1)^{1/2} + (M^2 - 1)^{-1/2}}{\rho q^2} + \frac{a'(p)}{a(M^2 - 1)^{1/2}}\right] dp = S(p) dp.
\end{aligned} \tag{22}$$

To find the relation between ϕ, ζ and p, we start with

$$\frac{dr}{v_r} = r\frac{d\phi}{v_t} = d\tau = \frac{d\lambda}{q}. \tag{23}$$

Since $v_t = -a$ and $v_r = (q^2 - a^2)^{1/2}$, we obtain

$$\frac{dr}{r} = \frac{v_r}{v_t}d\phi = -(M^2 - 1)^{1/2}d\phi,$$

$$r = \xi A_0 M R(\phi) \quad \text{with } R(\phi) = \exp\left(-\int_{p_0}^{p} \frac{M^2-1}{\rho q^2}dp\right),$$

where $A_0 = T_0 \sin(\beta_0 - \theta_0) = (U_0{}^2 + V_0{}^2)^{1/2} \sin(\beta_0 - \theta_0)$ and ξ remains constant along a streamline. From (23),

$$d\lambda = \frac{qr}{v_t}d\phi = -Mrd\phi,$$

$$\lambda = \lambda_0 - \xi A_0 \int_{\phi_0}^{\phi} M^2 R(\phi)d\phi. \tag{24}$$

By virtue of (22), we have

$$\zeta = \frac{\lambda}{\xi} = \zeta_0 - A_0 \int_{p_0}^{p} M^2 R(\phi(p))S(p)dp, \tag{25}$$

and in particular,

$$\zeta_1 = \zeta_0 - A_0 \int_{p_0}^{p_1} M^2 R(\phi(p))S(p)dp, \tag{26}$$

$$\zeta_{1/2} = \zeta_0 - A_0 \int_{p_0}^{p_{1/2}} M^2 R(\phi(p))S(p)dp, \tag{27}$$

where p_1 is the pressure downstream of the rearward Mach line, obtained from the Riemann solver, and $p_{1/2} = 1/2(p_0 + p_1)$ is the pressure in the middle of the fan.

With the above preliminaries, we first apply the divergence theorem to the streamtube area $AA_+B_+EC_+C_-B_-A_-A$, which is bounded by a pair of streamlines $\xi - \epsilon$ ($A_-B_-C_-$) and $\xi + \epsilon$ ($A_+B_+EC_+$) and two λ lines λ_0 (A_-AA_+) and λ (C_-CC_+). Then consider $\mathbf{T}$ as the limit when ϵ approaches zero:

$$\mathbf{T} = \lim \frac{1}{2\epsilon}\int_{\xi-\epsilon}^{\xi+\epsilon} \mathbf{T}d\xi = \mathbf{T}_0 + \lim \frac{1}{2\epsilon}\left[\int_{A_+B_+EC_+} \frac{\mathbf{V}}{q}d\lambda - \int_{A_-B_-C_-} \frac{\mathbf{V}}{q}d\lambda\right]$$

$$= \mathbf{T}_0 + \lim \frac{1}{2\epsilon}\left[\int_{B_+EC_+} \frac{\mathbf{V}}{q}d\lambda - \int_{B_-C_-} \frac{\mathbf{V}}{q}d\lambda\right], \tag{28}$$

since

$$\int_{A_+B_+} \frac{\mathbf{V}}{q}d\lambda = \int_{A_-B_-} \frac{\mathbf{V}}{q}d\lambda.$$

The limiting integrals on the right-hand side of (28) are evaluated by the l'Hôpital rule:

$$\lim \frac{1}{2\epsilon}\left[\int_{B_+EC_+} \frac{\mathbf{V}}{q}d\lambda - \int_{B_-C_-} \frac{\mathbf{V}}{q}d\lambda\right] = \frac{\xi_0}{\lambda_0}\frac{\mathbf{V}_0}{q_0} - A_0\int_{\phi_0}^{\phi} \mathbf{V}M\frac{R(\phi)}{a}d\phi - \frac{r}{a}\phi'(\xi)\mathbf{V}(\phi),$$

where $\phi'(\xi)$ can be found by differentiating (24) with respect to ξ (with λ held unchanged)

$$\phi'(\xi) = \frac{a}{r}\frac{\zeta}{q}.$$

We finally achieve an expression for $\mathbf{T}$ in the expansion region II:

$$\mathbf{T} = \mathbf{T}_0 + \zeta_0\frac{\mathbf{V}_0}{q_0} - \zeta\frac{\mathbf{V}}{q} - A_0\int_{p_0}^{p} \frac{M}{a}\mathbf{V}R(\phi)S(p)dp. \tag{29}$$

In particular, in Region III,

$$\mathbf{T}_1 = \mathbf{T}_0 + \zeta_0\frac{\mathbf{V}_0}{q_0} - \zeta_1\frac{\mathbf{V}_1}{q_1} - A_0\int_{p_0}^{p_1} \frac{M}{a}\mathbf{V}R(\phi)S(p)dp. \tag{30}$$

In the expansion fan, $\mathbf{T}$ can be approximated and simplified by a mean value when applied to the RCM:

$$\mathbf{T}_{1/2} = \mathbf{T}_0 + \zeta_0\frac{\mathbf{V}_0}{q_0} - \zeta_{1/2}\frac{\mathbf{V}_{1/2}}{q_{1/2}} - A_0\int_{p_0}^{p_{1/2}} \frac{M}{a}\mathbf{V}R(\phi)S(p)dp, \tag{31}$$

where the subscript "1/2" represents the values corresponding to $p = 1/2(p_0 + p_1)$ in the middle of the fan.

In summary, the complete Riemann solution in the expansion state is

$$(\mathbf{Q}, \mathbf{T}) = \begin{cases} (\mathbf{Q}_0, \mathbf{T}_0), & \zeta < \zeta_0, \\ (\mathbf{Q}_{1/2}, \mathbf{T}_{1/2}), & \zeta_0 \le \zeta \le \zeta_1, \\ (\mathbf{Q}_1, \mathbf{T}_1), & \zeta > \zeta_1, \end{cases} \tag{32}$$

where $\mathbf{Q}_0$, $\mathbf{Q}_1$ are, respectively, the flow states upstream of the forward Mach line and downstream of the rearward Mach line from the Riemann solver, and $\mathbf{Q}_{1/2}$ corresponds to $p = p_{1/2}$ representing approximately the flow variables in the expansion fan.

3.2. Application of the RCM. To solve the initial boundary value problem of the hyperbolic system (7), the computational domain (Fig. 1) is divided by streamlines $0 = \xi_0 < \xi_1 < \xi_2 < \cdots < \xi_N$, into N cells of size $h_j = \xi_j - \xi_{j-1}$, $j = 1, 2, \ldots, N$. Initially at $\lambda = \lambda_0$, the flow variables $\mathbf{Q} = (p, \rho, u, v)^T$ and the geometrical variables $\mathbf{T} = (U, V)^T$ are given along the initial λ line. The solution is to be evaluated for every cell $j = 1, 2, \ldots, N$, at $\lambda = \lambda_1, \lambda_2, \ldots$.

For $\lambda = \lambda_n$ the flow variables $\mathbf{Q}$ and $\mathbf{T}$ are assumed given and constant within each cell j, denoted as $\mathbf{Q}_j$, $\mathbf{T}_j$. A sequence of Riemann problems at all cell interfaces with initial data

$$\mathbf{Q} = \begin{cases} \mathbf{Q}_{j+1}, & \xi \ge \xi_j \\ \mathbf{Q}_j, & \xi < \xi_j \end{cases} \qquad j = 1, 2, \ldots, N-1,$$

are solved to determine the interaction between flows in adjacent cells and subsequently the solution regions in a cell (Fig. 1).

We denote the known flow states at λ_n in cell j as $(\mathbf{Q}_j^n, \mathbf{T}_j^n)^T$, $j = 1, 2, \ldots, N$ and restrict the marching step $\Delta\lambda_n = \lambda_{n+1} - \lambda_n$ to be small enough so that the waves resulting from the

interactions at both boundaries of cell j do not intersect with each other within the cell j, $j = 1, 2, \ldots, N$. Under this Courant–Friedrichs–Lewy (CFL) condition the flow inside the cell j at λ_{n+1} is composed of the lower side (compression or expansion) of the Riemann solution due to the interaction of cell j and cell $j+1$ and the upper side (compression or expansion) of the Riemann solution due to the interation of cell j and cell $j-1$, the two sides being separated in between by the original flow $(\mathbf{Q}_j^n, \mathbf{T}_j^n)^T$. There are several possible combinations. To be specific, we discuss a typical situation depicted in Fig. 1(c) where the flow in cell j at λ_{n+1} is composed of a compression side J_+A below the slip line $\xi_{j+1/2}$ and an expansion side J_-DC above the slip line $\xi_{j-1/2}$, and the original flow $(\mathbf{Q}_j^n, \mathbf{T}_j^n)^T$ in AC.

For this typical case, we first determine the values of the similarity variable ζ at A, B, C, D, namely, ζ_A, ζ_B, ζ_C, ζ_D as follows. Here, B coincides with A. We use $(\mathbf{Q}_j^n, \mathbf{T}_j^n)^T$ as the "0" state.

• For ζ_A and ζ_B:

Solve the Riemann problem between cells j and $j+1$ to get (p^*, θ^*), as explained in the above section. We then find the shock angle χ, and ζ_s from (16), and let

$$\zeta_A = \zeta_B = \zeta_s. \tag{33}$$

• For ζ_C:

We let ζ_C correspond to the forward Mach line,

$$\zeta_C = \zeta_0. \tag{34}$$

• For ζ_D:

Solve the Riemann problem between cells j and $j-1$ to get (p^*, θ^*) and then calculate ζ_1 from (26). Let

$$\zeta_D = \zeta_1. \tag{35}$$

In the case that the upper flow is composed of an expansion side instead of a compression one, $\zeta_A \neq \zeta_B$ and they are defined in a similar way as in (34) and (35).

In the random choice method, as suggested by many researchers [8], [12], the Van der Corput pseudorandom number generator [21] is the best choice, since it produces the least variance in the sense of l_1 and l_∞ norm. Throughout the present paper, we use a (2,1) Van der Corput pseudorandom number generator. A uniformly distributed pseudorandom number κ $(0 < \kappa < 1)$ is then generated to produce the sample point in the cell j:

$$\xi_j^* = \xi_j - \kappa h_j, \qquad (0 < \kappa < 1). \tag{36}$$

Only one random number is required at a marching step for all the cells j, $j = 1, 2, \ldots, N$. (see Fig. 1(c)). We now determine the flow state at this sample point (λ_{n+1}, ξ_j^*) and use it to represent the flow state of the Cell j. Let $k_n = \Delta\lambda_n$:

• If

$$0 < \kappa < \frac{k_n}{h_j \zeta_A}, \tag{37}$$

the point (λ_{n+1}, ξ_j^*) falls in J_+A (or J_+B), see Fig. 1(c), and the flow state is given by the second equation of (18).

• If

$$\frac{k_n}{h_j \zeta_A} \leq \kappa < 1 - \frac{k_n}{h_j \zeta_C}, \tag{38}$$

the point (λ_{n+1}, ξ_j^*) lies in BC (or AC) and the flow state remains unchanged and identical to the free stream state $(\mathbf{Q}_j^n, \mathbf{T}_j^n)^T$.

• If

$$1 - \frac{k_n}{h_j \zeta_C} \leq \kappa < 1 - \frac{k_n}{h_j \zeta_D}, \tag{39}$$

the point (λ_{n+1}, ξ_j^*) falls in the expansion fan CD and the flow state is given by the second equation of (32).

• If

$$1 - \frac{k_n}{h_j \zeta_D} \leq \kappa < 1, \tag{40}$$

the point (λ_{n+1}, ξ_j^*) falls in DJ_- and the flow state is given by the third equation of (32).

We note that a unique feature of the Lagrangian formulation is that the cell boundaries are always slip lines. This ensures that there exist at most five regions of different states in a cell (this occurs when both top and bottom waves are composed of the expansion side of their corresponding Riemann solutions) and that the flows on opposite sides of the cell boundary can be handled separately. The corresponding situation in the Eulerian formulation is more complicated.

The marching computation at the new λ line $\lambda = \lambda_{n+1}$ completes with the evaluation of the new cell locations (x_j^{n+1}, y_j^{n+1}), $j = 1, 2, \ldots, N$. This is done by simple integration along the streamlines:

$$\begin{aligned} x_j^{n+1} &= x_j^n + \frac{1}{2}\Delta\lambda_n \left(\frac{u_j^n}{q_j^n} + \frac{u_j^{n+1}}{q_j^{n+1}} \right), \\ y_j^{n+1} &= y_j^n + \frac{1}{2}\Delta\lambda_n \left(\frac{v_j^n}{q_j^n} + \frac{v_j^{n+1}}{q_j^{n+1}} \right). \end{aligned} \tag{41}$$

Thus, the procedure of marching by one step is completed. To march further forward, one goes back to the very beginning and repeats the procedure.

4. Test problems. To show the accuracy and robustness, the new Lagrangian random choice method is tested in several examples. These examples include features of the basic elementary waves, namely, the oblique shocks (+), the slip lines (0), and the Prandtl–Meyer expansions (–). The numerical results are compared with the available exact solutions. In some examples, we present the real gas solution as well as the perfect gas solution for comparison. Throughout the examples, the standard metric system is used. However, for convenience, the unit for pressure is atmospheric pressure (atm), the unit for temperature is K, and a unit for velocity is equivalent to 318.3 m/sec.

We first consider a pure initial value problem, namely a Riemann problem. The top and bottom states are shown in Fig. 6. In this example, we intend to validate that the real gas code works well for a perfect gas EOS. Here we use 40 uniform cells with $h_j = 0.01$ and the EOS is simply the one for perfect gas, i.e., $e = 1/(\gamma - 1)\ p/\rho$; $\gamma = 1.4$. From Fig. 6 it is seen that the numerical results agree well with the perfect gas exact solution (solid lines) and the slip line and shock are resolved sharply except that their positions are slightly off in a random way.

We then turn to consider initial-boundary value problems. The second example is the Prandtl–Meyer flow with a turning angle of 15° at the body surface. The free stream state is $p = 1$, $\rho = 1$ and $M = 6$. In this problem 100 uniform cells are used. Figure 7 illustrates

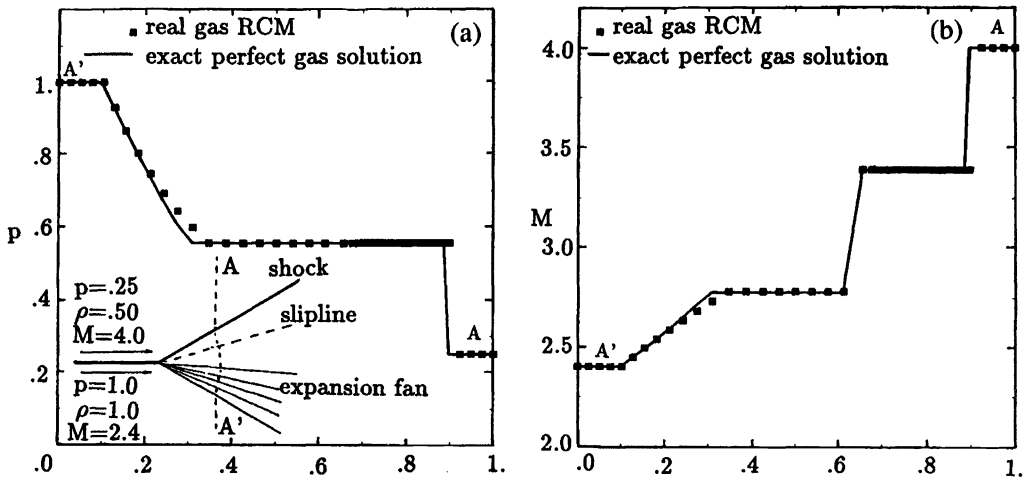


FIG. 6. *Numerical results with the real gas* RCM *for a perfect gas Riemann problem*: (a) *pressure*, (b) *Mach number along a typical* λ *line* AA.

the numerical result by the real gas random choice code developed in the present paper. For comparison the perfect gas exact solution is also included. This problem is another low temperature case and the numerical result agrees very well with the exact solution.

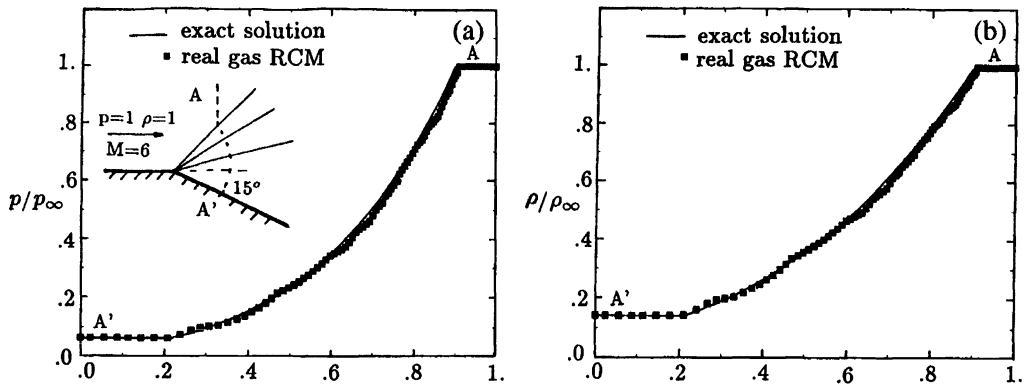


FIG. 7. *Computed flow variables for a perfect gas* P-M *flow*: (a) *pressure and* (b) *density along a typical* λ *line.*

In the next example, we still consider a Prandtl–Meyer expansion. However, the temperature is much higher so that the real gas effect is significant. We intend to show the numerical results with real gas effect. In this case the turning angle of the body surface is 20° and the free stream data reads

$$p_\infty = 2, \quad \rho_\infty = 0.25, \quad u_\infty = 15, \quad v_\infty = 0.$$

Here 44 uniform cells are employed with $h = 0.01$. With the same free stream data, numerical computations using real gas EOS are performed. The results are illustrated in Fig. 8. For comparison both real gas and perfect gas exact solutions are also included. It is observed, with exactly the same initial data, that there are differences due to the real gas effect in the pressure and densities.

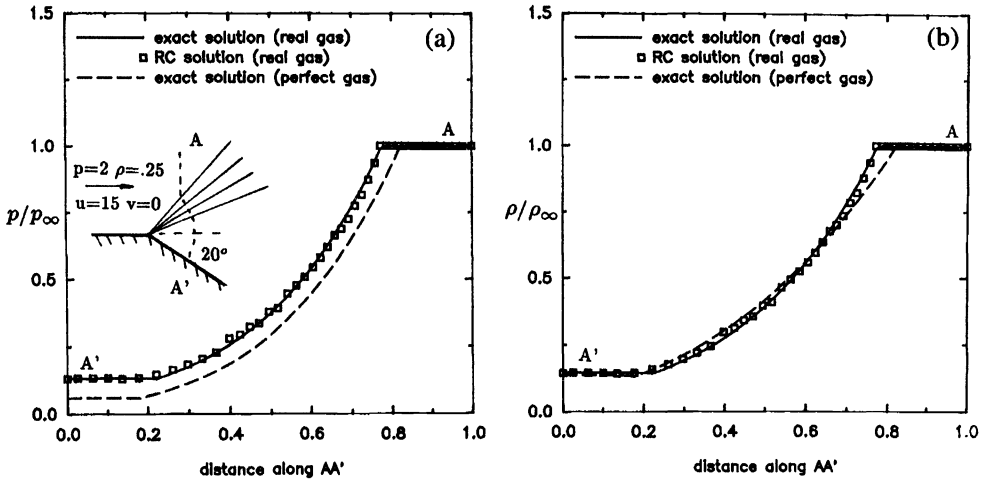


FIG. 8. *Computed flow variables for a real gas* P-M *flow problem with comparison to perfect gas solution*: (a) *pressure and* (b) *density along a typical* λ *line.*

In the fourth example, we investigate oblique shocks and their interaction. First, two shocks are generated on both the upper and lower wall at the inlet of a converging channel. Then the shocks collide with each other and produce two new shocks and a slip line between them. The upper and the lower wall wedge angles are 10° and 20°, respectively. The flow variables of the oncoming free stream are

$$p = 2, \quad \rho = 1, \quad u = 15, \quad v = 0.$$

The same initial data is used for real gas computation and perfect gas computation. In both computations, 80 uniform cells with $h_j = 0.005$ are employed. Figures 9(a), (b), and (c) illustrate the pressure, density, and temperature along a typical λ line after the shock collision in the real gas case ($M = 9.13$), along with the exact solutions (solid lines). The numerical results agree well with the exact ones, the shocks and the slip line are seen to be resolved sharply, except that their precise positions, even though slightly off, cannot be predicted deterministically. Figures 9(d), (e), and (f) are the pressure, density, and temperature contours for the real gas flow. For comparison, the density contours for perfect gas flow with the same initial and boundary conditions are presented in Figure 9(g). It is observed that real gas shock collision takes place slightly later than its perfect gas counterpart.

In our last example, we intend to demonstrate the robustness of the Lagrangian RCM in handling complicated wave interactions. The problem simulates an engine inlet with the upper boundary y_T and the lower boundary y_B defined as

$$y_T = \begin{cases} 0.2 - 0.25x^2, & x < 0.3, \\ 0.1775, & x \geq 0.3, \end{cases}$$

$$y_B = \begin{cases} 0.5x^2, & x < 0.3, \\ 0.045, & x \geq 0.3. \end{cases}$$

The flow variables of the free stream are

$$p = 2, \quad \rho = 1, \quad u = 8, \quad v = 0.$$

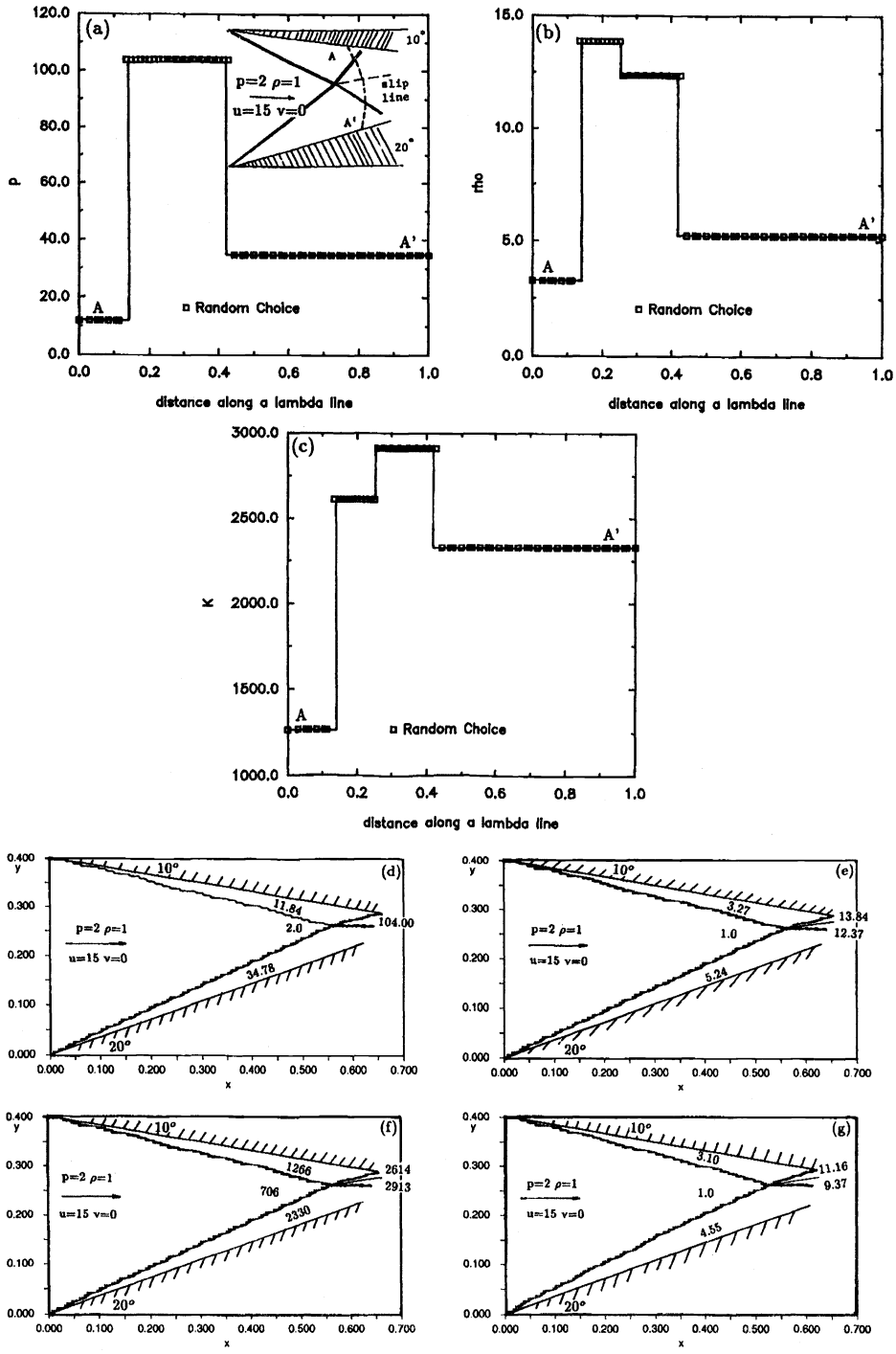


FIG. 9. *Computed results for flow past a converging channel*: (a)–(c) *flow variables along a typical λ line*: (a) *pressure*, (b) *density, and* (c) *temperature*; (d)–(f) *contours*: (d) *isobars*, (e) *isopycnics* (*density contours*), *and* (f) *isotherms. For comparison* (g) *shows the computed isopycnics for perfect gas with identical initial and boundary conditions.*

We use 80 uniform cells with $h = 0.0025$ in the computation. In all the contours of Fig. 10, 100 uniform relative levels are employed. Figures 10(a) and (b) illustrate the isobars and isomachs, respectively. Due to the concave walls in the front, continuous compression waves are first generated and then converge into two floating shocks. These shocks collide with each other and produce two new shocks with a slipline in between. Soon, the two new shocks interact with the Prandtl–Meyer expansion fans issuing from the sudden turn at the walls and are bent slightly. Meanwhile, the two expansion fans, refracted across the shocks, run into and interact with each other. The isobars in Fig. 10(a) clearly show their interaction. At the point where the two floating shocks collide, apart from the new shocks, a slipline (contact discontinuity) is generated and propogates in the flow field. Then, it impinges with the reflected shock from the upper wall and is deflected. The slipline is weak and is barely identifiable from the random fluctuations in the Mach number distributions at Stations A and B (Figs. 10(d) and (f)). However, it is clearly displayed in the isomachs of Fig. 10(b). The two reflected shocks then collide with each other and generate another pair of shocks and a new slipline with vanishing strength and propagate further downstream. Our computation is now terminated at this stage since the method has amply demonstrated its capability of handling complicated interactions between various waves.

In Figs. 10(c) and (d) we show the pressure and Mach number distributions along a typical λ line right after the collision of the two floating shocks. The uniform area around the walls and the expansion fan areas, the shocks are clearly displayed. In Figs. 10(e) and (f) we display, respectively, the pressure and Mach number distributions along another typical λ line at Station B.

Figures 10(g) and (h) illustrate pressure distributions along the walls. The gentle compressions, sudden expansions and shocks are crisply captured.

Before concluding this section, we reiterate that the Lagrangian RCM is valid and robust for flows that are supersonic everywhere. Subject to the solvability of the Riemann problems, the method fails when even a small portion of the flow field falls in the transsonic/ subsonic regime.

5. Concluding remarks. In this paper, we have shown the potential of a Lagrangian random choice method for solving 2-D steady supersonic real gas flows. A key issue is the evaluation of the Lagrangian geometrical quantities $\mathbf{T} = (U, V)^T$. The calculations demonstrated the accuracy of the Lagrangian approach; as in all the random choice methods, the shock position is randomly dispaced, typically by one cell in our results. On the other hand, there is no numerical structure inside the shock, unlike the results from the deterministic approach in which several intermediate points are normally inserted. As compared to the Eulerian RCM (such as [13]), the present method has the advantages in that in each computational cell there are at most five states to choose and that it produces a smooth body surface streamline, rather than a randomly fluctuating one. In other words, the present study suggests that the RCM is more suitable to the solution of the Lagrangian formulation than the Eulerian one. Moreover, since a computational cell is literally a fluid particle in the Lagrangian approach, it requires exchanges of information (fluxes) only with its immediate neighboring particles. This inherent parallelism lends the Lagrangian method naturally to massively parallel computation. Recent results demonstrated significant gain in efficiency for the deterministic Lagrangian approach on the Connection Machine computer (CM-2) [22]. The Lagrangian RCM approach adds another attraction to the parallel computation since no special boundary treatment is needed as in the deterministic approach [23].

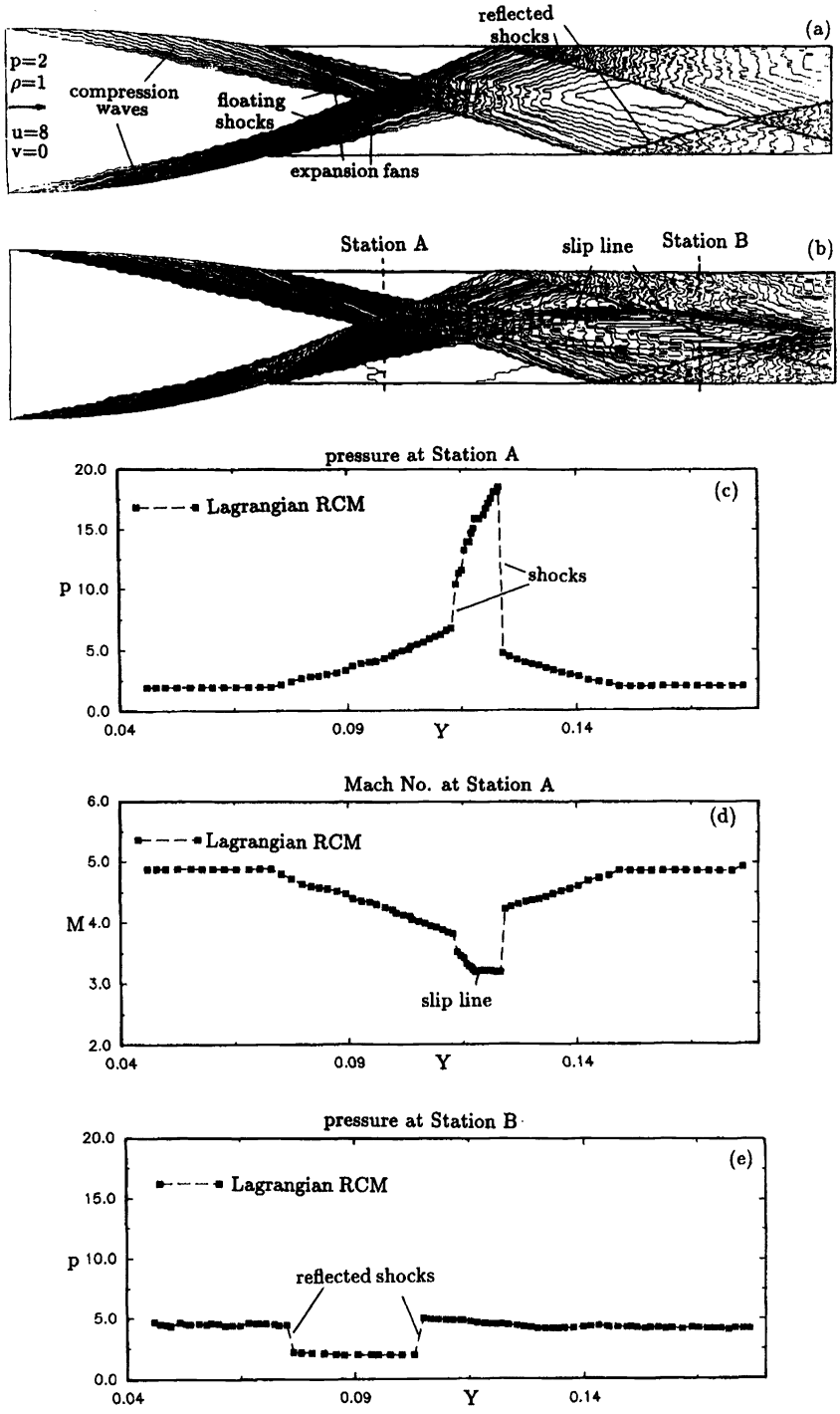


FIG. 10. *Computed results for real gas flow past a channel*: (a) *isobars*; (b) *isomachs*; (c) *pressure*; (d) *Mach number distributions at a tytpical λ line.—Station* A; (e) *pressure.*

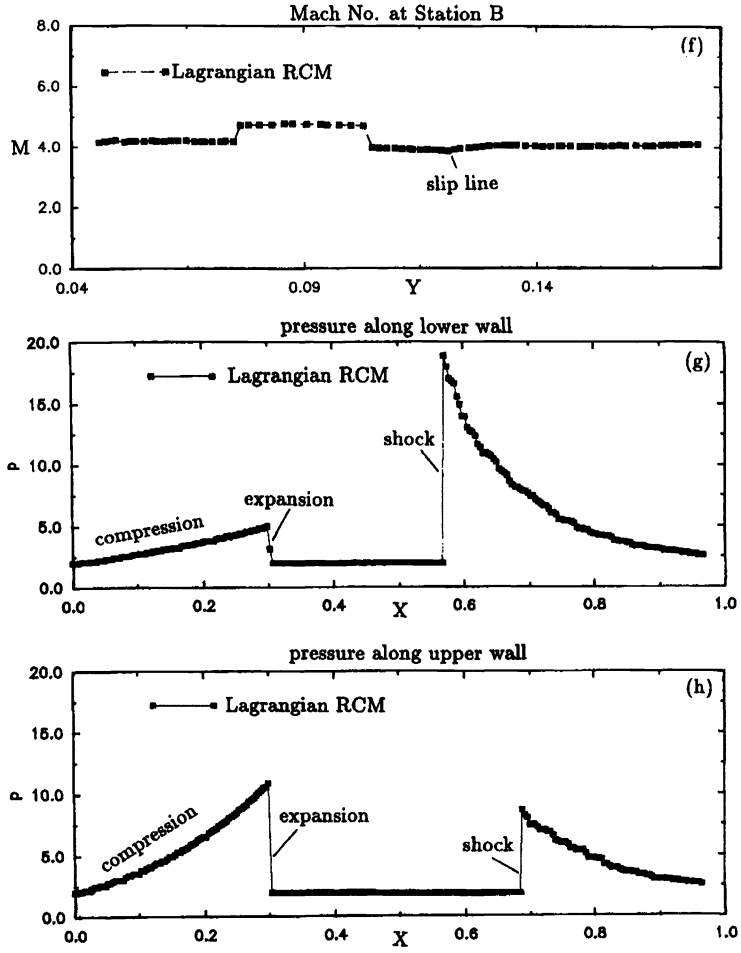


FIG. 10 (*Continued*). *Computed results for real gas flow past a channel*: (f) *Mach number distributions at another typical* λ *line—Station* B; (g) *and* (h) *pressure distributions along lower and upper walls.*

REFERENCES

[1] S. K. GUDONOV, *Difference method of numerical computations of discontinuous solutions in hydrodynamics*, Mat. Sbornik, 47 (1959), pp. 271–306.

[2] S. OSHER AND F. SOLOMON, *Upwind difference schemes for hyperbolic systems of conservation laws*, Math. Comput., 38 (1982), pp. 339–374.

[3] P. L. ROE, *Approximate Riemann solvers, parameter vectors and difference schemes*, J. Comput. Phys., 43 (1981), pp. 357–372.

[4] B. VAN LEER, *On the relation between the upwind differencing schemes of Godunov, Enquist-Osher and Roe*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 1–20.

[5] J. L. STEGER AND R. F. WARMING, *Flux vector splitting of the inviscid gasdynamic equations with application to finite difference methods*, J. Comput. Phys., 40 (1981), pp. 263–293.

[6] J. G. GLIMM, *Solution in the large for nonlinear hyperbolic systems of equations*, Commun. Pure Appl. Math., 18 (1965), pp. 697–715.

[7] A. J. CHORIN, *Random choice solution of hyperbolic systems*, J. Comput. Phys., 22 (1976), pp. 517–536.

[8] P. COLELLA, *Glimm's method for gas Dynamics*, SIAM J. Sci. Statist. Comput., 3 (1982), pp. 76–110.

[9] G. A. SOD, *A numerical study of a cylindrical shock*, J. Fluid Mech., 83 (1977), pp. 785–794.

[10] P. CONCUS AND W. PROSKUROWSKI, *Numerical solution of a nonlinear hyperbolic equation by the random choice method*, J. Comput. Phys., 30 (1979), pp. 153–166.

[11] A. J. CHORIN, *Vortex sheet approximation of boundary layers*, J. Comput. Phys., 27 (1978), pp. 428–442.

[12] J. G. GLIMM, D. MARCHESIN, AND O. MCBRYAN, *A numerical method for two phase flow with an unstable interface*, J. Comput. Phys., 39 (1981), pp. 179–200.

[13] G. MARSHALL AND B. PLOHR, *A random choice method for two-dimensional shock wave diffraction problems*, J. Comput. Phys., 56 (1984), pp. 410–427.

[14] H. OLIVER AND H. GRÖNIG, *The random choice method applied to two-dimensional shock focusing and diffraction problems*, J. Comput. Phys., 63 (1986), pp. 85–106.

[15] C. Y. LOH AND W. H. HUI, *A new Lagrangian random choice method for steady two-dimensional supersonic/hypersonic flow*, AIAA paper 91-1546, American Institute of Aeronautics and Astronautics, New York, 1991.

[16] ———, *A new Lagrangian method for steady supersonic flow computation, part* I: *Godunov scheme*, J. Comput. Phys., 89 (1990), pp. 207–240.

[17] C. Y. LOH AND M.-S. LIOU, *Lagrangian solution of supersonic real gas flow*, J. Comput. Phys., 104 (1993), pp. 150–161.

[18] W. H. HUI AND H. J. VAN ROESSEL, *NATO AGARD symposium on unsteady aerodynamcis–fundamentals and application to aircraft dynamics*, CP-386 (1985), paper S1.

[19] W. H. HUI AND Y. C. ZHAO, *A generalized Lagrangian method for solving the euler equations*, in Proceedings of the Fourth International Conference on Hyperbolic Problems, A. Donato and F. Oliveri, eds., Notes on Numerical Fluid Mechanics, Friedr. Vieweg & Sohn, Wiesbaden 1993.

[20] J. C. TANNEHILL AND P. H. MUGGE, *Improved curve fits for the thermodynamic properties of equilibrium air suitable for numerical computation using time dependent or shock-capturing methods*, NASA CR 2470, 1974.

[21] J. M. HAMMERSLEY AND D. C. HANDSCOMB, *Monte Carlo Methods*, Methuen, London 1964.

[22] M.-F. LIOU AND C. Y. LOH, *Parallel computing using a Lagrangian form*, NASA TM-104446, 1991; in Proc. Parallel Comput. Fluid Dynamics, K. G. Reinsch et al., eds., Stuttgart, Germany, 1991.

[23] W. H. HUI AND C. Y. LOH, *A new Lagrangian method for steady supersonic flow computration, part* II, *slip-line resolution*, J. Comput. Phys., 103 (1993), pp. 450–464.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1059–1063, September 1994

004

REMARK ON ALGORITHMS TO FIND ROOTS OF POLYNOMIALS*

S. GOEDECKER†

Abstract. The problem of finding the roots of a polynomial is equivalent to finding the eigenvalues of an upper Hessenberg matrix, which can be done with the QR algorithm. It is shown that the QR algorithm has considerable advantages over other standard algorithms to find the roots of a polynomial.

Key words. roots of polynomials

AMS subject classifications. 11C++, 65Y25

1. The QR algorithm to find roots of polynomials. It has already been realized in the literature [1] that the problem of finding the roots of the polynomial

$$a_n z^n + a_{n-1} z^{n-1} + \cdots + a_0 = 0 \tag{1}$$

is equivalent to finding the eigenvalues of the upper Hessenberg matrix A

$$\begin{pmatrix} \frac{a_{n-1}}{-a_n} & \frac{a_{n-2}}{-a_n} & \frac{a_{n-3}}{-a_n} & \frac{a_{n-4}}{-a_n} & \cdots & \frac{a_0}{-a_n} \\ 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \cdots & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

If λ is a root of the polynomial (1), λ is an eigenvalue of A and the associated left eigenvector v is given by

$$\begin{pmatrix} \lambda^{n-1} \\ \lambda^{n-2} \\ \lambda^{n-3} \\ \cdots\cdot \\ \lambda^0 \end{pmatrix}.$$

Plugging in the above eigenvector in the eigenvalue equation

$$Av = \lambda v$$

immediately gives (1). This equivalence has been used in [1] to find the roots by a procedure related to the well-known inverse iteration.

Instead of using inverse iteration, the eigenvalues of an upper Hessenberg matrix can of course also be found by the QR algorithm [2].

*Received by the editors October 13, 1992; accepted for publication (in revised form) July 5, 1993. The author was supported in part by a grant from the Swiss National Foundation.

†Ecole Polytechnique Féderale de Lausanne, Institut de Physique Appliquée, CH-1015 Lausanne, Switzerland. Current address: Theory Center, Cornell University, Ithaca, New York 14853-3801 (goedeck@titan.tc.cornell.edu).

2. Test criteria. We compared the QR algorithm, as it is implemented in the Eispack library, with standard root finders of the Numerical Algorithms Group (NAG) and International Mathematics and Statistical Libraries (IMSL) under the following criteria of reliability, accuracy, and speed for both vector and nonvector machines.

• *Reliability.* We tested whether the program can find all the roots without overflow. This is of course not only a characteristic of the algorithm but depends also on the range of floating point numbers that can be represented on the computer. Overflows are a serious problem on VAX computers, where the largest double precision number is of the order of 10^{38}.

• *Accuracy.* The exact roots were not known for most of the test polynomials. To assess the accuracy of the numerical solution we therefore calculated the value of the polynomial at the values of the numerically found roots. In the interval $[-1,1]$ the derivative of the polynomial is bound by $\sum_i i\ |a_i|$. Since this bound is reasonably small in the cases we considered, the value of the polynomial at a numerical root gives an indication of the precision of the numerical root. The situation is however quite different outside the interval $[-1,1]$. The derivative can grow exponentially with the degree of the polynomial and the above procedure cannot be used to estimate the accuracy of the roots. Only one class of test polynomials, namely random polynomials, had several roots outside the interval and we did not check their accuracy. The HQR routine we used did not query the machine constants to optimize accuracy. Implementations of the HQR algorithm, which do this, gave however only a precision which was better by a factor of 2 in the cases we checked.

• *Speed.* The speed was measured by introducing CPU timer calls into the test program. In the case of the NAG and IMSL libraries we made calls to the libraries that were installed on these machines. In the case of the Eispack routine, the Fortran routine was included in the program. Thus exactly the same code was run on all machines. On most machines, implementations of the Eispack routines that are optimized for the architecture are available. Their use would somewhat reduce the timings of the HQR algorithm.

The (double precision) programs we used were the IMSL rootfinder ZPORC [3], which is based on the Jenkins–Traub method [1], [4]; the NAG rootfinder C02AGF [5], which is based on Laguerre method [6]; and the subroutine HQR from the Eispack library [7], which is an implementation of the QR algorithm. The IMSL rootfinder is limited to polynomials of degree less than 100. The test were performed on an IBM RS/6000-550, a Cray2, and a VAX 4000-300. Several classes of polynomials with different properties were tested.

3. Test polynomials.

3.1. Fibocacci polynomials. The name Fibocacci polynomials is not standard terminology, but it is useful to have a name to refer to this class of polynomials. Their coefficients are $a_0 = a_1 = \cdots = a_{n-1} = 1$, $a_n = -1$. The Fibocacci polynomials have n (mostly complex) roots of modulus less than one and one root $\lambda_{\max}$ with modulus greater than one and that has the asymptotic value $\lambda_{\max} = 2 - 2^{-n}$. This distribution of the roots makes them extremely useful for test purposes. The accuracy of the roots in the interval $[-1,1]$ can be assessed by the method discussed above. The values of the Fibocacci polynomial for arguments of modulus less than one can be calculated with nearly machine precision by recursion. In the case of the Fibocacci polynomials, it is however also possible to check the accuracy of the largest root. Since the largest root $\lambda_{\max}$ is well separated from the other roots, it can be calculated with nearly machine precision through the Bernoulli recursion formula. The detailed result are shown in Tables 1 and 2. We separately report the error for the largest root and the largest absolute value $\max(|p(\lambda_i|)$ that the polynomials take on when evaluated at all the other numerical roots. If an entry in the table reads "fail," overflow occurred. "False" means that the errors were so large that they could not be assigned to different roots. For low-order polynomials the HQR method is faster than both the NAG and IMSL library. For high-order polynomials

the NAG library is fastest,the HQR retains the second place, and the IMSL library is slowest on the RS/6000 workstations. On the vector computer Cray2, the HQR is always the fastest. For high-order polynomials only the HQR algorithm gives accurate results.

TABLE 1
Fibocacci polynomials on IBM RS/6000-550.

n	5	10	15	20	30	50	100	150
Time in $m$sec NAG	3.2	7.5	11	15.9	26.5	54.5	120	fail
Time in $m$sec IMSL	2.4	12.8	25.3	88.3	377	1160	3860	fail
Time in $m$sec EISPACK	.40	2.2	4.2	8.9	22.5	72.5	410	1180
$\max(\lvert p(\lambda_i)\rvert)$ NAG	$5\,10^{-16}$	$3\,10^{-14}$	$2\,10^{-14}$	$8\,10^{-15}$	$5\,10^{-14}$	$1\,10^{-12}$	false	fail
$\max(\lvert p(\lambda_i)\rvert)$ IMSL	$3\,10^{-16}$	$6\,10^{-15}$	$3\,10^{-13}$	$1\,10^{-13}$	$7\,10^{-12}$	$5\,10^{-6}$	$8\,10^{-2}$	fail
$\max(\lvert p(\lambda_i)\rvert)$ EISPACK	$2\,10^{-15}$	$7\,10^{-15}$	$1\,10^{-14}$	$6\,10^{-14}$	$2\,10^{-13}$	$2\,10^{-12}$	$3\,10^{-12}$	$1\,10^{-11}$
Error in $\lambda_{\max}$ NAG	$1\,10^{-17}$	$4\,10^{-16}$	$4\,10^{-16}$	$4\,10^{-16}$	$4\,10^{-16}$	$7\,10^{-16}$	false	fail
Error in $\lambda_{\max}$ IMSL	$1\,10^{-17}$	$2\,10^{-16}$	$2\,10^{-16}$	$2\,10^{-14}$	$7\,10^{-16}$	$1\,10^{-17}$	$7\,10^{-15}$	fail
Error in $\lambda_{\max}$ EISPACK	$1\,10^{-15}$	$9\,10^{-16}$	$2\,10^{-16}$	$7\,10^{-16}$	$2\,10^{-15}$	$4\,10^{-16}$	$1\,10^{-15}$	$6\,10^{-15}$

TABLE 2
Fibocacci polynomials on Cray2.

n	5	10	15	20	30	40	50	100
Time in $m$sec NAG	4.7	12	18	25	42	60	fail	fail
Time in $m$sec IMSL	4.3	24	70	98	320	520	940	4560
Time in $m$sec EISPACK	.63	2.5	5.0	8.9	18	36	48	220

3.2. Random polynomials. Random polynomials have roots inside and outside the interval $[-1,1]$. We systematically only checked the accuracy of the roots in this interval. Whenever the root with the largest modulus was well separated we checked it against the Bernoulli recursion. The results were completely analogous to the results we obtained for the Fibocacci polynomials. For low-order polynomials both methods give highly accurate results, but the HQR algorithm is faster. For high-order polynomials the HQR algorithm is somewhat slower on serial machines but always gives high precision results, whereas the precision rapidly degrades with the other methods.

3.3. Legendre polynomials. Legendre polynomials have n real roots in the interval $[-1, 1]$. The values of the polynomial in this interval can be calculated with nearly machine precision by the well-known stable recursion relations. The coefficients of the Legendre polynomials can be calculated with machine precision only for polynomials up to degree 24 (double precision). The results are shown in Table 3. Again the HQR algorithm is the winner because of its speed and precision.

3.4. The polynomials $x^n - 1$. This polynomial is particularly difficult for the HQR method since it requires the application of the exceptional shift to the resulting matrix. In this case all roots have modulus 1 and they can be calculated analytically. Therefore the error in the root itself is listed in Table 4. The NAG library is very fast for this example, but again the HQR algorithm gives the best precision for high-order polynomials and is therefore the winner.

3.5. Multiple roots: Powers of the Fibocacci polynomials. Multiple roots are a numerically more difficult than single roots. Since the slope at the root is zero, the root can be

TABLE 3
Legendre polynomials on IBM RS/6000-550.

n	5	10	15	20	24	50	100
Time in $m$sec NAG	3.0	13.0	22.2	35.0	44.3	78.0	148
Time in $m$sec IMSL	1.3	12.6	23.4	35.2	49.1	330	997
Time in $m$sec EISPACK	0.2	1.5	3.4	6.5	10.1	101	336
max($\|p(\lambda_i)\|$) NAG	$1\ 10^{-15}$	$1\ 10^{-14}$	$3\ 10^{-12}$	$3\ 10^{-10}$	$3\ 10^{-9}$		
max($\|p(\lambda_i)\|$) IMSL	$7\ 10^{-15}$	$2\ 10^{-14}$	$8\ 10^{-12}$	$9\ 10^{-11}$	$6\ 10^{-9}$		
max($\|p(\lambda_i)\|$) EISPACK	$7\ 10^{-15}$	$1\ 10^{-13}$	$2\ 10^{-11}$	$3\ 10^{-10}$	$3\ 10^{-9}$		

TABLE 4
$x^n - 1$ *on* IBM RS/6000-550.

n	5	10	20	50	100
Time in $m$sec NAG	0.8	1.8	4.9	23	48
Time in $m$sec IMSL	5.0	13	40	504	3710
Time in $m$sec EISPACK	0.8	2.6	9.8	65	320
max($err(\lambda_i)$) NAG	$3\ 10^{-16}$	$3\ 10^{-15}$	$1\ 10^{-14}$	$1\ 10^{-14}$	$7\ 10^{-1}$
max($err(\lambda_i)$) IMSL	$2\ 10^{-16}$	$8\ 10^{-15}$	$7\ 10^{-14}$	$2\ 10^{-8}$	$6\ 10^{-1}$
max($err(\lambda_i)$) EISPACK	$5\ 10^{-16}$	$1\ 10^{-15}$	$7\ 10^{-15}$	$1\ 10^{-15}$	$2\ 10^{-13}$

located with much less precision. We generated polynomials that had only double or quadruple roots by taking the Fibocacci polynomials to the second or fourth power. The results are shown in Tables 5 and 6. Both the NAG and IMSL rootfinder showed a rather unpredictable behavior. The precision did not uniformly decay for higher degree polynomials, but showed some unexpected outliers. None of the methods gives acceptable results for polynomials of degree higher than 50.

TABLE 5
Double roots on IBM RS/6000-550.

n	4	8	16	24	32	48
Time in $m$sec NAG	1.0	2.7	7.0	12	18	32
Time in $m$sec IMSL	1.1	6.0	45	140	270	680
Time in $m$sec EISPACK	0.2	2.0	8.0	18	33	77
max($\|p(\lambda_i\|$) NAG	$5\ 10^{-15}$	$8\ 10^{-14}$	$1\ 10^{-5}$	$3\ 10^{-8}$	$2\ 10^{-5}$	.4
max($\|p(\lambda_i\|$) IMSL	$2\ 10^{-14}$	$7\ 10^{-14}$	$7\ 10^{-12}$	$7\ 10^{-9}$	$6\ 10^{-8}$	.5
max($\|p(\lambda_i\|$) EISPACK	$3\ 10^{-15}$	$2\ 10^{-14}$	$5\ 10^{-11}$	$2\ 10^{-10}$	$3\ 10^{-6}$	$5\ 10^{-3}$

4. Test results. Let us summarize the results.

• *Reliability.* For polynomials of high degree overflow occurs frequently in the root finders (especially on VAX), whereas the QR algorithm never gave overflow.

• *Accuracy.* For low-order polynomials, all the subroutines gave highly accurate results. For all high-order polynomials without multiple roots, the rootfinders rapidly lost accuracy, whereas the QR algorithm gave still accurate results. In the case of multiple roots the QR algorithm also gave best overall performance.

• *Speed.* The fact, that the QR algorithm has a complexity of n^3 compared to n^2 for the rootfinders lets one expect that it will be slower for polynomials of high degree. In practice, however, one is usually interested in polynomials of low degree and for these the QR algorithm is faster than the other algorithms. In spite of its higher complexity, the QR algorithm was

TABLE 6
Quadruple roots on IBM RS/6000-550.

n	8	16	24	32	48
Time in $m$sec NAG	2.6	5.8	12	17	27
Time in $m$sec IMSL	100	78	160	325	780
Time in $m$sec EISPACK	1.9	8.1	18	31	78
$\max(\lvert p(\lambda_i)\rvert)$ NAG	$2\,10^{-11}$	$4\,10^{-8}$	$2\,10^{-7}$	$3\,10^{-4}$	10
$\max(\lvert p(\lambda_i)\rvert)$ IMSL	$2\,10^{-7}$	$1\,10^{-14}$	$3\,10^{-8}$	$7\,10^{-9}$	20
$\max(\lvert p(\lambda_i)\rvert)$ EISPACK	$2\,10^{-14}$	$2\,10^{-12}$	$4\,10^{-8}$	$1\,10^{-6}$	$2\,10^{-1}$

not significantly slower than the rootfinders on the IBM RS/6000. On a vector machine like a Cray2, a crossover of the required time due to the cubically increasing number of operations in the QR algorithm cannot be seen since the increase of the length of the vectorized inner loops results in an higher speed. On a vector machine the QR algorithm is therefore significantly faster for any reasonable value of n. The IMSL routine is under all circumstances by far the slowest.

5. Conclusions. It is surprising to realize that the standard libraries for such an old and presumably rather easy problem, namely the determination of polynomial roots, are not optimal both from the point of accuracy and efficiency. A different approach based on the QR algorithm is preferred. The use of the QR algorithm can be recommended without restriction for polynomials of any degree. For low-order polynomials it is faster than rootfinders and for high-order polynomials it gives the correct result without overflow and in most cases with higher precision. If roots of high multiplicity exist, the HQR method as well as any other method have to be used with caution.

Acknowledgement. I thank Dr. K. Maschke for the careful reading of the manuscript.

Note added in proof. The Lapack library, which replaces the Eispack library, is typically two times faster for matrix eigenvalue problems on a workstation. Unfortunately, the Lapack library was not yet available when the tests were done.

REFERENCES

[1] M. A. JENKINS AND J. F. TRAUB, Numer. Math., 14 (1970), p. 252; SIAM J. Numer. Anal., 7 (1970), p. 545.
[2] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1965.
[3] *IMSL User's manual*, version 1.0, (1987), chapter 7.
[4] M. A. JENKINS, ACM Trans. Math. Software, 1 (1975), p. 178.
[5] *NAG Fortran Library Manual*, Mark 13 (1988), vol. 1.
[6] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND W. T. VETTERLING, *Numerical Recipes*, Cambridge University Press, Cambridge, 1989.
[7] *Eispack Guide*, Springer Lecture Notes in Computer Science, Springer, Berlin, 1974.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1064–1082, September 1994

005

THE SCHWARZ ALTERNATING METHOD FOR SINGULARITY PROBLEMS*

ZI-CAI LI†

Abstract. A discrete technique of the Schwarz alternating procedure is presented to combine the Ritz–Galerkin and finite element methods. This technique is suited for solving singularity problems in parallel and requires a little more computation for a large overlap of subdomains. The convergence rate of the iterative procedure, which depends upon overlap of subdomains, has also been studied. Additionally, a balance strategy has been proposed to couple the iteration number with the element size used in the finite element method. For a sample of singularity problems, the total CPU time using the technique in this paper proves to be much less than that by the nonconforming combination.

Key words. Schwarz alternating method, singularity problem, parallel computation, combined method, matching technique, elliptic equation, convergence rate, overlap of subdomain

AMS subject classification. 65N10, 65N30

1. Introduction. Domain decomposition methods have become very attractive due to parallel performance. Of a number of reports [2], [3], [5], [7], [12]–[16] on this subject, only a few, such as [1], [21], are concerned with solution singularity. In this paper, we will study the parallel algorithms for singularity problems by considering a typical problem of singularities [9], [11]:

$$-\Delta u + u = 0 \quad \text{in } S^*, \tag{1.1a}$$

$$u = 1 \quad \text{on } y = 0, \tag{1.1b}$$

$$u = 1 \quad \text{on } x = 0 \cap 0 < y < 1, \tag{1.1c}$$

where $\Delta = (\partial^2/\partial x^2) + (\partial^2/\partial y^2)$, and S^* is the upper semiplane, but excludes the crack section

$$x = 0 \cap 0 < y < 1. \tag{1.1d}$$

Since there exist two kinds of singularities of the problem (1.1), a crack singular point (0, 1) and infinity, the existing domain decomposition methods should be modified for applications to solve (1.1). We have developed some variation and innovation of the substructing iterative methods [2], [3], [5], [7] and the Schwarz alternating procedure [12], [16]. In this paper, we will describe a numerical technique combining the Ritz–Galerkin method and the finite element method to complement the Schwarz alternating procedure.

In the nonconforming combination, the Ritz–Galerkin method and the finite element method are used in the subdomains with and without solution singularity, respectively. In the Schwarz alternating procedure, the subdomains used must be overlapped with each other, and these two methods can be carried out independently and alternatively. In fact, Miller [15] presented numerical analogs using the Ritz–Galerkin and finite difference methods to the Schwarz alternating procedure and provided error bounds of solutions. Recently, Chan, Hou, and Lions [6] reported some convergence results related for L-, T-, and C-shaped domains in the domain decomposition methods. This paper, however, attempts to study the following:

(1) How to evaluate convergence rates of the procedure for the typical problem (1.1), whose subdomains are not rectangles as in [6], but may be circles or sectors.

*Received by the editors November 23, 1992; accepted for publication (in revised form) July 21, 1993. This work was supported in part by research grants received from the Natural Sciences and Engineering Research Council of Canada, the Fond pour la Formation de chercheurs et l'Aide a la Recherche of Quebec by the Ministere de l'Enseignement superieur et de la Science (Action Structurante), and the National Science Council of Republic of China.

†Department of Applied Mathematics, National Sun Yat-Sen University, Kaohsiung, Taiwan 80424, Republic of China (zcli@ibm24.math.nsysu.edu.tw).

(2) How to choose the iteration number in the Schwarz alternating procedure.

We shall first describe, in the next section, a discrete algorithm of the iterative procedures and then provide, in §§3 and 4, analysis and numerical experiments. From these we can find convergence rates of iteration and derive a strategy to match the iteration number with the boundary length of finite elements used.

2. A discrete technique. Based on the symmetry of the geometry of (1.1) we can solve the following problem only on half the region: $S\{(x, y), x > 0 \text{ and } y > 0\}$

$$-\Delta u + u = 0 \quad \text{in } S, \tag{2.1a}$$

$$u = 1 \quad \text{on } y = 0, \tag{2.1b}$$

$$u = 1 \quad \text{on } x = 0 \cap 0 < y < 1, \tag{2.1c}$$

$$\frac{\partial u}{\partial x} = 0 \quad \text{on } x = 0 \cap y > 1. \tag{2.1d}$$

For the separation of singularities, we split the solution domain S into three subdomains S_1, S_2, and S_3, where S_1 is the region $\{(r, \theta), r \le r_1, 0 \le \theta \le \pi\}$ including the crack singular point, $S_2\{(\rho, \phi), \rho \ge \rho_1, 0 \le \phi \le \pi/2\}$ with infinity, and S_3 a bounded subdomain with the interior and exterior boundaries (see Fig. 1):

$$l_{\text{in}} = [(r, \theta), r = r_{\text{in}}, 0 \le \theta \le \pi], \tag{2.2a}$$

$$L_{\text{out}} = [(\rho, \phi), \rho = \rho_{\text{out}}, 0 \le \phi \le \tfrac{\pi}{2}], \tag{2.2b}$$

where the polar coordinates (r, θ) and (ρ, ϕ) have the origins (0, 1) and (0, 0), respectively. The partition of Fig. 1 has overlaps:

$$S_i \cap S_j \neq \varnothing, \qquad i, j = 1, 2, 3. \tag{2.3}$$

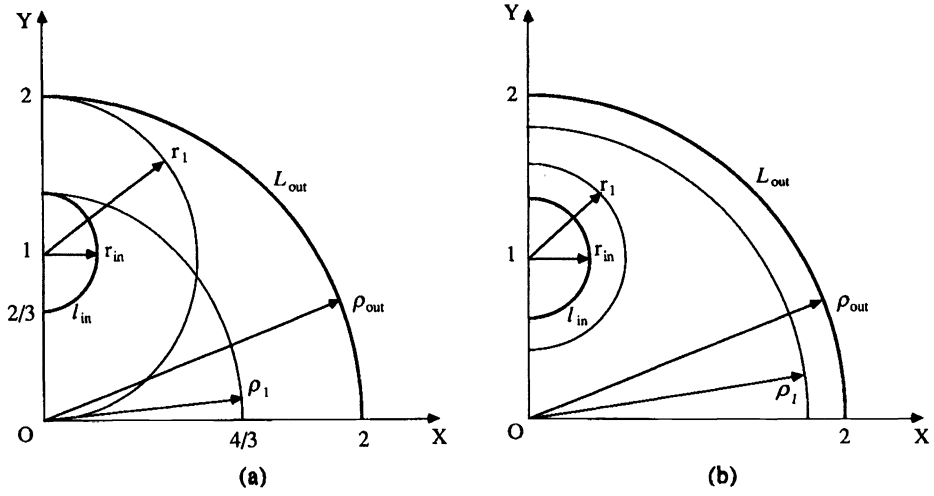


FIG. 1. *Partitions of the infinite domain S into* S_1, S_2, *and* S_3 : (a) *Partition* I; (b) *other partitions.*

Since the solution in S_3 is smooth, such that

$$u \in H^2(S_3), \tag{2.4}$$

where $H^2(S_3)$ is the Sobolev space [4], the finite element method with piecewise linear approximation can be employed to obtain an optimal convergence rate of approximate solutions. Assume that the true solution on l_{in} and L_{out} is known; we may solve the Dirichlet–Neumann problem in S_3 by the finite element method [4].

On the other hand, the true solution in S_1 and S_2 can be expanded as in [9], [11]

$$u = \cosh(r\sin\theta) + \sum_{n=0}^{\infty} a_n I_{n+1/2}(r)\sin\left(\left(n+\frac{1}{2}\right)\theta\right), \qquad r \le \text{Min}(1, \rho_{\text{out}} - 1), \tag{2.5a}$$

$$u = e^{-\rho\sin\phi} + \sum_{n=0}^{\infty} c_n K_{2n+1}(\rho)\sin((2n+1)\phi), \qquad \rho \ge 1 + r_{\text{in}}, \tag{2.5b}$$

where $I_\mu(r)$ and $K_n(\rho)$ are the Bessel and Hankel functions for a purely imaginary argument defined by

$$I_\mu(r) = \sum_{k=0}^{\infty} \frac{1}{\Gamma(k+1)\Gamma(k+\mu+1)}\left(\frac{r}{2}\right)^{2k+\mu}, \tag{2.6a}$$

$$K_n(\rho) = \frac{1}{2}\int_{-\infty}^{\infty} e^{-\rho\cosh\eta - n\eta}\, d\eta. \tag{2.6b}$$

Hence, the following expansions with finite terms are chosen:

$$u_I = u_I(a_n) = \cosh(r\sin\theta) + \sum_{n=0}^{L} a_n \frac{I_{n+1/2}(r)}{I_{n+1/2}(1)}\sin\left(\left(n+\frac{1}{2}\right)\theta\right), \tag{2.7a}$$

$$u_{II} = u_{II}(c_n) = e^{-\rho\sin\phi} + \sum_{n=0}^{LL} c_n \frac{K_{2n+1}(\rho)}{K_{2n+1}(1)}\sin((2n+1)\phi), \tag{2.7b}$$

where $I_{n+1/2}(1)$ and $K_{2n+1}(1)$ in the denominators are used as factors. Approximate values of the coefficients a_n and c_n can be obtained from the Ritz–Galerkin method.

Define the norms on the arcs $l_r\{(r,\theta), r = \text{const and } 0 \le \theta \le \pi)\}$ and $L_\rho\{(\rho,\phi), \rho = \text{const and } 0 \le \phi \le \frac{\pi}{2}\}$ such that

$$|u|_{I,r} = \left[\int_{l_r} u^2 dl\right]^{1/2} = \left[\int_0^{\pi} u^2(r,\theta) r d\theta\right]^{1/2}, \tag{2.8a}$$

$$|u|_{II,\rho} = \left[\int_{L_\rho} u^2 dl\right]^{1/2} = \left[\int_0^{\pi/2} u^2(\rho,\phi)\rho d\phi\right]^{1/2}. \tag{2.8b}$$

To use the Ritz–Galerkin and finite element methods, we need the approximate solutions $\tilde{u}_I$ and $\tilde{u}_{II}$, which can be found by choosing the coefficients a_n and c_n such that

$$|\tilde{u}_I(\tilde{a}_n) - u_h|_{I,r_1} = \underset{\text{all } a_n}{\text{Min}} |u_I(a_n) - u_h|_{I,r_1}, \tag{2.9a}$$

$$|\tilde{u}_{II}(\tilde{c}_n) - u_h|_{II,\rho_1} = \underset{\text{all } c_n}{\text{Min}} |u_{II}(c_n) - u_h|_{II,\rho_1}, \tag{2.9b}$$

where u_h is the finite element approximation obtained.

We summarize below the numerical technique of the Schwarz alternating procedure by the following four steps.

Step 1. Let the initial approximation be

$$\tilde{u}_I^{(0)} = 1 \quad \text{on } l_{\text{in}} \quad \text{and} \quad \tilde{u}_{II}^{(0)} = 1 \quad \text{on } l_{\text{out}} \quad (\text{i.e., } \tilde{u}_h^{(0)} = 1 \text{ in } S_1 \cup S_2). \tag{2.10}$$

Step 2. The finite element approximation $u_h^{(2k+1)} \in V^{(k)}$, $k = 0, 1, 2, \ldots$, is obtained in odd iterations by

$$I(\tilde{u}_h^{(2k+1)}) = \operatorname*{Min}_{u_h \in V^{(k)}} I(u_h) \tag{2.11a}$$

with the energy

$$I(v) = \int\int_{\hat{S}_3^h} \left(\left(\frac{\partial v}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 + v^2 \right) dS, \tag{2.11b}$$

where $\hat{S}_3^h$ is the union of triangular elements of S_3 (see Fig. 2) and $V^{(k)}$ denotes the linear finite element space of the functions satisfying

$$u_h^{(2k+1)} = \tilde{u}_I^{(2k)} \quad \text{on } l_{\text{in}}, \tag{2.12a}$$

$$u_h^{(2k+1)} = \tilde{u}_{II}^{(2k)} \quad \text{on } L_{\text{out}}, \tag{2.12b}$$

$$u_h^{(2k+1)} = 1 \quad \text{on } y = 0 \cap 0 \le x \le \rho_{\text{out}}, \tag{2.12c}$$

$$u_h^{(2k+1)} = 1 \quad \text{on } x = 0 \cap 0 \le y \le 1 - r_{\text{in}}, \tag{2.12d}$$

$$\frac{\partial u_h^{(2k+1)}}{\partial x} = 0 \quad \text{on } x = 0 \cap 1 + r_{\text{in}} \le y \le \rho_{\text{out}}. \tag{2.12e}$$

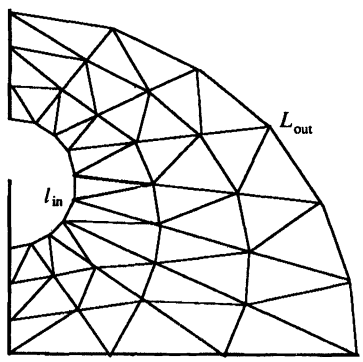


FIG. 2. *Triangulation of S_3 into finite elements.*

Step 3. The solutions $\tilde{u}_I^{(2k+2)}$ and $\tilde{u}_{II}^{(2k+2)}$, $k = 0, 1, \ldots$, as in (2.7) are obtained in even iterations from the Ritz–Galerkin method:

$$|\tilde{u}_I^{(2k+2)} - \tilde{u}_h^{(2k+1)}|_{I,r_1} = \operatorname*{Min}_{\text{all } a_n} |u_I - \tilde{u}_h^{(2k+1)}|_{I,r_1}, \tag{2.13a}$$

$$|\tilde{u}_{II}^{(2k+2)} - \tilde{u}_h^{(2k+1)}|_{II,\rho_1} = \operatorname*{Min}_{\text{all } c_n} |u_{II} - \tilde{u}_h^{(2k+1)}|_{II,\rho_1}. \tag{2.13b}$$

Step 4. Return to Step 2 until the satisfactory solutions $\tilde{u}_h^{(2k+1)}$, $\tilde{u}_I^{(2k+2)}$, and $\tilde{u}_{II}^{(2k+2)}$ have been found.

Step 3, which is, indeed, of the least squares approach, can be easily used for different subdomains S_1 and S_2, and this technique requires only a little more computation even for a large overlap of subdomains (cf. [6], [12], [14], [18]).

Wigley's method [19], [20] may be interpreted as a variation of the Schwarz alternating procedure. For instance, when the discrete approximation with an analytic solution on the entire solution domain is sought, the expansion coefficients $\tilde{a}_n$ and $\tilde{c}_n$ can be evaluated directly from

$$\tilde{a}_n = \frac{2}{\pi}\frac{I_{n+1/2}(1)}{I_{n+1/2}(r)}\int_0^{\pi}[u_h(r,\theta) - \cosh(r\sin\theta)]\sin\left(n+\frac{1}{2}\right)\theta d\theta, \tag{2.14a}$$

$$\tilde{c}_n = \frac{4}{\pi}\frac{K_{2n+1}(1)}{K_{2n+1}(\rho)}\int_0^{\pi/2}[u_h(\rho,\phi) - e^{-\rho\sin\phi}]\sin(2n+1)\phi d\phi. \tag{2.14b}$$

Equations (2.14) are a little simpler than (2.13) in Step 3; but the latter, from the least squares sense, yields smaller errors of numerical solutions.

It is worthwhile to point out that Steps 1–4 can also be applied to problems without singularities. Analytical expansions of u may replace the singular expansions (2.7). Since the Ritz–Galerkin method using particular solutions (or expansions) requires much less CPU time (see §4), the techniques in this paper can also be used for solving in parallel the problems with nonuniform smoothness of solutions.

3. Convergence rates of the iterative procedure. It is well known from Lions [12]–[14] that the Schwarz alternating procedure is convergent in both the maximal norm $|u|_{\infty,S}$ and the energy norm

$$E = \sum_{i=1}^{3}\int\int_{S_i}\left[\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + u^2\right]dS, \tag{3.1}$$

provided that there are some uniform overlaps:

$$S_1 \cap S_3 \neq \varnothing, \qquad S_2 \cap S_3 \neq \varnothing, \tag{3.2}$$

which can be guaranteed by

$$r_1 > r_{\text{in}}, \qquad \rho_1 < \rho_{\text{out}}. \tag{3.3}$$

Combining (3.3) and (2.5) yields the bounds of radii:

$$r_{\text{in}} < r_1 \leq \text{Min}(1, \rho_{\text{out}} - 1), \tag{3.4a}$$

$$1 + r_{\text{in}} \leq \rho_1 < \rho_{\text{out}}. \tag{3.4b}$$

In this section, we focus on the convergence rates of the iterative procedure as Steps 1–4, first to derive a prior rate and then to conduct an empirical rate.

Define the norms

$$|u|_{r,\rho} = \left\{|u|_{I,r}^2 + |u|_{II,\rho}^2\right\}^{1/2}, \tag{3.5a}$$

$$|u|_B = \left\{|u|_{\text{in}}^2 + |u|_{\text{out}}^2\right\}^{1/2}, \tag{3.5b}$$

$$|u|_{\text{in}} = |u|_{I,r_{\text{in}}}, \quad |u|_{\text{out}} = |u|_{II,\rho_{\text{out}}}. \tag{3.5c}$$

The convergence of $|u|_{\infty,S}$ and (3.1) implies the convergence of $|u|_B$, since

$$|u|_B \le C|u|_{\infty,S} \quad \text{and} \quad |u|_B \le CE, \tag{3.6}$$

where C is a bounded constant independent of k.

The L-norm, which is equivalent to (3.5), is defined by

$$[v]_{r,\rho} = \left[[v]^2_{\text{in,r}} + [v]^2_{\text{out},\rho}\right]^{1/2}, \tag{3.7a}$$

where r and ρ are constant, and

$$[v]_{\text{in,r}} = \left[\int_0^{\pi} v^2(r,\theta)\, d\theta\right]^{1/2}, \quad [v]_{\text{out},\rho} = \left[\int_0^{\pi/2} v^2(\rho,\phi)\, d\phi\right]^{1/2}. \tag{3.7b}$$

In fact, $[v]_{\text{in},r}$ is an L-norm of the function $u(r,\theta)$ on $[0,\pi]$. Also denote

$$[v]_B = [v]_{r_{\text{in}},\rho_{\text{out}}} = ([v]^2_{\text{in}} + [v]^2_{\text{out}})^{1/2}, \tag{3.7c}$$

where

$$[v]_{\text{in}} = [v]_{\text{in},r_{\text{in}}}, \qquad [v]_{\text{out}} = [v]_{\text{out},\rho_{\text{out}}}. \tag{3.7d}$$

The limits are

$$\tilde{u}_I = \tilde{u}_I(\tilde{a}_n) = \lim_{k\to\infty} \tilde{u}_I^{(2k)}(\tilde{a}_n^{(2k)}), \tag{3.8a}$$

$$\tilde{u}_{II} = \tilde{u}_{II}(\tilde{c}_n) = \lim_{k\to\infty} \tilde{u}_{II}^{(2k)}(\tilde{c}_n^{(2k)}), \tag{3.8b}$$

as $k \to \infty$, where $u_I(a_n)$ and $u_{II}(c_n)$ are given in (2.7). Let the errors

$$\tilde{\varepsilon}^{(2k)} = \begin{cases} \tilde{u}_I^{(2k)} - \tilde{u}_I & \text{in } S_1, \\ \tilde{u}_{II}^{(2k)} - \tilde{u}_{II} & \text{in } S_2, \end{cases} \tag{3.9}$$

we have

$$[\tilde{\varepsilon}^{(2k)}]_{\text{in},r} = \left[\frac{\pi}{2}\sum_{n=0}^{L}(\tilde{a}_n^{(2k)} - \tilde{a}_n)^2 \frac{I^2_{n+1/2}(r)}{I^2_{n+1/2}(1)}\right]^{1/2}, \tag{3.10a}$$

$$[\tilde{\varepsilon}^{(2k)}]_{\text{out},\rho} = \left[\frac{\pi}{4}\sum_{n=0}^{LL}(\tilde{c}_n^{(2k)} - \tilde{c}_n)^2 \frac{K^2_{2n+1}(\rho)}{K^2_{2n+1}(1)}\right]^{1/2}. \tag{3.10b}$$

Suppose that there exist the positive constants $\alpha_1 < 1$ and $\alpha_2 \le 1$ such that (see Miller [15])

$$[\tilde{u}_I^{(2k)} - \tilde{u}_I]_{\text{in}} \le \alpha_1 [\tilde{u}_I^{(2k)} - \tilde{u}_I]_{\text{in},r_1}, \tag{3.11a}$$

$$[\tilde{u}_{II}^{(2k)} - \tilde{u}_{II}]_{\text{out}} \le \alpha_1 [\tilde{u}_{II}^{(2k)} - \tilde{u}_{II}]_{\text{out},\rho_1}, \tag{3.11b}$$

$$[\tilde{u}^{(2k+2)} - \tilde{u}]_{r_1,\rho_1} \le \alpha_2 [\tilde{u}^{(2k+1)} - \tilde{u}]_B, \tag{3.12}$$

where $\tilde{u} = \tilde{u}_I$ or $\tilde{u}_{II}$. It follows from (2.12a), (2.12b) that

$$[\tilde{\varepsilon}^{(2k)}]_B \le (\alpha_1\alpha_2)[\tilde{\varepsilon}^{(2k-2)}]_B \le (\alpha_1\alpha_2)^k [\tilde{\varepsilon}^{(0)}]_B \to 0 \quad \text{as } k \to \infty. \tag{3.13}$$

Note that the estimates (3.11) and (3.12) result from the solution errors in Steps 2 and 3. We shall prove a prior estimate for α_1 in the following theorem.

THEOREM 1. *Let* (3.4) *hold. There exist the bounds:*

$$[\tilde{u}_I^{(2k)} - \tilde{u}_I]_{\text{in}} \le \left(\frac{r_{\text{in}}}{r_1}\right)^{1/2} [\tilde{u}_I^{(2k)} - \tilde{u}_I]_{\text{in},r_1}, \tag{3.14a}$$

$$[\tilde{u}_{II}^{(2k)} - \tilde{u}_{II}]_{\text{out}} \le e^{-(\rho_{\text{out}}-\rho_1)} [\tilde{u}_{II}^{(2k)} - \tilde{u}_{II}]_{\text{out},\rho_1}. \tag{3.14b}$$

Proof. Since the series terms in (2.6a) and the integrand in (2.6b) are positive, we have

$$I_\mu(r_{\text{in}}) = \sum_{k=0}^{\infty} \frac{\left(\frac{r_{\text{in}}}{r_1}\right)^{2k+\mu} \left(\frac{r_1}{2}\right)^{2k+\mu}}{\Gamma(k+1)\Gamma(k+\mu+1)} \le \left(\frac{r_{\text{in}}}{r_1}\right)^{\mu} I_\mu(r_1), \tag{3.15a}$$

$$K_n(\rho_{\text{out}}) = \frac{1}{2}\int_{-\infty}^{\infty} e^{-\rho_1 \cosh\eta - n\eta} \times e^{-(\rho_{\text{out}}-\rho_1)\cosh\eta} d\eta \le e^{-(\rho_{\text{out}}-\rho_1)} K_n(\rho_1). \tag{3.15b}$$

Then from (3.10)

(3.16a)

$$[\tilde{u}_I^{(2k)} - \tilde{u}_I]_{\text{in}} \le \left[\frac{r_{\text{in}}}{r_1}\frac{\pi}{2}\sum_{n=0}^{L}(\tilde{a}_n^{(2k)} - \tilde{a}_n)^2 \frac{I_{n+1/2}^2(r_1)}{I_{n+1/2}^2(1)}\right]^{1/2} \le \left(\frac{r_{\text{in}}}{r_1}\right)^{1/2} [\tilde{u}_I^{(2k)} - \tilde{u}_I]_{\text{in},r_1},$$

(3.16b)

$$[\tilde{u}_{II}^{(2k)} - \tilde{u}_{II}]_{\text{out}} \le \left[e^{-2(\rho_{\text{out}}-\rho)}\frac{\pi}{4}\sum_{n=0}^{LL}(\tilde{c}_n^{(2k)} - \tilde{c}_n)^2 \frac{K_{2n+1}^2(\rho_1)}{K_{2n+1}^2(1)}\right]^{1/2} \le e^{-(\rho_{\text{out}}-\rho_1)} [\tilde{u}_{II}^{(2k)} - \tilde{u}_{II}]_{\text{out},\rho_1}.$$

This completes the proof of Theorem 1. □

Define the ratio

$$R = \lim_{k\to\infty} [\tilde{\varepsilon}^{(2k)}]_B / [\tilde{\varepsilon}^{(2k+2)}]_B, \tag{3.17}$$

which is the reciprocal of the contraction factor $\alpha_1\alpha_2$ in (3.13). The convergence rate is given by Hageman and Young [8]:

$$\text{convergence rate} = \log R. \tag{3.18}$$

The iterative procedure converges when $R > 1$. The larger the R, the faster the convergence of Steps 1–4. Hence we may choose R to measure convergence of the Schwarz alternative procedure.

For partitioning I shown in Fig. 1(a), we choose

$$r_{\text{in}} = \tfrac{1}{3}, \quad \rho_{\text{out}} = 2, \tag{3.19a}$$

$$r_1 = 1, \quad \rho_1 = \tfrac{4}{3}. \tag{3.19b}$$

Note that in Partition I, there exists also an overlap between S_1 and S_2. Then a prior estimate R^* of R for Partition I follows from Theorem 1 and the property $\alpha_2 \le 1$

$$R^* \le \operatorname{Min}\left\{\left(\frac{r_1}{r_{\text{in}}}\right)^{1/2}, e^{\rho_{\text{out}}-\rho_1}\right\} = 1.73. \tag{3.20}$$

In practice, using the following sequential errors is more convenient since the solution limits (3.8) are often unknown:

$$|\Delta\tilde{\varepsilon}^{(2k)}|_B = [|\Delta\tilde{\varepsilon}_I^{(2k)}|_{\text{in}}^2 + |\Delta\tilde{\varepsilon}_{II}^{(2k)}|_{\text{out}}^2]^{1/2}, \tag{3.21a}$$

where

$$\Delta\tilde{\varepsilon}_I^{(2k)} = \tilde{u}_I^{(2k)} - \tilde{u}_I^{(2k-2)}, \qquad \Delta\tilde{\varepsilon}_{II}^{(2k)} = \tilde{u}_{II}^{(2k)} - \tilde{u}_{II}^{(2k-2)}. \tag{3.21b}$$

Denote M_S and N_S the numbers of finite elements along the radius and radian directions, respectively, then $M_S = 3$, $N_S = 9$ as in Fig. 2. We have carried out the computation through Steps 1–4 for Partition I, computed the errors $|\Delta\tilde{\varepsilon}^{(2k)}|_B$ and $|\tilde{\varepsilon}^{(2k)}|_B$, etc. listed in Table 1, and depicted the error curves in Figs. 3. Note that Fig. 3(a) exhibits a constant ratio:

$$\bar{R} = |\Delta\tilde{\varepsilon}^{(2k)}|_B / |\Delta\tilde{\varepsilon}^{(2k+2)}|_B = 2.8. \tag{3.22}$$

From (3.17) and the bounds

$$|\tilde{\varepsilon}^{(2k)}|_B \le \frac{1}{\bar{R}^{k-1}(\bar{R}-1)} |\Delta\tilde{\varepsilon}^{(2)}|_B \tag{3.23}$$

in [17, p. 194], we may simply regard

$$R \approx \bar{R} = 2.8. \tag{3.24}$$

The empirical value of $R = 2.8$ is much better than $R^* = 1.73$ in (3.20). We are interested in the relation of R with the overlap of subdomains. In fact, Theorem 1 already provides for such a function relation. Furthermore, we will also carry out a trial computation to discover the practical convergence rate.

Now, let us fix the radii as given in (3.19a), and then change r_1 and ρ_1 according to the following two choices:

Choice I. Change S_1 and S_2 by

$$r_1 = r_{\text{in}} + \beta_i, \qquad \rho_1 = \rho_{\text{out}} - \beta_i, \tag{3.25a}$$

where the radii

$$\beta_i = \tfrac{2}{3}, \tfrac{1}{2}, \tfrac{1}{3}, \tfrac{1}{6}, \text{ and } \tfrac{1}{12} \tag{3.25b}$$

is chosen for computation.

Choice II. We match S_1 and S_2 according to a balance of partition I, e.g.,

$$e^{2-\rho_1}/e^{2/3} = \left(\frac{r_1}{r_{\text{in}}}\right)^{1/2} / \sqrt{3}, \tag{3.26a}$$

by means of the prior estimate in Theorem 1. Then the radius

$$\rho_1 = \frac{4}{3} - \frac{1}{2}\ln\left(\frac{1}{3}\frac{r_1}{r_{\text{in}}}\right), \tag{3.26b}$$

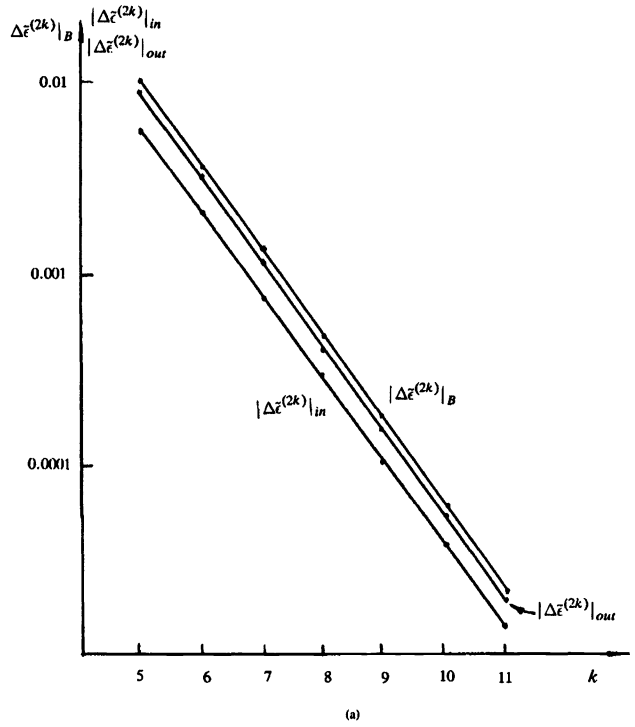


FIG. 3(a). *The straight lines of the sequential errors* $|\Delta\tilde{\varepsilon}^{(2k)}|_{\text{in}}$, $|\Delta\tilde{\varepsilon}^{(2k)}|_{\text{out}}$, *and* $|\Delta\tilde{\varepsilon}^{(2k)}|_B$ *versus* k.

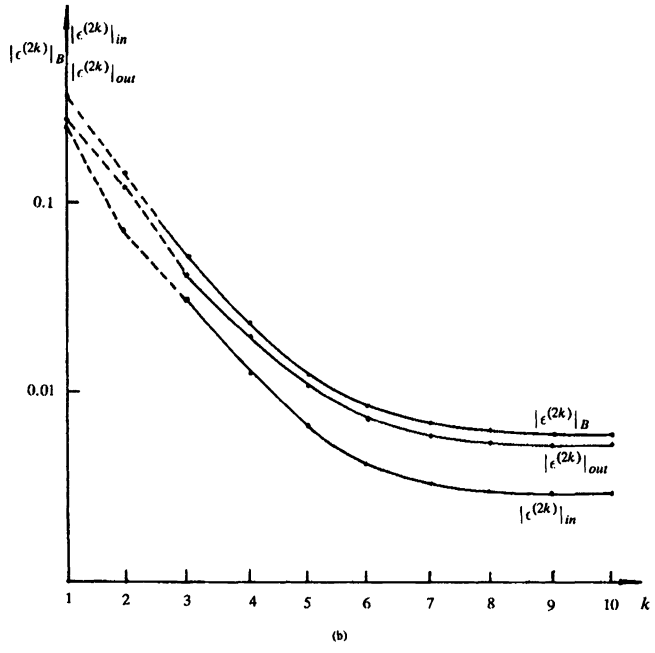


FIG. 3(b). *The curves of the true errors* $|\varepsilon^{(2k)}|_{\text{in}}$, $|\varepsilon^{(2k)}|_{\text{out}}$ *and* $|\varepsilon^{(2k)}|_B$ *versus* k.

TABLE 1
Sequential and true errors in the Schwarz alternating procedure for Partition I *as in Fig.* 1(a).

k	Sequential errors			True errors		
	$\lvert\Delta\tilde{\varepsilon}_I^{(2k)}\rvert_{\text{in}}$	$\lvert\Delta\tilde{\varepsilon}_{II}^{(2k)}\rvert_{\text{out}}$	$\lvert\Delta\tilde{\varepsilon}^{(2k)}\rvert_B$	$\lvert\varepsilon^{(2k)}\rvert_{\text{in}}$	$\lvert\varepsilon^{(2k)}\rvert_{\text{out}}$	$\lvert\varepsilon^{(2k)}\rvert_B$
1	/	/	/	0.249	0.263	0.362
2	0.180	0.142	0.229	0.0695	0.125	0.143
3	0.0369	0.0835	0.0912	0.0327	0.0411	0.0526
4	0.0205	0.0212	0.0295	0.0122	0.0200	0.0234
5	0.00564	0.00982	0.0113	0.00658	0.0102	0.0122
6	0.00248	0.00300	0.00389	0.00415	0.00726	0.00836
7	7.86×10^{-4}	0.00122	0.00145	0.0339	0.00607	0.00696
8	3.11×10^{-4}	4.06×10^{-4}	5.11×10^{-4}	0.00310	0.00568	0.00647
9	1.05×10^{-4}	1.55×10^{-4}	1.87×10^{-4}	0.00300	0.00553	0.00629
10	3.97×10^{-5}	5.38×10^{-5}	6.69×10^{-5}	0.00296	0.00547	0.00622
11	1.39×10^{-5}	1.99×10^{-5}	2.43×10^{-5}	0.00295	0.00545	0.00620
12	5.13×10^{-6}	7.07×10^{-6}	8.73×10^{-6}	0.00294	0.00545	0.00620
13	1.83×10^{-6}	2.58×10^{-6}	3.16×10^{-6}	-	-	-
14	6.66×10^{-7}	9.25×10^{-7}	1.14×10^{-6}	-	-	-
15	2.39×10^{-7}	3.36×10^{-7}	4.12×10^{-7}	-	-	-
16	8.66×10^{-8}	1.21×10^{-7}	1.49×10^{-7}	-	-	-
17	3.12×10^{-8}	4.37×10^{-8}	5.37×10^{-8}	-	-	-
18	1.13×10^{-8}	1.57×10^{-8}	1.94×10^{-8}	-	-	-
19	4.06×10^{-9}	5.69×10^{-9}	6.99×10^{-9}	-	-	-
20	1.47×10^{-9}	2.05×10^{-9}	2.52×10^{-9}	-	-	-

where the following ratios are chosen for computation:

$$\frac{r_1}{r_{\text{in}}} = 3, 2.5, 2, 1.5, 1.1, \text{ and } 1.05. \tag{3.26c}$$

Based on Choices I and II, we have listed the calculated results in Table 2, and drawn curves in Figs. 4 and 5. Partition I* with the same S_3 given in [10] can be viewed as a variation of Partition I by choosing $r_{\text{in}} = \frac{1}{2}$ instead, i.e.,

$$r_{\text{in}} = \tfrac{1}{2}, \quad \rho_{\text{out}} = 2, \quad r_1 = 1, \quad \rho_1 = \tfrac{3}{2}. \tag{3.27}$$

The empirical value of $R = 2.04$, smaller than (3.24), implies a preference for Partition I.

In Fig. 4 the straight lines of $|\Delta\tilde{\varepsilon}^{(2k)}|_B$, in the logarithmic coordinate, versus k indicate an invariance of R. Hence an asymptotic relation for Choice I can be easily found from Fig. 5(a):

$$R = \begin{cases} 1.18e^{1.3\beta}, & \beta \geq \frac{1}{3}, \\ e^{1.78\beta}, & \beta \leq \frac{1}{3}. \end{cases} \tag{3.28}$$

The order $O(e^{1.3\beta})$ or $O(e^{1.78\beta})$ in (3.28) is better than $O(e^{\beta})$ from Theorem 1. The exponential behavior of (3.28) can be interpreted by the fact that the errors from S_2 play a dominant role (see Fig. 3(a), (b)).

For Choice II, we can see from Fig. 5(b)

$$R = \begin{cases} 0.8(r_1/r_{\text{in}}) + 0.4, & r_1/r_{\text{in}} \geq 2, \\ r_1/r_{\text{in}}, & r_1/r_{\text{in}} \leq 2. \end{cases} \tag{3.29}$$

The linear function of R with respect to r_1/r_{in} is also better than the square-root behavior in (3.14a). It is interesting to note that R in (3.29) just equals r_1/r_{in} when r_1/r_{in} is small (≤ 2)!

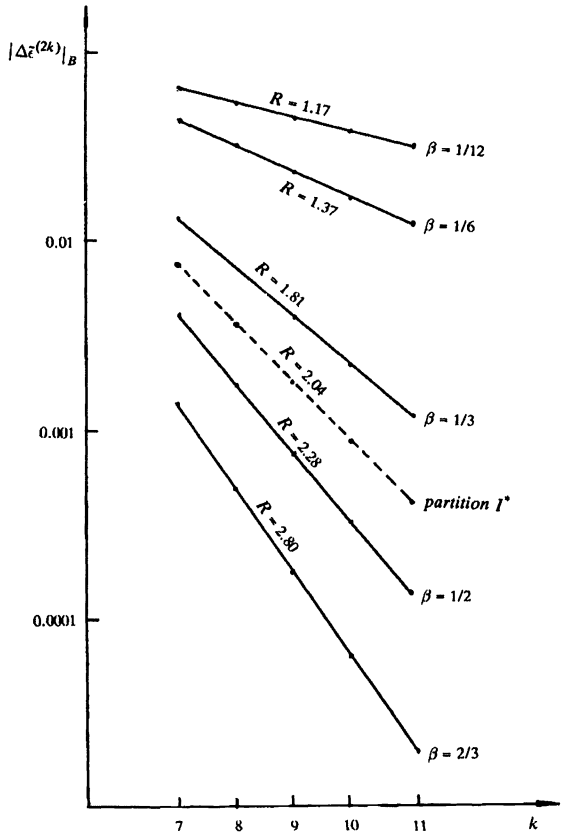


FIG. 4. *The straight lines of the sequential errors* $|\Delta\tilde{\varepsilon}^{(2k)}|_B$ *versus k for different* β.

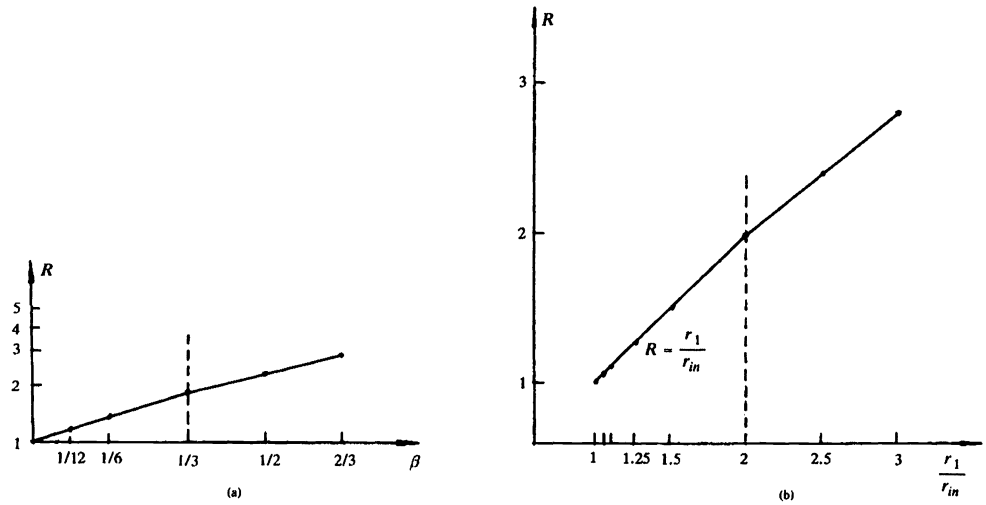


FIG. 5. (a). *The piecewise straight line of* ln R *versus* β *for Choice* I. (b). *The piecewise straight line of* R *versus* r_1/r_{in} *for Choice* II.

TABLE 2(a)
The sequential errors $|\Delta\tilde{\varepsilon}^{(2k)}|_B$, *the true errors* $|\varepsilon^{(2k)}|_B$ *and* R *for different* β *in Choice* I.

β	$\frac{1}{2}$		$\frac{1}{3}$		$\frac{1}{6}$		$\frac{1}{12}$		Partition I*	
Patterns	Partition II		Partition III		Partition IV		Partition V			
	$r_1=\frac{5}{6}, \rho_1=\frac{3}{2}$		$r_1=\frac{2}{3}, \rho_1=\frac{5}{3}$		$r_1=\frac{1}{2}, \rho_1=\frac{11}{6}$		$r_1=\frac{5}{12}, \rho_1=\frac{23}{12}$		$r_1=1, \rho_1=\frac{3}{2}$	
k	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	$\|\varepsilon^{(2k)}\|_B$	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	$\|\varepsilon^{(2k)}\|_B$	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	$\|\varepsilon^{(2k)}\|_B$	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	$\|\varepsilon^{(2k)}\|_B$	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	$\|\varepsilon^{(2k)}\|_B$
1	/	0.464	/	0.627	/	0.876	/	1.04	/	0.565
2	0.259	0.208	0.286	0.344	0.242	0.637	0.156	0.883	0.294	0.281
...	...	...	...	...	...	...	...	...	...	...
7	0.00404	0.0117	0.0134	0.0318	0.0456	0.155	0.0653	0.415	0.00798	0.0128
8	0.00177	0.0100	0.00737	0.0247	0.0331	0.122	0.0553	0.360	0.00386	0.00902
9	7.71×10^{-3}	0.00931	0.00406	0.0208	0.0241	0.0986	0.0468	0.313	0.00190	0.00722
10	3.37×10^{-3}	0.00899	0.00223	0.0187	0.0176	0.0814	0.0397	0.274	9.23×10^{-4}	0.00636
11	1.47×10^{-4}	0.00884	0.00123	0.0175	0.0128	0.0689	0.0337	0.240	4.52×10^{-4}	0.00595
...	...	...	...	...	...	...	...	...	...	...
18	4.45×10^{-6}	0.00874	1.88×10^{-5}	0.0162	0.00141	0.0400	0.0109	0.112	3.00×10^{-6}	0.00557
19	1.94×10^{-7}	0.00874	1.03×10^{-5}	0.0162	0.00103	0.0391	0.00928	0.103	1.47×10^{-6}	0.00556
20	8.49×10^{-8}	0.00874	5.69×10^{-5}	0.0162	0.000752	0.0384	0.00790	0.0954	7.15×10^{-7}	0.00556
R	2.28		1.81		1.37		1.17		2.04	
*	$r_{\text{in}}=\frac{1}{3}, \rho_{\text{out}}=2$								$r_{\text{in}}=\frac{1}{2}, \rho_{\text{out}}=2$	

TABLE 2(b)
The values of R *for different* r_1/r_{in} *in Choice* II.

r_1/r_{in}	3	2.5	2	1.5	1.25	1.1	1.05
r_1	1.0	$\frac{5}{6}$	$\frac{2}{3}$	$\frac{1}{2}$	$\frac{5}{12}$	$\frac{11}{30}$	0.35
ρ_1	$\frac{4}{3}$	1.424494	1.536066	1.679907	1.771068	1.834944	1.858244
R	2.80	2.41	2.01	1.52	1.26	1.11	1.06

In order to discover whether the value of R depends upon $M_S \to \infty$, we increase M_S as well as L, LL correspondingly. It can be found from Table 3 that R does not decrease as $M_S \to \infty$ (even R increases a little bit)!

TABLE 3
The sequential errors $|\Delta\tilde{\varepsilon}^{(2k)}|_B$, *and* R *versus different* M_S *by Steps* 1–4 *for Partition* I.

k				8		14		20	
M_S	N_S	L	LL	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	R	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	R	$\|\Delta\tilde{\varepsilon}^{(2k)}\|_B$	R
2	6	4	2	4.42×10^{-4}	2.86	9.84×10^{-6}	2.78	2.52×10^{-9}	2.77
3	9	5	3	4.37×10^{-4}	2.90	9.81×10^{-6}	2.78	2.52×10^{-9}	2.77
4	12	5	3	4.21×10^{-4}	2.91	9.15×10^{-6}	2.80	2.23×10^{-9}	2.79
6	18	6	4	4.12×10^{-4}	2.93	8.80×10^{-6}	2.81	2.20×10^{-9}	2.80
8	24	6	4	4.05×10^{-4}	2.93	8.52×10^{-6}	2.81	2.05×10^{-9}	2.80
12	36	7	5	4.03×10^{-4}	2.94	8.46×10^{-6}	2.82	2.03×10^{-9}	2.80

4. Error estimates of solutions in Step 2 and matching strategy between iteration number and element size. The following true errors are important in error estimates:

$$|\varepsilon^{(2k)}|_B = \left\{ |\tilde{u}_I^{(2k)} - u_I|_{\text{in}}^2 + |\tilde{u}_{II}^{(2k)} - u_{II}|_{\text{out}}^2 \right\}^{1/2}, \tag{4.1}$$

where u_I and u_{II} are given in (2.7) with the true coefficients a_n and c_n given in [11]. This reminds us of two other errors: $\tilde{\varepsilon}^{(2k)}$ in (3.9) and $\Delta\tilde{\varepsilon}^{(2k)}$ in (3.21).

From the heuristic Fig. 3(b), for saving CPU time the iteration number k should not be too large, but just large enough to balance the errors from the iterative procedure and from the Ritz–Galerkin-finite element method (FEM) methods. In this section, we shall derive error bounds of the obtained solutions in S_3 and then provide a useful matching formula.

Define a continuous solution u^* in S_3 that satisfies (2.1) and

$$u^* = \tilde{u}_I \quad \text{on } l_{\text{in}}, \tag{4.2a}$$

$$u^* = \tilde{u}_{II} \quad \text{on } L_{\text{out}}, \tag{4.2b}$$

where $\tilde{u}_I$ and $\tilde{u}_{II}$ are given in (3.8). Also denote the norm

$$\|v\|_{1/2,B} = (\|v\|^2_{1/2,l_{\text{in}}} + \|v\|^2_{1/2,L\text{out}})^{1/2}, \tag{4.3}$$

where $\|v\|_{1/2,l_{\text{in}}}$ is the Sobolev half-norm, then we have the following lemma.

LEMMA 2. *Let*

$$u \in H^2(S_3), \tag{4.4}$$

then there exists a bounded constant C independent of r_{in}, ρ_{out}, L, LL, *and h such that*

$$\|u^* - u\|_{1/2,B} \le C\left[r_{\text{in}}^{L+3/2} + \left(\frac{1}{\rho_{\text{out}}}\right)^{2LL+3} + h^{3/2}\right]. \tag{4.5}$$

Proof. Let $\hat{u}$ in S_3 satisfy (2.1) and

$$\hat{u} = u_I \quad \text{on } l_{\text{in}}, \tag{4.6a}$$

$$\hat{u} = u_{II} \quad \text{on } L_{\text{out}}, \tag{4.6b}$$

where u_I and u_{II} are given in (2.7) with the true coefficients a_n and c_n. By means of the triangular inequality, we have

$$\|u^* - u\|_{1/2,B} \le \|u^* - \hat{u}\|_{1/2,B} + \|\hat{u} - u\|_{1/2,B}. \tag{4.7}$$

It is easy to obtain from (4.3), (2.5), and (2.7)

$$\|\hat{u}-u\|_{1/2,B} \le C\left\{\left[\sum_{l=L+1}^{\infty} a_l^2(l+1)^2\frac{I^2_{l+1/2}(r_{\text{in}})}{I^2_{l+1/2}(1)}\right]^2 + \left[\sum_{l=LL+1}^{\infty} c_l^2(2l+1)^2\frac{K^2_{2l+1}(\rho_{\text{out}})}{K^2_{2l+1}(1)}\right]^2\right\}^{1/2} \tag{4.8}$$

From (4.4) and [9], [10],

$$a_l = O(\tfrac{1}{l}), \qquad c_l = O(\tfrac{1}{l}), \tag{4.9}$$

$$I_{l+1/2}(r_{\text{in}})/I_{l+1/2}(1) \le C r_{\text{in}}^{l+1/2}, \tag{4.10a}$$

$$K_{2l+1}(\rho_{\text{out}})/K_{2l+1}(1) \le C\left(\frac{1}{\rho_{\text{out}}}\right)^{2l+1}. \tag{4.10b}$$

Therefore, combining (4.8)–(4.10) yields

(4.11)
$$\|\hat{u}-u\|_{1/2,B} = C\left\{\sum_{l=L+1}^{\infty} r_{\text{in}}^{2l+1} + \sum_{l=LL+1}^{\infty}\left(\frac{1}{\rho_{\text{out}}}\right)^{4l+2}\right\}^{1/2} \le C\left\{r_{\text{in}}^{L+3/2} + \left(\frac{1}{\rho_{\text{out}}}\right)^{2LL+3}\right\}.$$

On the other hand, the expansions $\tilde{u}_I$ and $\tilde{u}_{II}$ in (3.8) are computed from the finite element approximation of $\hat{u}$ on l_{in} and L_{out}. Then

$$\|u^* - \hat{u}\|_{1/2,B} \le Ch^{3/2}, \tag{4.12}$$

where h is the maximal boundary length of quasi-uniform elements. The desired result (4.5) is obtained from (4.7), (4.11), and (4.12). □

THEOREM 3. *Let* (4.4) *hold, and suppose that there exists a constant* $R(>1)$ *independent of* h, k, L, *and* LL *such that*

$$|\Delta\tilde{\varepsilon}^{(2k+2)}|_B \le \frac{1}{R}|\Delta\tilde{\varepsilon}^{(2k)}|_B, \tag{4.13}$$

where h *is the maximal length of quasi-uniform elements used in* S_3. *Then the errors of solutions from Step* 2, *the finite element method, have the bounds:*

$$\|\tilde{u}_h^{(2k+1)} - u\|_{1,\hat{S}_3^h} \le C\left\{h + \frac{1}{R^k\sqrt{h}} + r_{\text{in}}^{L+3/2} + \left(\frac{1}{\rho_{\text{out}}}\right)^{2LL+3}\right\}, \tag{4.14}$$

where C *is also a bounded constant independent of* r_{in}, ρ_{out}, h, k, L, *and* LL, *and the Sobolev norm*

$$\|v\|_{1,S} = \left\{\int\int_S \left[\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 + v^2\right] dS\right\}^{1/2}. \tag{4.15}$$

Proof. By means of the triangular inequality again we obtain

$$\|\tilde{u}_h^{(2k+1)} - u\|_{1,\hat{S}_3^h} \le \|\tilde{u}_h^{(2k+1)} - u_h^*\|_{1,\hat{S}_3^h} + \|u_h^* - u^*\|_{1,\hat{S}_3^h} + \|u^* - u\|_{1,\hat{S}_3^h}. \tag{4.16}$$

The second term in (4.16) is bounded from the finite element theory [4]

$$\|u_h^* - u^*\|_{1,\hat{S}_3^h} \le Ch|u^*|_{2,\hat{S}_3^h}. \tag{4.17}$$

We can see from Lemma 2

$$\|u^* - u\|_{1,\hat{S}_3^h} \le C\|u^* - u\|_{1/2,B} \le C\left\{r_{\text{in}}^{L+3/2} + \left(\frac{1}{\rho_{\text{out}}}\right)^{2LL+3} + h^{3/2}\right\}. \tag{4.18}$$

Since $\tilde{u}_h^{(2k+1)}$ and u_h^* are the finite element approximations from different values of solutions on l_{in} and L_{out}, we can provide the bounds of the term based on the extension theory for the finite element method in [3], [7]:

$$\begin{aligned}\|\tilde{u}_h^{(2k+1)} - u_h^*\|_{1,\hat{S}_3^h} &\le C\|\tilde{u}_h^{(2k+1)} - u_h^*\|_{1/2,B} \le \frac{C}{\sqrt{h}}|\tilde{u}^{(2k+1)} - u_h^*|_B \\ &\le \frac{C}{\sqrt{h}}|\tilde{\varepsilon}^{(2k)}|_B \le \frac{C}{R^k\sqrt{h}}|\Delta\varepsilon^{(2)}|_B.\end{aligned} \tag{4.19}$$

The last step of (4.19) follows from (4.13) and (3.23). Consequently, combining (4.16)–(4.19) leads to (4.14). □

In order to grant the optimal convergence rate

$$\|\tilde{u}_h^{(2k+1)} - u\|_{1,\tilde{S}_3^h} = O(h), \tag{4.20}$$

the following relations have to be fulfilled based on Theorem 3:

$$\frac{1}{R^k} = O(h^{3/2}), \tag{4.21}$$

$$r_{\text{in}}^{L+3/2} = O(h), \tag{4.22a}$$

$$\left(\frac{1}{\rho_{\text{out}}}\right)^{2LL+3} = O(h). \tag{4.22b}$$

From (4.22) an optimal matching balancing h and $L_S(= L + 1), LL_S(= LL + 1)$ has been obtained in [9], [10]. Below, an optimal strategy for balancing k and h can be found from (4.21).

COROLLARY 4. *Let all the conditions in Theorem* 3 *hold, and a pair* (k_0, h_0) *be an optimal balance obtained through trial computation. Then any pair* (k_1, h_1) *of* $k_1(\geq k_0)$ *and* $h_1(\leq h_0)$ *satisfying*

$$k_1 = k_0 + \tfrac{3}{2}\ln(h_0/h_1)/\ln R \tag{4.23}$$

is also an optimal balance. In particular, when

$$h_1/h_0 = \tfrac{1}{2}, \qquad R \approx 2.8, \tag{4.24a}$$

then

$$k_1 \approx k_0 + 1. \tag{4.24b}$$

Equations (4.24) are significant since only one more iteration is needed for optimal balance while the boundary length of elements used in S_3 decreases to its half!

From $M_S = 0(\frac{1}{h})$ and the trial computation given in Table 1 and Fig. 3(b), the pair

$$k = 8, \qquad M_S = 3 \tag{4.25a}$$

can be regarded as optimally in balance for Partition I. Since $R(\approx 2.8)$ does not decrease as $h \to 0$ (see Table 3), from Corollary 4 we can predict some new optimal pairs in balance such as:

$$k = 9, \qquad M_S = 6, \tag{4.25b}$$

$$k = 10, \qquad M_S = 12. \tag{4.25c}$$

Based on (4.25), the computed results are given in Table 4, where the norm

$$\|\varepsilon\|_h = [\|\varepsilon\|_{1,S_1}^2 + \|\varepsilon\|_{1,S_2}^2 + \|\varepsilon\|_{1,\hat{S}_3^h}^2]^{1/2}$$

with $\varepsilon = \tilde{u}_h^{(2k+1)} - u$. The curves in Fig. 6(a), (b) drawn from Table 4 exhibit the following asymptotic relations:

$$\|\varepsilon\|_h = O(h), \tag{4.26a}$$

$$\|\varepsilon\|_{0,S} = O(h^2), \tag{4.26b}$$

$$\|\varepsilon\|_{\infty,S} = O(h^{5/3}), \tag{4.26c}$$

$$|\varepsilon|_B = O(h^2). \tag{4.26d}$$

Equation (4.26) perfectly verifies the analysis of the optimal balance, as well as Theorem 3 and Corollary 4. It is remarkable that only ten iterations in Steps 1–4 are required to match the finest triangulation in S_3 with $M_S = 12$.

TABLE 4
Error norms versus different M_S by Steps 1–4 for Partition I.

M_S	N_S	L	LL	k	Max	$\|\varepsilon\|_{0,S}$	$\|\varepsilon\|_h$	$\|\varepsilon\|_{in}$	$\|\varepsilon\|_{out}$	$\|\varepsilon\|_B$
2	6	4	2	8	5.93×10^{-2}	4.22×10^{-2}	0.226	7.75×10^{-3}	1.37×10^{-2}	1.57×10^{-2}
3	9	5	3	8	3.24×10^{-2}	1.91×10^{-2}	0.154	3.10×10^{-3}	5.68×10^{-3}	6.47×10^{-3}
4	12	5	3	9	2.14×10^{-2}	1.09×10^{-2}	1.19×10^{-2}	1.94×10^{-3}	3.08×10^{-3}	3.58×10^{-3}
6	18	6	4	9	1.17×10^{-2}	4.98×10^{-3}	8.05×10^{-2}	8.40×10^{-4}	1.37×10^{-3}	1.61×10^{-3}
8	24	6	4	10	7.22×10^{-3}	2.76×10^{-3}	6.10×10^{-2}	3.38×10^{-4}	6.88×10^{-4}	7.67×10^{-4}
12	36	7	5	10	3.71×10^{-3}	1.31×10^{-3}	4.16×10^{-2}	2.45×10^{-4}	3.59×10^{-4}	4.34×10^{-4}

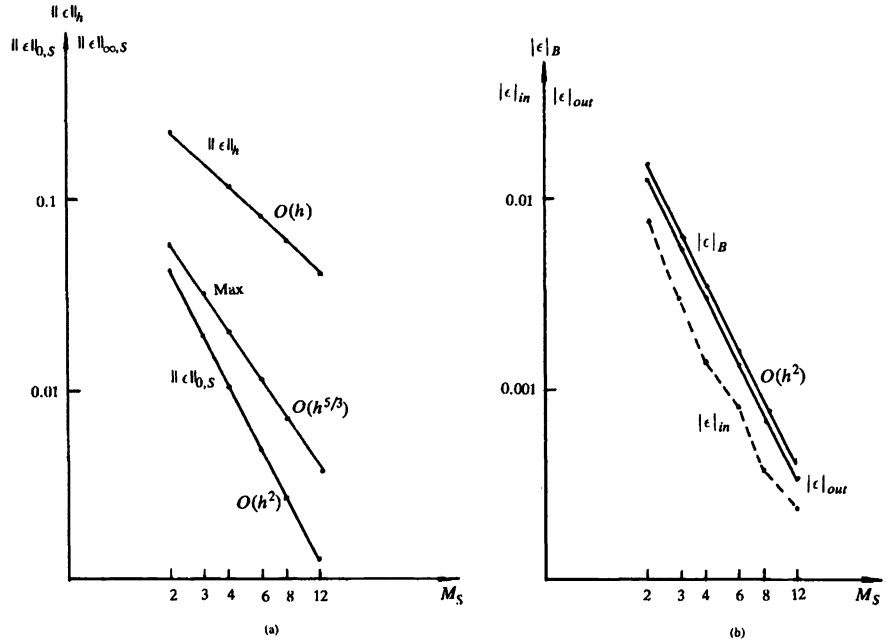


FIG. 6. *The error curves versus M_S : (a) $\|\varepsilon\|_h$, $\|\varepsilon\|_{0,S}$, and $\|\varepsilon\|_{\infty,S}$; (b) $|\varepsilon|_{in}$, $|\varepsilon|_{out}$, and $|\varepsilon|_B$.*

In Tables 5 and 6, error norms and the coefficients from different methods are close to each other. "integ. tech." denotes the algorithm of Steps 1, 2, 4, and (2.14), and "nonconf. comb." the nonconforming combination [9], [10]. The trapezoid rule is employed to approximate the integrals in Step 3 and (2.14).

TABLE 5
Error norms obtained from different methods $M_S = 12$, $N_S = 36$, $L = 7$, $LL = 5$, and $k = 10$ for Partition I.

Methods	Max	$\|\varepsilon\|_{0,S}$	$\|\varepsilon\|_h$	$\|\varepsilon\|_B$	$\|\Delta\tilde{\varepsilon}\|_B$
Steps 1–4	3.71×10^{-3}	1.31×10^{-3}	4.16×10^{-2}	4.34×10^{-4}	6.04×10^{-5}
Integ. tech.	3.73×10^{-3}	1.33×10^{-3}	4.16×10^{-2}	4.35×10^{-4}	6.06×10^{-5}
Nonconf. comb.	3.06×10^{-3}	1.09×10^{-3}	4.15×10^{-2}	/	/

TABLE 6 (a)
The coefficients obtained from different methods with $M_S = 12$, $N_S = 36$, $L = 7$, $LL = 5$, and $k = 10$ for Partition I.

Methods a_n	a_0	a_1	a_2	a_3	a_4	a_5	a_6
Steps 1–4	−1.133658	−0.2032187	0.1804700	0.0499725	0.0128508	0.0080610	0.0044849
Integ. tech.	−1.1337735	−0.2032034	0.1803313	0.0498226	0.0143730	0.0071777	0.0038693
Nonconf. comb.	−1.1340029	−0.2060732	0.1731232	0.0432767	0.0304488	0.0650627	0.1009656
True coeff.	−1.1343218	−0.2035664	0.1802645	0.0498517	0.0144863	0.0073170	0.0040211

TABLE 6 (b)
The coefficients obtained from different methods with $M_S = 12$, $N_S = 36$, $L = 7$, $LL = 5$, and $k = 10$ for Partition I.

Methods c_n	c_0	c_1	c_2	c_3	c_4
Steps 1–4	0.2781283	−0.0905171	0.0450525	−0.0278164	0.0166794
Integ. tech.	0.2780580	−0.0906010	0.0449813	−0.0293295	0.0176094
Nonconf. comb.	0.2773019	−0.0929922	0.0422839	−0.0335831	0.0024471
True coeff.	0.2769005	−0.0906655	0.0456200	−0.0282828	0.0196416

Comparing Steps 1–4 with the nonconforming combination, we find that both the error norms $\|\varepsilon\|_h$ and $\|\varepsilon\|_{0,S}$ are optimal, but the norm $\|\varepsilon\|_{\infty,S}$ has a lower order $O(h^{5/3})$ than $O(h^{2-\delta})$ obtained in [9], [10], where $\delta(> 0)$ is arbitrarily small.

Finally, let us compare the CPU time of Steps 1–4 with that of the nonconforming combination obtained in [9], [10]. Suppose that the Gaussian elimination method is used for solving the algebraic equations obtained. When we order the variables at element nodes along the radius direction, the band width of lower off-diagonals of the stiff matrix in Step 2 is M_S. Since we need the L^T decomposition once, and repeatedly use the forward-backward procedure, the number of arithmetic operations is [17]

$$\frac{1}{2}M_S(M_S+3)N_SM_S^* + k(2N_SM_S^*(M_S+1)),$$

where $M_S^* = M_s - 1$ and $N_SM_S^*$ is the total number of interior nodes of finite elements.

In Step 3, we need approximately

$$k[N_S(L_S + L_{SS}) + L_S^3/3 + LL_S^3/3$$

operations (when using (2.14), only $kN_S(L_S + L_{SS})$ operations are required). Therefore, the total number of arithmetic operations in Steps 1–4 is approximately

$$\text{(4.27)}\quad \tfrac{1}{2}M_S(M_S+3)N_SM_S^* + k[2N_SM_S^*(M_S+1) + N_S(L_S + L_{SS}) + L_S^3/3 + LL_S^3/3].$$

On the other hand, the order of variables at nodes in the nonconforming combination should be arranged along the radian direction because the continuity constraints of admissible functions are enforced along l_{in} and L_{out} between S_3 and S_1, S_2. Hence the bandwidth of the associate matrix is $N_S + 1$, and the number of arithmetic operations is approximately

$$\text{(4.28)} \quad \tfrac{1}{2}(N_S M_S^* + L_S + LL_S)(N_S + 1)(N_S + 4) + 2(N_S + 2)(N_S M_S^* + L_S + L_{SS})$$

by the Gaussian elimination method.

When the optimal matchings (4.21) and (4.22) are used, then

$$\text{(4.29a)} \quad k = O(|\ln h|),$$

$$\text{(4.29b)} \quad L_S = O(|\ln h|), \qquad LL_S = O(|\ln h|).$$

The main parts of (4.27) and (4.28) are reduced to

$$\text{(4.30)} \quad c_0 M_S^2, \qquad c_0 N_S^2$$

as $h \to 0$, where $c_0 = \frac{1}{2} N_S M_S^*$. By noting the fact that $N_S = 3M_S$, we have the following proposition.

PROPOSITION 5. *Let all conditions in Theorem 3 and the optimal matching* (4.21) *and* (4.22) *hold. Suppose that the Gaussian elimination method is used for solving the algebraic equations obtained from the finite element method (as shown in Fig. 2), the CPU time of Steps* 1–4 *in this paper is about* $\frac{1}{9}$ *of that in the nonconforming combination* [9], [10] *as* $h \to 0$.

Proposition 5 displays an advantage of the technique in this paper over [9], [10]. In Step 2, the finite element method in the subdomain S_3 without singularity can be split into parallel from [2], [3], [5], [7]. Also we may use S_3 as a union of rectangles given in Fig. 7 so that the fast Fourier transformation can be applied in Step 2 to significantly save CPU time. It should be noted that the interior corner points A, B, and C in Fig. 7 are not really singular points for the true solution of (2.1). Hence, the traditional domain decomposition methods [2], [3], [5], [7], [12]–[14] can be employed as usual.

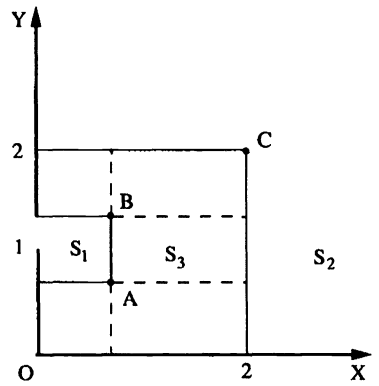


FIG. 7. *A partition of* S_3 *consisting of rectangles.*

REFERENCES

[1] C. R. ANDERSON, *Domain decomposition techniques and the solution of Poisson's equation in infinite domains*, in Domain Decomposition Methods, T. F. Chan et al., eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 129–139.

[2] P. E. BJØRSTAD AND O. B. WIDLUND, *Iterative method for the solution of elliptic problems on the region partitioned into substructures*, SIAM J. Numer. Anal., 33 (1986), pp. 1097–1120.
[3] J. H. BRAMBLE, J. E. PASCIAK, AND A. H. SCHATZ, *The construction of preconditioners for elliptic problems by substructing* I, Math. Comp., 47 (1986), pp. 103–139.
[4] P. G. CIARLET, *The Finite Element Method for Elliptic Problem*, North-Holland, Amsterdam, New York, 1978.
[5] T. F. CHAN, *Analysis of domain decomposition preconditioners*, SIAM J. Numer. Anal., 24 (1987), pp. 382–390.
[6] T. F. CHAN, T. Y. HOU, AND P. L. LIONS, *Geometry related convergence results for domain decomposition algorithms*, SIAM J. Numer. Anal., 28 (1991), pp. 378–391.
[7] M. DRYJA, *A capacity matrix method for Dirichlet problem in polygon region*, Numer. Math., 39 (1982), pp. 51–64.
[8] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, San Diego, 1981.
[9] Z. C. LI, *Numerical Methods for Elliptic Problems with Singularities, Boundary Methods and Nonconforming Combinations*, World Scientific, Singapore, 1990.
[10] ———, *A combined method for solving elliptic problems on unbounded domains*, Comput. Methods Appl. Mech. Engrg., 73 (1989), pp. 191–208.
[11] Z. C. LI AND R. MATHON, *Boundary approximation methods for solving elliptic problems on unbounded domains*, J. Comput. Phys., 89 (1990), pp. 414–431.
[12] P. L. LIONS, *On the Schwarz alternating method* I, in The First Inter. Symposium on Domain Decomposition Method for Partial Differential Equations, R. Glowinski et al., eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988, pp. 1–61.
[13] ———, *On the Schwarz alternating method* II, *Stochastic interpretation and order properties*, in Domain Decomposition Methods, T. F. Chan et al., eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989, pp. 47–70.
[14] ———, *On the Schwarz alternating method* III, *A variant for nonoverlaping subdomains*, in Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, ed. et al., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990, pp. 202–223.
[15] K. MILLER, *Numerical analogs to the Schwarz alternating procedure*, Numer. Math., 7 (1965), pp. 91–103.
[16] H. A. SCHWARZ, *Über einige Abbildungsaufgaben*, Ges. Math. Abh., 11 (1869), pp. 65–83.
[17] H. R. SCHWARZ, *Numerical Analysis, A Comprehensive Introduction*, John Wiley, Chichester, 1989.
[18] W. P. TANG, *Relief from the Pain of Overlap, Generalized Schwarz Splitting*, Tech. Report, Dep. of Computer Science, Waterloo University, 1989.
[19] N. M. WIGLEY, *Stress intensity factors and improved convergence estimates at a corner*, SIAM J. Numer. Anal., 24 (1987), pp. 350–354.
[20] ———, *An efficient method for subtracting off singularities at corners for Laplace's equation*, J. Comput. Phys., 78 (1988), pp. 368–376.
[21] E. G. YANIK, *A Schwarz alternating procedure using spline collection methods*, Inter. J. Numer. Methods Engrg., 28 (1989), pp. 621–627.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1083–1104, September 1994

006

TWO-GRID ITERATION METHODS FOR LINEAR INTEGRAL EQUATIONS OF THE SECOND KIND ON PIECEWISE SMOOTH SURFACES IN $\mathbb{R}^3$*

KENDALL E. ATKINSON†

Abstract. The numerical solution of integral equations of the second kind on surfaces in $\mathbb{R}^3$ often leads to large linear systems that must be solved by iteration. An especially important class of such equations is boundary integral equation (BIE) reformulations of elliptic partial differential equations; and, in this paper BIEs of the second kind are considered for Laplace's equation. The numerical methods used are based on piecewise polynomial isoparametric interpolation over the surface and the surface is also approximated by such interpolation. Two-grid iteration methods are considered for (1) integral equations with a smooth kernel function, (2) BIEs over smooth surfaces, and (3) BIEs over piecewise smooth surfaces. In the last case, standard two-grid iteration does not perform well, and a modified two-grid iteration method is proposed and examined empirically.

Key words. boundary integral equations, iteration methods

AMS subject classifications. primary 65R20; secondary 65N99, 65F10, 35J05

1. Introduction.

The numerical solution of boundary integral equations that are reformulations of partial differential equations in $\mathbb{R}^3$ often leads to the solution of very large systems of linear equations. At present, these linear systems are usually solved by iteration and, traditionally, a geometric series (also called a Neumann series) has been the basis of the iteration scheme. Here we consider two-grid methods for the solution of integral equations of the second kind. Two-grid methods usually converge faster than the geometric series and, in practical terms, they are usually comparable in speed to multigrid methods, while being simpler to program.

As our test case for solving a BIE, we consider the exterior Neumann problem for Laplace's equation:

$$\begin{aligned} &\Delta u(A) = 0, \quad A \in D_e, \\ &\frac{\partial u(P)}{\partial \nu_P} = f(P), \quad P \in S, \\ &u(A) = O(|A|^{-1}), \quad |\nabla U(A)| = O(|A|^{-2}) \quad \text{as } |P| \to \infty. \end{aligned} \tag{1.1}$$

The region $D_e = \mathbb{R}^3 \backslash \bar{D}$, with D an open, bounded, and simply connected region, and S is the boundary of D. The unit normal at $P \in S$, directed into D, is denoted by ν_P. Using Green's third identity,

$$4\pi u(A) = \int_S f(Q)\frac{1}{|A-Q|}dS_Q - \int_S u(Q)\frac{\partial}{\partial \nu_Q}\left[\frac{1}{|A-Q|}\right]dS_Q, \qquad A \in D_e. \tag{1.2}$$

To find u on S, we solve the integral equation

$$\begin{aligned} 2\pi u(P) + \int_S u(Q)\frac{\partial}{\partial \nu_Q}\left[\frac{1}{|P-Q|}\right]dS_Q + [2\pi - \Omega(P)]u(P) \\ = \int_S f(Q)\frac{1}{|P-Q|}dS_Q, \qquad P \in S. \end{aligned} \tag{1.3}$$

*Received by the editors August 6, 1991; accepted for publication (in revised form) July 25, 1993. This work was supported in part by National Science Foundation grant DMS-9003287.

†Department of Mathematics, University of Iowa, Iowa City, Iowa 52242 (atkinson@math.uiowa.edu).

Here, $\Omega(P)$ denotes the inner solid angle of S at $P \in S$. We assume

$$0 < \Omega(P) < 4\pi,$$

which avoids surfaces with cusps.

Symbolically, we write (1.3) as

$$(2\pi + \mathcal{K})u = f. \tag{1.4}$$

Under suitable additional assumptions on S, the operator $\mathcal{K}$ maps $C(S)$ into $C(S)$ and is bounded; see [36]. It can also be shown for such surfaces that (1.4) is uniquely solvable for all $f \in C(S)$, with $(2\pi + \mathcal{K})^{-1}$ a bounded operator; throughout this paper, we assume this is the case. In addition, if S has a parametrization that is differentiable with derivatives that are Hölder continuous with exponent γ, some $\gamma > 0$, then $\mathcal{K}$ is a compact operator; see [26], [18].

The integral equation (1.3) can be solved in a number of ways. If S is a "smooth surface," then in [4], [5], [8], a Galerkin method was presented with approximants based on spherical harmonics. For such problems, this is quite efficient and we recommend that it be considered seriously. Here we consider collocation methods based on triangulating S and approximating u by piecewise quadratic isoparametric interpolation over the triangulation. Such methods were considered previously in [6], [7], [9], and [14]. In [27], Galerkin methods for BIEs of the first kind were considered in a similar context. In the engineering literature such collocation methods have been popular, but usually with only piecewise constant or piecewise linear approximations.

In §2, we review some results on our collocation method that will be needed in later sections. In §3, we give a two-grid iteration method for a general integral equation $(2\pi + \mathcal{K})u = f$, with $\mathcal{K}$ compact and approximated by a numerical integration operator $\mathcal{K}_n$, $n \geq 1$. For BIEs this is too expensive a method, and in §4 we present a variant that follows ideas of Hackbusch, [19], [20, Chap. 16]. Convergence results are given for S smooth, and numerical examples are given. In §5, we extend the iteration method to S, a piecewise smooth surface, discussing the difficulties inherent in this case. The major difficulty traces back to the lack of compactness for $\mathcal{K}$ when S is only piecewise smooth; this requires a modified iteration method. This is analyzed for planar BIEs in Atkinson and Graham [13]; some of these ideas are generalized here to handle BIEs for S a piecewise smooth surface.

2. Preliminaries. We consider the integral equation

$$\lambda u(P) - \int_S K(P, Q)u(Q)dS_Q = g(P), \qquad P \in S \tag{2.1}$$

with the integral operator $\mathcal{K}$ a compact operator from $C(S)$ into $C(S)$. It is assumed that the parameter λ is nonzero and is not an eigenvalue of $\mathcal{K}$; thus $(\lambda - \mathcal{K})^{-1}$ exists as a bounded operator from $C(S)$ onto $C(S)$.

Assume that S can be decomposed as

$$S = S_1 \cup \cdots \cup S_J \tag{2.2}$$

with each S_i the image of a polygonal region R_i in the plane $\mathbb{R}^2$:

$$F_i : R_i \xrightarrow[\text{onto}]{1-1} S_i \qquad i = 1, \ldots, J. \tag{2.3}$$

We assume that $F_i \in C^6(R_i)$. In addition, the subsurfaces S_i are to intersect along only common boundary points. We triangulate each S_i by first triangulating R_i. Then this triangulation

is mapped onto S_i using F_i. Let $\{\hat{\Delta}_{i,k} \mid 1 \le k \le N_i\}$ be a triangulation of R_i and define $\Delta_{i,k} = F_i(\hat{\Delta}_{i,k})$, $1 \le k \le N_i$, $1 \le i \le J$. Collect together the triangulations of each S_i, and refer to them collectively as

$$\mathcal{T}_N = \{\Delta_1, \dots, \Delta_N\}.$$

We assume a number of properties for this triangulation of S. When two sections S_i and S_j have a section Γ of their boundary curves in common, then the triangulations $\{\Delta_{i,k}\}$ and $\{\Delta_{j,l}\}$ must also agree along this common curve Γ. In addition, if S is only a piecewise smooth surface, then (1) the union of the edges of all $\Delta_k \in \mathcal{T}_N$ must contain the union of the edges of S, and (2) the set of all vertices of the $\Delta_k \in \mathcal{T}_N$ must contain all vertices of the original surface S.

For convenience in defining integration and interpolation over the triangulation, we introduce a standard parametrization of each triangle $\Delta_l \in \mathcal{T}_N$. Define the unit simplex:

$$\sigma = \{(s,t) \mid 0 \le s, t, s+t \le 1\}.$$

For $\Delta_{i,k} \subset S_i$, define $m_{i,k} : \sigma \xrightarrow[\text{onto}]{1-1} \Delta_{i,k}$ by

$$m_{i,k}(s,t) = F_i(u\hat{v}_{k,1}^{(i)} + t\hat{v}_{k,2}^{(i)} + s\hat{v}_{k,3}^{(i)}), \qquad (s,t) \in \sigma \tag{2.4}$$

with $u = 1-s-t$ and $\{\hat{v}_{k,1}^{(i)}, \hat{v}_{k,2}^{(i)}, \hat{v}_{k,3}^{(i)}\}$ the vertices of $\hat{\Delta}_{i,k}$. When referring to the triangulation $\Delta_l \in \mathcal{T}_N$, we use $m_l : \sigma \to \Delta_l$ and the vertices of $\hat{\Delta}_l$, Δ_l are denoted by $\{\hat{v}_{l,1}, \hat{v}_{l,2}, \hat{v}_{l,3}\}$ and $\{v_{l,1}, v_{l,2}, v_{l,3}\}$, respectively. It is also necessary to introduce some other node points in Δ_l.

Let $q_1, \dots, q_6$ in σ be defined by

$$q_1 = (0,0), \quad q_2 = (0,1), \quad q_3 = (1,0),$$
$$q_4 = (0,.5), \quad q_5 = (.5,.5), \quad q_6 = (.5,0).$$

For $m_l : \sigma \to \Delta_l$, let $v_{l,j} = m_l(q_j)$, $j = 1, \dots, 6$. The nodes $v_{l,4}, v_{l,5}, v_{l,6}$ are called the "midpoints" of the sides of Δ_l, although they are usually not true midpoints and are used to define quadratic isoparametric interpolation over Δ_l. Collectively, the nodes of the triangulation $\mathcal{T}_N$ are also referred to by

$$\mathcal{V}_N = \{v_1, \dots, v_{Nv}\}.$$

For a surface S that encloses a simple closed region,

$$N_v = 2(N+1). \tag{2.5}$$

Quadratic interpolation. Given $f \in C(\Delta_l)$, for any $1 \le l \le N$, define a quadratic interpolant of $f(m_l(s,t))$ by

$$f(m_l(s,t)) \approx \sum_{i=1}^{6} f(v_{l,i})\ell_i(s,t), \qquad (s,t) \in \sigma \tag{2.6}$$

with $\ell_1, \dots, \ell_6$ the quadratic basis polynomials for which

$$\ell_i(q_j) = \delta_{ij}.$$

For $f \in C(S)$, denote the piecewise quadratic isoparametric interpolation of f, defined by (2.6), by $\mathcal{P}_N f$.

The operator $\mathcal{P}_N$ is a bounded projection on $C(S)$, and its properties were discussed in [6]. In particular,

$$\|\mathcal{P}_N\| = \frac{5}{3} \tag{2.7}$$

and for $f \in C^3(S_i), i = 1, \dots, J$,

$$\|f - \mathcal{P}_N f\|_\infty = O(h^3), \tag{2.8}$$

where

$$h = \operatorname*{Max}_{1 \le l \le N} \text{diameter } (\Delta_l).$$

In dealing with integrals over S, we write

$$\begin{aligned} \int_S g(Q)dS_Q &= \sum_{k=1}^N \int_{\Delta_k} g(Q)dS_Q \\ &= \sum_{k=1}^N \int_\sigma g(m_k(s,t))|D_s m_k \times D_t m_k| ds\, dt \end{aligned} \tag{2.9}$$

with $D_s m_k \equiv (\partial m_k(s,t)/\partial s)$ and similarly for $D_t m_k$. From (2.4), we must have parametrizations of each S_i that are explicitly differentiable. To avoid this requirement, we introduce an approximating surface $\tilde{S}$ to approximate S, and the surface $\tilde{S}$ is used to calculate approximations to the surface differentials in (2.9).

For each $\Delta_k \in \mathcal{T}_N$, define an interpolating mapping $\tilde{m}_k$ by

$$\tilde{m}_k(s,t) = \sum_{i=1}^6 m_k(q_i)\ell_i(s,t) = \sum_{i=1}^6 v_{k,i}\ell_i(s,t). \tag{2.10}$$

By the interpolation error results, and by using the assumed differentiability of m_l,

$$\operatorname*{Max}_{1 \le k \le N} \operatorname*{Max}_{(s,t)\in\sigma} |m_k(s,t) - \tilde{m}_k(s,t)| = O(h^3). \tag{2.11}$$

Denote $\tilde{\Delta}_k = \tilde{m}_k(\sigma)$, and $\tilde{S} = \cup\tilde{\Delta}_k$. The surface S will often be replaced by $\tilde{S}$ in our calculations.

The above discussion has been only for the case of piecewise quadratic interpolation. Polynomial interpolation of other degrees can be used; for example, piecewise linear interpolation of the surface S is very common, along with piecewise constant interpolation for the definition of the collocation method. We have used quadratic interpolation since it illustrates a higher order case than is ordinarily used in practice and has convergence results that are of a higher order than would ordinarily be expected. The results in [14] and [15] can be generalized to handle other degrees of interpolation.

Numerical integration and the Nyström method. To evaluate the surface integrals over S, we use formulas for numerical integration over σ:

$$\int_\sigma g(s,t)ds\, dt \approx \sum_{j=1}^r \omega_j g(s_j, t_j).$$

As an important illustrative case, consider the formula based on integrating the quadratic polynomial that interpolates g at $\{q_1, \ldots, q_6\}$;

$$\int_\sigma g(s,t)ds\,dt \approx \frac{1}{6}\sum_{j=4}^{6} g(q_j). \tag{2.12}$$

This formula has degree of precision two.

When (2.12) is applied to a general triangulation of S, we obtain a composite formula

$$I(G) \equiv \int_S G(Q)\,dS_Q \approx I_N(G) \equiv \sum_{i=1}^{N_v} w_i G(v_i). \tag{2.13}$$

In this case, we have applied (2.12) with $g(s,t) = G(m_k(s,t))|D_S m_k \times D_t m_k|$ for each $\Delta_k \in \mathcal{T}_N$. Actually, this formula involves only the midpoints of the sides of the triangulation, and this means that only $\frac{3}{2}N$ of the weights in (2.13) are nonzero, a savings of about 25% in function evaluations. For the error, we can show that

$$I(G) - I_N(G) = O(h^3). \tag{2.14}$$

This result can be improved when the refinement process for the triangulation is defined in a special way. For a general triangle $\hat{\Delta} = \hat{\Delta}_{i,k} \subset R_i$, we refine it by connecting the midpoints of its sides. Doing this with all elements of the triangulation means that N increases by a factor of four with each refinement of the triangulation. Also, we consider replacing S by $\tilde{S}$ to obtain the integration formula

$$\begin{aligned}\int_S G(Q)\,dS_Q &\approx \sum_{k=1}^{N}\frac{1}{6}\sum_{j=4}^{6} G(v_{k,j})|(D_s\tilde{m}_k \times D_t\tilde{m}_k)(q_j)| \\ &= \tilde{I}_N(G) \equiv \sum_{i=1}^{N_v}\tilde{w}_i G(v_i)\end{aligned} \tag{2.15}$$

with $\tilde{w}_i \equiv \tilde{w}_i^{(N)}$. With the refinement process we have described, it has been shown by Chien [14] that for G sufficiently differentiable on each section S_j of S, the quadrature error satisfies

$$I(G) - \tilde{I}_N(G) = O(h^4). \tag{2.16}$$

We apply (2.15) to the solution of the integral equation (2.1) in the case that the kernel function $K(P, Q)$ is sufficiently differentiable. Nyström's method is used with the numerical integration operator

$$\mathcal{K}_N u(P) = \sum_{i=1}^{N_v}\tilde{w}_i K(P, v_i)u(v_i), \quad P \in S, \quad u \in C(S). \tag{2.17}$$

In (2.1), $\mathcal{K}$ is replaced by $\mathcal{K}_N$, and we solve the approximating equation $(\lambda - \mathcal{K}_N)u_N = g$. This amounts to solving the linear system

$$\lambda u_N(v_i) - \sum_{j=1}^{N_v}\tilde{w}_j K(v_i, v_j)u_N(v_j) = g(v_i), \qquad i = 1, \ldots, N. \tag{2.18}$$

The solution is extended to other points by using the Nyström interpolation formula

$$u_N(P) = \frac{1}{\lambda}\left[g(P) + \sum_{j=1}^{N_v} \tilde{w}_j K(P, v_j) u_N(v_j)\right]. \tag{2.19}$$

A complete theory of Nyström's method is given in [3, Part II, Chap. 3] and we omit it here. Using this theory, and using the results of Chien [14], we obtain

$$\|u - u_N\|_\infty = O(h^4). \tag{2.20}$$

For a further development and numerical illustrations, see [14] and [15].

Collocation methods. The collocation method for solving (2.1) is given by

$$(\lambda - \mathcal{P}_N \mathcal{K}) u_N = \mathcal{P}_N g. \tag{2.21}$$

We write

$$u_N(P) = \sum_{j=1}^{6} u_N(v_{k,j}) \ell_j(s,t), \qquad P = m_k(s,t) \in \Delta_k \tag{2.22}$$

for $1 \le k \le N$. The values $u_N(v_{k,j})$ are obtained by solving the system

(2.23)

$$\lambda u_N(v_i) - \sum_{k=1}^{N}\sum_{j=1}^{6} u_N(v_{k,j}) \int_\sigma K(v_i, \tilde{m}_k(s,t)) \ell_j(s,t) |D_s \tilde{m}_k \times D_t \tilde{m}_k)(s,t)| d\sigma = g(v_i),$$
$$i = 1, \dots, N_v.$$

Note that we are using the approximate surface $\tilde{S}$, so that we should replace $\mathcal{K}$ in (2.21) with an integral operator $\tilde{\mathcal{K}}_N$ based on integration over $\tilde{S}$. Thus we would write

$$(\lambda - \mathcal{P}_N \tilde{\mathcal{K}}_N) u_N = \mathcal{P}_N g. \tag{2.24}$$

For S a smooth surface, we assume that the kernel function $K(P, Q)$ has a smooth extension to points Q in a nearby neighborhood of S; and for S piecewise smooth, we assume that $K(P, Q)$ has a smooth extension to points Q in a neighborhood of each section S_i of the original surface. This assumption can be justified rigorously, but we omit it here. For larger values of N, the calculation of the coefficients in (2.23) is often the most time-consuming part of the collocation method.

For a solvability theory for collocation methods, see [3], and for the method at hand, see [6] and [7]. In [14], it is shown that

$$\operatorname*{Max}_{v \in \mathcal{V}_N} |u(v) - u_N(v)| = O(h^4) \tag{2.25}$$

provided that u is sufficiently smooth. In contrast, the standard theory for collocation methods states that

$$\|u - u_N\|_\infty = O(\|u - \mathcal{P}_N u\|_\infty), \tag{2.26}$$

which from (2.8) is only $O(h^3)$. Thus the node points $v \in \mathcal{V}_N$ are points of superconvergence of the solution u_N.

The integrals in (2.23) are often quite time consuming to evaluate numerically, and hence the Nyström method of (2.18)–(2.19) is often preferable if the kernel function $K(P, Q)$ is

sufficiently smooth. For cases in which $K(P, Q)$ is singular, as with our intended applications to boundary integral equations, the numerical evaluation of the integrals in (2.23) is discussed at length in [9] and [11], and empirical investigations of orders of convergence are given there. Those numerical methods will be used in the applications of §4 and §5.

Numerical example. Consider solving (1.1) with the true solution being

$$u(x, y, z) = \frac{1}{r} e^{x/r^2} \cos\left(\frac{z}{r^2}\right), \qquad r = [x^2 + y^2 + z^2]^{1/2}. \tag{2.27}$$

Let S be the ellipsoidal surface

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 + \left(\frac{z}{c}\right)^2 = 1 \tag{2.28}$$

with $(a, b, c) = (2, 2.5, 3)$. We use the above collocation method (2.24) to solve the boundary integral equation (1.3). For the normal derivative in (1.3), the approximate surface $\tilde{S}$ is used to calculate an approximation to the unit normal:

$$v_Q \approx \tilde{v}_Q \equiv \frac{(D_s \tilde{m}_k \times D_t \tilde{m}_k)(s, t)}{|(D_s \tilde{m}_k \times D_t \tilde{m}_k)(s, t)|}, \qquad Q = m_k(s, t), \quad Q \in \Delta_k. \tag{2.29}$$

Tables 1 and 2 contain numerical results on the error and its empirical rate of convergence to zero. For the column headings, let

$$\text{Error} = E_N \equiv \max_{v \in \mathcal{V}_N} |u(v) - u_N(v)|,$$

$$\text{Order} = p \equiv \log\left[\frac{E_{N/4}}{E_N}\right].$$

At most node points, the order of convergence appears more clearly to be $O(h^4)$; thus it is likely that the values of Order in the tables are approaching four. The theory for this is not yet complete; so far, we have been able to predict an order of only two. (When the true surface S is used, rather than the approximate surface $\tilde{S}$, it is shown in [11] that the maximum error at the node points is $O(h^4 \log(h))$.)

TABLE 1
Solution of (1.3) *using* (2.23).

N	N_v	Error	Order
8	18	1.740E–2	
32	66	2.849E–3	2.61
128	258	2.524E–4	3.50
512	1026	2.004E–5	3.65

TABLE 2
Solution of (1.3) *using* (2.23).

N	N_v	Error	Order
20	42	3.741E–3	
80	120	7.009E–4	2.42
320	480	6.757E–5	3.37

3. The two-grid method for Nyström's method. We consider the iterative solution of the linear system associated with $(\lambda - \mathcal{K}_M)u_M = g$, with $\mathcal{K}_M$ defined as in (2.17). The linear system to be solved is

(3.1)
$$\lambda u_M\left(v_i^{(M)}\right) - \sum_{j=1}^{M_v} \tilde{w}_j^{(M)} K\left(v_i^{(M)}, v_j^{(M)}\right) u_M\left(v_j^{(M)}\right) = g\left(v_i^{(M)}\right), \qquad i = 1, \ldots, M,$$

where the superscript is included to show explicitly the dependence of the weights and nodes on M. If the order M_v is relatively large, say $M_v \geq 1000$, then on most computers this system must be solved iteratively. Let N be less that M, generally much less, and assume that the linear system associated with $(\lambda - \mathcal{K}_N)u_N = g$ can be solved directly. We use the explicit knowledge of $(\lambda - \mathcal{K}_N)^{-1}$ to iteratively solve (3.1).

The following two-grid iteration is discussed in detail in [3, Part 2, Chap. 4], and we merely summarize some of those results for the present case. For the operator equation, the iteration is as follows. We assume that an initial guess $u_M^{(0)}$ is given.

I1 $r^{(l)} = g - (\lambda - \mathcal{K}_M)u_M^{(l)}$,
I2 $p^{(l)} = \mathcal{K}_M r^{(l)}$,
I3 $d^{(l)} = (\lambda - \mathcal{K}_N)^{-1} p^{(l)}$,
I4 $u_M^{(l+1)} = u_M^{(l)} + \frac{1}{\lambda}\left[r^{(l)} + d^{(l)}\right]$.

These relations can be specialized to the linear system (3.1) and the analogous linear system for parameter N; the Nyström interpolation formula (2.19) furnishes the means of connecting the systems of order M_v and N_v.

It can be shown that the iterates satisfy the error equation

(3.2)
$$u_M - u_M^{(l+1)} = \frac{1}{\lambda}(\lambda - \mathcal{K}_N)^{-1}(\mathcal{K}_M - \mathcal{K}_N)\mathcal{K}_M(u_M - u_M^{(l)}).$$

It can be shown that $\{\mathcal{K}_M | M \geq 1\}$ is a collectively compact and pointwise convergent family of operators on $C(S)$; consequently,

(3.3)
$$\operatorname*{Limit}_{N\to\infty} \operatorname*{Sup}_{M\geq N} \|(\mathcal{K}_m - \mathcal{K}_N)\mathcal{K}_M\| = 0.$$

This shows that

(3.4)
$$\left\| \frac{1}{\lambda}(\lambda - \mathcal{K}_N)^{-1}(\mathcal{K}_M - \mathcal{K}_N)\mathcal{K}_M \right\| < 1$$

for all sufficiently large N, uniformly for $M > N$. More can be said about this two-grid method, but we refer the reader to [2] or [3].

Numerical example. Consider the integral equation

(3.5)
$$\lambda u(P) - \int_S u(Q)\frac{\partial}{\partial v_Q}(|P - Q|)^2 dS_Q = g(P), \qquad P \in S$$

with S the ellipsoidal surface (2.34) and $(a, b, c) = (1.0, .75, .5)$. The right-hand function $g(P)$ is chosen so that the true solution is $u(x, y, z) = e^z$. In Table 3, we give iteration convergence rates for various values of N, M, and λ. This rate is defined by

(3.6)
$$\text{Ratio} = \operatorname*{Limit}_{l\to\infty} \frac{\|u_M^{(l)} - u_M^{(l-1)}\|_\infty}{\|u_M^{(l-1)} - u_M^{(l-2)}\|},$$

which we compute empirically. In all cases, such a limit existed computationally. The numerical results illustrate that the value of Ratio should approach a constant as $M \to \infty$,

TABLE 3
Convergence rates for two-grid iteration with system (3.1) *for* (3.5).

λ	N	M	Ratio
30	8	32	.0119
	8	128	.0138
	8	512	.0140
	32	128	.0018
	32	512	.0020
	20	80	.0039
	20	320	.0043
	80	320	.00037
10	8	32	.5006
	8	128	.5817
	8	512	.5889
	32	128	.1616
	32	512	.1760

which can be proven from (3.2). Also note that the values of Ratio decrease substantially when N is increased.

The norm on $C(S)$ of the integral operator $\mathcal{K}$ of (3.5) used in this example is $\|\mathcal{K}\| = 3\pi$. The variation in convergence rates between $\lambda = 30$ and $\lambda = 10$ shows the effect of choosing λ closer to the spectrum of $\mathcal{K}$. As λ approaches the spectrum of $\mathcal{K}$, the size of $\|(\lambda - \mathcal{K})^{-1}\|$ increases in size and thus the linear multiplying factor in the iteration error equation (3.2) also increases in size.

4. Two-grid iteration for the collocation method. We modify the two-grid iteration of §3, in line with ideas in [19] and [20]. In the two-grid iteration I1–I4 of §3, we moved between coarse grid functions and fine grid functions by means of a Nyström interpolation, as in (2.19). Here we use a scheme of prolongation and restriction operators.

The linear systems associated with

$$(\lambda - \mathcal{P}_M \tilde{\mathcal{K}}_M) u_M = \mathcal{P}_M g \tag{4.1}$$

and

$$(\lambda - \mathcal{P}_N \tilde{\mathcal{K}}_N) u_N = \mathcal{P}_N g \tag{4.2}$$

have orders M_v and N_v, respectively, and are associated with triangulations $\mathcal{T}_M = \{\Delta_k^{(M)}\}$ and $\mathcal{T}_N = \{\Delta_k^{(N)}\}$, respectively. We need a way to move between functions defined on the grids $\mathcal{V}_M = \{v_k^{(M)}\}$ and $\mathcal{V}_N = \{v_k^{(N)}\} \subset \mathcal{V}_M$.

Define a *restriction operator*

$$\mathcal{R}_{MN} : \mathbb{R}^{M_v} \to \mathbb{R}^{N_v}$$

by

$$(\mathcal{R}_{MN}\rho_M)\left(v_k^{(N)}\right) = \rho_M\left(v_k^{(N)}\right), \quad 1 \le k \le N_v, \quad \rho_M \in \mathbb{R}^{M_v}. \tag{4.3}$$

In this, ρ_M is regarded as a function on the grid $\mathcal{V}_M$, and $\mathcal{R}_{MN}\rho_M$ is regarded as a function on the grid $\mathcal{V}_N$.

Define a *prolongation operator*

$$\mathcal{P}_{NM} : \mathbb{R}^{N_v} \to \mathbb{R}^{M_v}$$

as follows. Let $\rho_n \in \mathbb{R}^{N_v}$ be a function on the coarse grid $\mathcal{V}_N$, which we wish to extend to a function on $\mathcal{V}_M$. Let $v_{k,j}^{(M)} \in \mathcal{V}_M$. Then $v_{k,j}^{(M)} \in \Delta_k^{(M)}$, and this triangular element is contained in a unique element $\Delta_l^{(N)}$ for some l, $1 \le l \le N$. There is

$$m_l^{(N)} : \sigma \underset{\text{onto}}{\overset{1-1}{\longrightarrow}} \Delta_l^{(N)}.$$

Because $\Delta_k^{(M)} \subset \Delta_l^{(N)}$, the point $v_{k,j}^{(M)} = m_l^{(N)}(\hat{s}, \hat{t})$ for some $(\hat{s}, \hat{t}) \in \sigma$. Define

$$(\mathcal{P}_{NM}\rho_N)\left(v_{k,j}^{(M)}\right) = \sum_{i=1}^{6} \rho_N\left(v_{l,i}^{(N)}\right) \ell_i(\hat{s}, \hat{t}). \tag{4.4}$$

When refining a triangulation, it is best to calculate and save the needed values of $(\hat{s}, \hat{t})$ while doing the refinement.

Write the coarse grid linear system (2.23), associated with (4.2), as

$$(\lambda I - K_N)\mu_N = g_N, \qquad \mu_N \in \mathbb{R}^{N_v}, \tag{4.5}$$

and write the corresponding fine grid linear system associated with (4.1) as

$$(\lambda I - K_M)\mu_M = g_M, \qquad \mu_M \in \mathbb{R}^{M_v}. \tag{4.6}$$

The two-grid iteration for solving the fine grid system is as follows.

J1 $r^{(l)} = g_M - (\lambda I - K_M)u_M^{(l)}$,

J2 $p^{(l)} = \mathcal{R}_{MN}K_M r^{(l)}$,

J3 $d^{(l)} = \mathcal{P}_{NM}(\lambda I - K_N)^{-1}p^{(l)}$,

J4 $\mu_M^{(l+1)} = \mu_M^{(l)} + \frac{1}{\lambda}\left[r^{(l)} + d^{(l)}\right]$.

An equivalent form can be given that reminds one of multigrid iteration.

$\hat{\text{J}}1$ $\hat{\mu}_M^{(l)} = \frac{1}{\lambda}[g_M + K_M\mu_M^{(l)}]$,

$\hat{\text{J}}2$ $\hat{r}^{(l)} = g_M - (\lambda I - K_M)\hat{\mu}_M^{(l)}$,

$\hat{\text{J}}3$ $\mu_M^{(l+1)} = \hat{\mu}_M^{(l)} + \mathcal{P}_{NM}(\lambda I - K_N)^{-1}\mathcal{R}_{MN}\hat{r}^{(l)}$.

Step $\hat{\text{J}}1$ is called the "smoothing step" and step $\hat{\text{J}}3$ is called the "correction step." The replacement of Nyström interpolation, as in (2.19), with the quadratic interpolation of (4.4) yields a less expensive way to move from a coarse grid function to a fine grid function following ideas in [19].

For the iteration error, it is straightforward to derive

$$\mu_M - \mu_M^{(l+1)} = C_{N,M}\left[\mu_M - \mu_M^{(l)}\right], \tag{4.7}$$

$$C_{N,M} = \frac{1}{\lambda}\left[I - \mathcal{P}_{NM}(\lambda I - K_N)^{-1}\mathcal{R}_{MN}(\lambda I - K_M)\right]K_M. \tag{4.8}$$

To have convergence of the iteration, one must show that

$$\|C_{N,M}\| < 1. \tag{4.9}$$

We consider here only the type of integral operators arising in boundary integral equations as in (1.3).

If S is a smooth surface, then the integral operator $\mathcal{K}$ of (1.3) is compact on $C(S)$ into $C(S)$; the same is true of other integral operators of potential theory with a weakly singular kernel. From this, we can use the tools associated with the theory of collectively compact operators to show that

$$\lim_{N\to\infty} \sup_{M>N} \|C_{N,M}\| = 0. \tag{4.10}$$

The proof is given in the Appendix of this paper.

Numerical example. Consider the BIE (1.3) with the known true solution (2.27), and use the same ellipsoidal surface S of (2.28). We consider the iterative solution of the linear systems for this earlier example. To save time in carrying out the study of the iteration method, we did not evaluate the collocation integrals

$$\int_\sigma K(v_i, \tilde{m}_k(s,t))\ell_j(s,t)|(D_s\tilde{m}_k \times D_t\tilde{m}_k)(s,t)|d\sigma$$

of (2.23) as accurately as had been done in obtaining the values in Tables 1 and 2. For this section, the above integrals were divided into two classes. For $v_i \in \Delta_k$, a change of variables was performed to eliminate the singularity and then a low-order product Gaussian quadrature formula was used. For $v_i \in \Delta_k$, the integration was performed using formula T2:5–1 of Stroud [35, p. 314]. This is a seven-point formula with degree of precision five. Empirically, we have found that there is very little difference in the performance of the iteration methods for the system obtained with the quadrature just described and with the linear system (2.23) using very accurate evaluation of all collocation integrals. For a further discussion of the evaluation of the collocation integrals, see [7], [9], [11], [34], and [17].

Table 4 contains the empirical iteration rates of (3.6). Note again that the rate of convergence improves as the coarse mesh parameter N is increased. This agrees with (4.10). For one case (indicated by †), the empirical limit in (2.41) did not occur, and we used a geometric average of the final few ratios to obtain the value given in the table. We also see in the table that for fixed N, the convergence ratios appear to be approximately constant as M increases.

TABLE 4
Iteration rates for solving BIE (1.3) *on an ellipsoid.*

(a, b, c)	N	N_v	M	M_v	Ratio
(2, 2.5, 3)	8	18	32	66	.17
	8	18	128	258	.14
	8	18	512	1026	.14
	32	66	128	258	.089
	32	66	512	1026	.072
	20	42	80	162	.11
	20	42	320	642	.089
	80	162	320	642	.062
(1, 2, 3)	8	18	32	66	.31†
	8	18	128	258	.25
	8	18	512	1026	.27
	32	66	128	258	.19
	32	66	512	1026	.18

Further discussion. In [21] and [31] methods are given for the rapid (approximate) evaluation of the matrix-vector product $K_N\rho$, $\rho \in \mathbb{R}^{N_v}$. The cost of straightforward evaluation

is approximately $2N_v^2$ arithmetic operations. The cost in [21] is $O(N_v \log^4 N_v)$ arithmetic operations; the method in [31] is similar in cost. Since the evaluation of $K_N\rho$ is a major part of the cost, and since the preceding authors also give a less expensive way to evaluate the matrix K_N, their methods can also be used with the present two-grid method to further reduce its cost. Alternatively, we can consider the use of vector/parallel processors in evaluating $K_N\rho$, which will also reduce the cost of the iteration. Regardless of how $K_N\rho$ is evaluated, the discussion of the two-grid iteration presented here remains valid as it is independent of the method of evaluating $K_N\rho$. For the case that S is a smooth surface, the two-grid iteration of this section seems to be an easy-to-use method with good convergence.

5. Iteration for BIEs on piecewise smooth surfaces. When the surface S is only piecewise smooth, the double-layer integral operator $\mathcal{K}$ of (1.3) is not compact on $C(S)$, although it is still bounded. Consequently, the standard theory of collectively compact operator approximations ([1]) does not apply in this case, and a convergence theory must be developed in some other way. For the limited convergence theory of collocation methods that is available for (1.3) when S is only piecewise smooth, see [36], [37], [23], [6], [7], [9], and [11].

The given theories depend on dividing the surface into two subsets, one subset of points "close" to edges and corners of S and the other subset the remaining part of S. The integral equation (1.4) is written as

$$2\pi \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} \mathcal{K}_{11} & \mathcal{K}_{12} \\ \mathcal{K}_{21} & \mathcal{K}_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}. \tag{5.1}$$

In this, all functions $u \in C(S)$ are decomposed as

$$u_1 = u|_{S_1}, \qquad u_2 = u|_{S_2} \tag{5.2}$$

and for the function space, we use $\mathcal{X} = C(S_1) \oplus C(S_2)$. Following this decomposition, the integral operators $\mathcal{K}_{ij}$ are defined in a natural way from $\mathcal{K}$, using the decomposition of S, and

$$\mathcal{K}_{ij} : C(S_j) \to C(S_i).$$

For the solvability theory for (1.3) and the convergence theory for numerical methods for solving (1.3), the invertibility of $2\pi + \mathcal{K}_{11}$ and its numerical approximants is established first. After doing this, the remaining part of the problem can be attacked using standard tools based on compactness, since $\mathcal{K}_{ij}$ is compact for $(i, j) \neq (1, 1)$. This approach has been used for planar BIEs on regions with a piecewise smooth boundary curve and is still the main way of approaching analogous equations in three dimensions. This approach to the problem is due originally to Radon [30] for planar problems, and it was extended to three-dimensional BIEs and their numerical solution by Wendland [36].

For the collocation method proposed in (2.24) in §2, we assume that the surface S satisfies the same properties as those assumed in [36]. With these properties, we obtain convergence of the numerical method (2.26) if

$$\frac{5}{3}|2\pi - \Omega(P)| < 1 \tag{5.3}$$

at all points $P \in S$ [6], [7]. The assumption (5.3) seems to be an artificial one based on the method of analysis being used. Empirically, (5.3) appears to be unnecessary.

In addition to the problems of the invertibility of $2\pi + \mathcal{K}$, there is also the problem of dealing with the behaviour of the solution $u(P)$ for points P near to edges and corners of S.

It is known from the planar theory that the function $u(P)$ usually has algebraic singularities that must be considered in the numerical solution of the equation. For a development of ideas in this direction for (1.3), see [16] and [29].

The two-grid iteration method. Because $\mathcal{K}$ is not compact, the condition (4.10) is not true and there is no reason to believe that the convergence requirement (4.9) is true. We have also studied the analogous iteration for planar BIEs in [13], and when the method converged, the rate of convergence did not depend on the size of N. We have investigated empirically the two-grid iteration J1–J4 and we give some of the results of that study here.

Let S be the elliptical paraboloid

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = z, \qquad 0 \le z \le c \tag{5.4a}$$

together with the "cap" of points (x, y, z) satisfying

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 \le c, \qquad z = c. \tag{5.4b}$$

We again solved (1.3) and the true solution was chosen to be

$$u(x, y, z) = [(x - \tfrac{1}{2})^2 + y^2 + z^2]^{-1/2}.$$

Results for the two-grid iteration method are given in Table 5.

The first observation is that the two-grid iteration still works, provided that N is sufficiently large. Second, the rate of convergence does not improve as N increases (provided that N is large enough to have Ratio < 1), contrary to the case with a smooth surface.

TABLE 5
Two-grid iteration for (1.3) *on a paraboloid.*

(a, b, c)	N	M	Ratio
(.5, .5, 1)	8	32	.615
	8	128	.592
	8	512	.586
	32	128	.594
	32	512	.586
(1, 1, 1)	8	32	.668
	8	128	.658
	8	512	.652
(2, 2, 1)	8	32	.761
	8	128	.758
	8	512	.754
	32	128	.757
	32	512	.754
(2.5, 2.5, 1)	8	32	89.326
	32	128	.790†
	32	512	.862

For the paraboloid surface of (5.4) with $a = b$, the interior solid angle on the edge of S at $z = c$ is

$$\Omega_a = 2\tan^{-1}\left(\frac{2\sqrt{c}}{a}\right). \tag{5.5}$$

Using Table 5 results and this formula, we find that for the convergence ratio $r_a \equiv$ Ratio, most of the values of r_a satisfy

$$r_a \doteq 1 - .157\Omega_a. \tag{5.6}$$

This makes some sense since

$$1 - .157\Omega_a \doteq 0 \Rightarrow \Omega_a \doteq 2\pi,$$

and that would correspond to a smooth surface. With a smooth surface, the earlier result (4.10) implies that the value of Ratio converges to zero as $N \to \infty$.

A modified two-grid iteration. The split (5.1) is required for not only a convergence analysis of the numerical solution of $(2\pi + \mathcal{K})u = g$, but also seems to be necessary when iteratively solving the linear system $(2\pi + K_M)\mu_M = g_M$. This has been noted previously in [32], [33], [23], and [13]. We use this split to obtain a modified linear system for which two-grid iteration performs better.

Based on the split (5.1), we make an analogous splitting of the domain of the grid function μ_M. Then $(2\pi + K_M)\mu_M = g_M$ is rewritten as

$$2\pi \begin{bmatrix} \mu_M^1 \\ \mu_M^2 \end{bmatrix} + \begin{bmatrix} K_M^{1,1} & K_M^{1,2} \\ K_M^{2,1} & K_M^{2,2} \end{bmatrix} \begin{bmatrix} \mu_M^1 \\ \mu_M^2 \end{bmatrix} = \begin{bmatrix} g_M^1 \\ g_M^2 \end{bmatrix}. \tag{5.7}$$

We assume that $K_M^{1,1}$ has been so constructed that $2\pi + K_M^{1,1}$ is nonsingular. Then (5.7) is rewritten as

$$\begin{bmatrix} 2\pi I & 2\pi[2\pi + K_M^{1,1}]^{-1}K_M^{1,2} \\ K_M^{2,1} & 2\pi + K_M^{2,2} \end{bmatrix} \begin{bmatrix} \mu_M^1 \\ \mu_M^2 \end{bmatrix} = \begin{bmatrix} 2\pi[2\pi + K_M^{1,1}]^{-1}g_M^1 \\ g_M^2 \end{bmatrix}. \tag{5.8}$$

Denote this linear system by

$$(2\pi + L_M)\mu_M = \gamma_M, \qquad M \geq 1. \tag{5.9}$$

We solve this by using two-grid iteration, using a lower order system

$$(2\pi + L_N)\mu_N = \gamma_N$$

as our coarse grid system.

How is the partition of the domain $\mathcal{V}_M$ of the grid function μ_M to be made? The usual definition is to choose a small $\varepsilon > 0$ and to then define

$$\begin{aligned} \mathcal{V}_M^1 &= \{v_i | \text{ the distance from } v_i \text{ to an edge or vertex of } S \text{ is } \leq \epsilon\}, \\ \mathcal{V}_M^2 &= \mathcal{V}_M \backslash \mathcal{V}_M^1. \end{aligned} \tag{5.10}$$

Then for $\mu \in \mathbb{R}^{Mv}$, define

$$\mu^1 = \mu|\mathcal{V}_M^1, \qquad \mu^2 = \mu|\mathcal{V}_M^2. \tag{5.11}$$

This type of scheme is used in both [32] and [13] for planar BIE problems and leads to efficient iteration methods. The value of ϵ should be small enough to have the order of $2\pi + K_M^{1,1}$, called D_M, be small enough to directly solve linear systems with $2\pi + K_M^{1,1}$ as the coefficient matrix. The order $D_M = O(\epsilon M)$ and, as M increases, D_M increases at the same rate as for the original system. For planar BIE problems, D_M is generally sufficiently small.

For three-dimensional systems, D_M increases much more rapidly and the associated linear system can easily become too large to be handled directly. To avoid having a system with a large order, we have experimented using $\epsilon = 0$. Thus $v_i \in \mathcal{V}_M^1$ if and only if v_i is a vertex of

the original surface S or is on an edge of S. This ensures that $D_M = O(\sqrt{M})$. It still grows with M, but generally remains manageable. When $\epsilon > 0$, we can prove convergence of the iteration method in some cases, but we have no proof for the case $\epsilon = 0$, although it does give significantly improved results over the two-grid method for the original linear system $(2\pi + K_M)\mu_M = g_M$.

Example. We repeat the solution of the linear systems used in obtaining Table 5. The new results are given in Table 6. In every case in the table, the iteration ratios

$$R_l \equiv \frac{\|\mu_M^{(l)} - \mu_M^{(l-1)}\|_\infty}{\|\mu_M^{(l-1)} - \mu_M^{(l-2)}\|_\infty} \tag{5.12}$$

behaved in a somewhat unusual manner, oscillating in a small interval without converging to a limit. In Table 6, we give the "limiting mean" $\bar{R}$ of those oscillating values. In Fig. 1, we show these iteration ratios R_l for the case

$$(a, b, c) = (2.5, 2.5, 1), \qquad N = 8 \tag{5.13}$$

TABLE 6
Modified two-grid iteration for (1.3) *on a paraboloid.*

(a, b, c)	N	N_v	M	M_v	D_M	$\bar{R}$	NI
(2, 2, 1)	8	18	32	66	16	.25	14
	8	18	128	258	32	.41	22
	8	18	512	1026	64	.50	28
(2.5, 2.5, 1)	8	18	32	66	16	.31	16
	8	18	128	258	32	.46	25
	8	18	512	1026	64	.55	32
	32	66	128	258	32	.32	16
	32	66	512	1026	64	.47	25
(3, 3, 1)	8	18	32	66	16	.43	18
	8	18	128	258	32	.53	28
	8	18	512	1026	64	.60	36
	32	66	128	258	32	.42	18
	32	66	512	1026	64	.56	28
(4, 4, 1)	8	18	32	66	16	3.45	
	32	66	128	258	32	.74	41
	32	66	512	1026	64	.90	102

with three values of M. The iteration was carried out until the estimated iteration error was less than 10^{-12}. We also give the number of iterates NI needed to satisfy the test

$$\|\mu_M^{(l)} - \mu_M^{(l-1)}\|_\infty \le 10^{-8}$$

when beginning with $\mu_M^{(0)} = 0$.

Example. Solve problem (1.1) on a solid simplex with vertices

$$(0, 0, 0), (a, 0, 0), (0, b, 0), (0, 0, c).$$

The true solution was chosen to be

$$u(x, y, z) = \frac{1}{|(x, y, x) - \frac{1}{4}(a, b, c)|}.$$

The values of $\bar{R}$ are given in Table 7 for various values of (a, b, c), N, and M. The simplexes with $(a, b, c) = (1, 1, 1)$ and $(3, 3, 3)$ gave essentially the same results on the rate of convergence of the iteration method. In contrast with the case with S as a paraboloid, the iteration

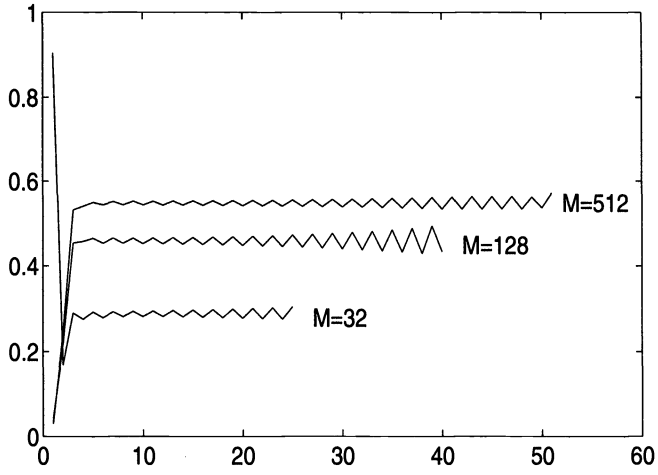


FIG. 1. *l vs.* R_l: *Iteration ratios for circular paraboloid.*

ratios R_l often converged to a limiting value. This was especially true when $M/N > 4$, but not always. Figure 2 contains graphs of the logarithms of the iteration corrections $\|\mu_M^{(l)} - \mu_M^{(l-1)}\|_\infty$ as l increases. The parameters for S in this figure are

$$(a, b, c) = (1, 1, 3), \quad N = 4, \quad M = 16, 64, 256.$$

From Fig. 2 and analogous figures for other values of the problem parameters, it is clear that the iteration is converging linearly, with the rate becoming worse as M increases.

TABLE 7
Modified two-grid iteration for (1.3) *on a simplex.*

(a, b, c)	N	N_v	M	M_v	D_M	$\bar{R}$	NI
(3, 3, 3)	4	10	16	34	22	.29	18
	4	10	64	130	46	.34	20
	4	10	256	514	94	.55	31
(1, 1, 1)	4	10	16	34	22	.29	19
	4	10	64	130	46	.35	21
	4	10	256	514	94	.55	33
	16	34	64	130	46	.28	17
	16	34	256	514	94	.43	25
(1, 1, 2)	4	10	16	34	22	.34	20
	4	10	64	130	46	.55	33
	4	10	256	514	94	.69	51
	16	34	64	130	46	.35	20
	16	34	256	514	94	.54	31
(1, 1, 3)	4	10	16	34	22	.42	24
	4	10	64	130	46	.62	42
	4	10	256	514	94	.74	65
	16	34	64	130	46	.42	24
	16	34	256	514	94	.62	35

6. Conclusions. Computations were carried out for a number of piecewise smooth surfaces with various choices of boundary data. From this, it is clear that the modified two-grid method based on (5.9) converges linearly. As N increases, the rate of linear convergence

as measured by $\bar{R}$ decreases, thus giving an improved rate of convergence for the iteration method. But as M increases for fixed N, the ratios $\bar{R}$ increase in size and the convergence is worse. The iteration based on (5.9) is clearly superior to the use of the unmodified two-grid iteration, as illustrated by comparing Tables 5 and 6.

It appears that $\bar{R}$ is a function of M/N, rather than a function of N and M separately. For constant M/N, the empirical rate of linear convergence $\bar{R}$ is also approximately constant. Thus we need to make N larger to maintain a good rate of convergence for larger values of M; of course, this means there is an additional cost in solving the coarse mesh equation. Since we also are solving systems of size D_M, it will not mean much of an additional cost to have N be approximately the same size as D_M. By doing so, the rate of convergence will not degrade as rapidly when M increases.

If we had used $\epsilon > 0$ in the definition (5.10), then we could have proved that the ratios would behave in the same manner as for the two-grid method on a smooth surface. But in that case, the size of D_M increases much more rapidly, as it is proportional to M rather than to $\sqrt{M}$. Whether it is better to use $\epsilon = 0$ or $\epsilon > 0$ depends on the particular problem being solved and the computer on which it is being solved. Generally it seems better to let $\epsilon = 0$ as long as the iteration is converging, and to turn to $\epsilon > 0$ only if convergence is not being attained.

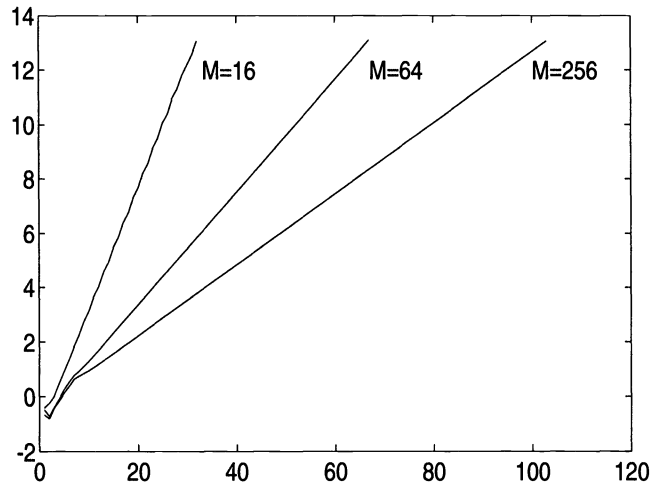


FIG. 2. *l vs.* $-\log_{10} \left\| \mu_M^{(l)} - \mu_M^{(l-1)} \right\|_\infty$. *Iteration corrections for simplex.*

Appendix. We give a proof of the result (4.10):

$$\text{(A.1)} \qquad \operatorname*{Limit}_{N\to\infty} \operatorname*{Sup}_{M>N} \|C_{N,M}\| = 0.$$

The proof is divided into two cases: (1) the exact surface S is used in all integrations and (2) the approximate surface $\tilde{S}$ is used. We begin by introducing some additional notation.

Define $\mathcal{R}_{\infty M} : C(S) \to \mathbb{R}^{M_v}$ by

$$\text{(A.2)} \qquad (\mathcal{R}_{\infty M} g)(v_i^{(M)}) = g(v_i^{(M)}), \quad i = 1, \ldots, M_v, \quad g \in C(S),$$

where elements of $\mathbb{R}^{M_v}$ are regarded as functions on the grid $\mathcal{V}_M$. Define $\mathcal{P}_{M\infty} : \mathbb{R}^{M_v} \to C(S)$ by

$$\text{(A.3)} \qquad (\mathcal{P}_{M\infty}\rho)(m_k(s,t)) = \sum_{i=1}^{6} \rho(v_{k,i}^{(M)}) \ell_i(s,t), \quad (s,t) \in \sigma, \quad k = 1, \ldots, M, \rho \in \mathbb{R}^{M_v}.$$

With these, we have the following identities:

$$\mathcal{R}_{\infty M}\mathcal{P}_{M\infty} = I, \tag{A.4}$$

$$\mathcal{P}_{M\infty}\mathcal{R}_{\infty M} = \mathcal{P}_M, \tag{A.5}$$

$$K_M = \mathcal{R}_{\infty M}\mathcal{K}\mathcal{P}_{M\infty}. \tag{A.6}$$

We also need the following relations between the coarse and fine grid operators:

$$\mathcal{R}_{MN}\mathcal{R}_{\infty M} = \mathcal{R}_{\infty N}, \tag{A.7}$$

$$\mathcal{P}_{M\infty}\mathcal{P}_{NM} = \mathcal{P}_{N\infty}. \tag{A.8}$$

The first of these relations is obvious; and the second depends on the uniqueness of quadratic interpolation and the fact that $\mathcal{T}_M$ is a refinement of $\mathcal{T}_N$.

To analyze the iteration J1–J4 of §4, we transform it into an equivalent problem on the space $C(S)$, thereby eliminating the use of the different spaces $\mathbb{R}^{M_v}$ and $\mathbb{R}^{N_v}$. Introduce

$$v^{(l)} := \mathcal{P}_{M\infty}\mu_M^{(l)}.$$

Recall the collocation equation

$$(I - \mathcal{P}_M\mathcal{K})u_M = \mathcal{P}_M g \tag{A.9}$$

and its matrix equivalent

$$(I - K_M)\mu_M = g_M. \tag{A.10}$$

The solutions of the two preceding equations satisfy

$$\mu_M = \mathcal{R}_{\infty M}u_M. \tag{A.11}$$

Using the above in the definitions of J1–J4 of §4, we obtain

$$\begin{aligned}
\mathcal{P}_{M\infty}r^{(l)} &= \mathcal{P}_{M\infty}g_M - \mathcal{P}_{M\infty}(\lambda - \mathcal{R}_{\infty M}\mathcal{K}\mathcal{P}_{M\infty})\mu_M^{(l)} \\
&= \mathcal{P}_M g - \lambda v^{(l)} + \mathcal{P}_M\mathcal{K}v^{(l)} = (\lambda - \mathcal{P}_M\mathcal{K})(u_M - v^{(l)}),
\end{aligned}$$

$$K_M r^{(l)} = \mathcal{R}_{\infty M}\mathcal{K}(\lambda - \mathcal{P}_M\mathcal{K})(u_M - v^{(l)}),$$

$$\begin{aligned}
d^{(l)} &= \mathcal{P}_{NM}(\lambda - K_N)^{-1}\mathcal{R}_{MN}\mathcal{R}_{\infty M}\mathcal{K}(\lambda - \mathcal{P}_M\mathcal{K})(u_M - v^{(l)}) \\
&= \mathcal{P}_{NM}(\lambda - K_N)^{-1}\mathcal{R}_{\infty N}\mathcal{K}(\lambda - \mathcal{P}_M\mathcal{K})(u_M - v^{(l)}),
\end{aligned}$$

$$\begin{aligned}
v^{(l+1)} = v(l) + \frac{1}{\lambda}\mathcal{P}_{M\infty}r^{(l)} + \frac{1}{\lambda}\mathcal{P}_{M\infty}d^{(l)} &= v^{(l)} + \frac{1}{\lambda}(\lambda - \mathcal{P}_M\mathcal{K})(u_M - v^{(l)}) \\
+ \frac{1}{\lambda}\mathcal{P}_{M\infty}\mathcal{P}_{NM}(\lambda - K_N)^{-1}\mathcal{R}_{\infty N}\mathcal{K}(\lambda - \mathcal{P}_M\mathcal{K})(u_M - v^{(l)}).&
\end{aligned}$$

Using (A.8), it holds that

$$u_M - v^{(l+1)} = \left\{ I - \frac{1}{\lambda}(\lambda - \mathcal{P}_M\mathcal{K}) - \frac{1}{\lambda}\mathcal{P}_{N\infty}(\lambda - K_N)^{-1}\mathcal{R}_{\infty N}\mathcal{K}(\lambda - \mathcal{P}_M\mathcal{K}) \right\}(u_M - v^{(l)}). \tag{A.12}$$

The following is a well-known identity. If $(I - BA)^{-1}$ exists, then $(I - AB)^{-1}$ exists and

$$(I - AB)^{-1} = I + A(I - BA)^{-1}B.$$

Apply this to convert $(\lambda - K_N)^{-1}$ to an equivalent form. With some straightforward manipulation and use of the earlier identities, we obtain

$$\begin{aligned} \mathcal{P}_{N\infty}(\lambda - K_N)^{-1}\mathcal{R}_{\infty N} &= \mathcal{P}_{N\infty}(\lambda - \mathcal{R}_{\infty N}\mathcal{K}\mathcal{P}_{N\infty})^{-1}\mathcal{R}_{\infty N} \\ &= \frac{1}{\lambda}\mathcal{P}_N(\lambda - \mathcal{K}\mathcal{P}_N)^{-1} = \frac{1}{\lambda}(\lambda - \mathcal{P}_N\mathcal{K})^{-1}\mathcal{P}_N. \end{aligned} \tag{A.13}$$

Using this result in (A.12) and some straightforward manipulation,

$$\begin{aligned} u_M - v^{(l+1)} &= \mathcal{C}_{N,M}(u_M - v^{(l)}), \\ \mathcal{C}_{N,M} &= (\lambda - \mathcal{P}_N\mathcal{K})^{-1}(\mathcal{P}_M - \mathcal{P}_N)\mathcal{K}. \end{aligned} \tag{A.14}$$

Since $\mathcal{K}$ is compact on $C(S)$, it is well known that

$$\operatorname*{Limit}_{N\to\infty} \|\mathcal{K} - \mathcal{P}_N\mathcal{K}\| = 0.$$

Thus $(\lambda - \mathcal{P}_N\mathcal{K})^{-1}$ is uniformly bounded in N and

$$\operatorname*{Limit}_{N\to\infty} \operatorname*{Sup}_{M>N} \|\mathcal{P}_M\mathcal{K} - \mathcal{P}_N\mathcal{K}\| = 0.$$

This proves that

$$\operatorname*{Limit}_{N\to\infty} \operatorname*{Sup}_{M>N} \|\mathcal{C}_{N,M}\| = 0, \tag{A.15}$$

which shows that the iterates $v^{(l)}$ converge to u_M as $l \to \infty$, provided that N is chosen sufficiently large. As $M \to \infty, \mathcal{C}_{N,M}$ approaches

$$(\lambda - \mathcal{P}_N\mathcal{K})^{-1}(I - \mathcal{P}_N)\mathcal{K}.$$

This, too, is less than one for N chosen sufficiently large; and thus the convergence in (A.14) will exhibit a "mesh-independence principle."

To complete the result, these results must be related to the matrix-vector equation (A.1). Write the error equation (A.14) as

$$e^{(l+1)} = \mathcal{C}_{N,M}e^{(l)} \tag{A.16}$$

and the error equation (4.7) as

$$\epsilon^{(l+1)} = C_{N,M}\epsilon^{(l)}. \tag{A.17}$$

The errors are connected by the relation

$$\mathcal{R}_{\infty M}e^{(l)} = \epsilon^{(l)}, \qquad l \geq 0. \tag{A.18}$$

Since $u_M \in \text{Range}(\mathcal{P}_M)$, we also have

$$\mathcal{P}_{\infty M}\epsilon^{(l)} = \mathcal{P}_{\infty M}\mathcal{R}_{\infty M}e^{(l)} = \mathcal{P}_M e^{(l)} = e^{(l)}, \qquad l \geq 0. \tag{A.19}$$

Using these results with (A.16)

$$\begin{aligned} \epsilon^{(l+1)} &= \mathcal{R}_{\infty M}e^{(l+1)} = \mathcal{R}_{\infty M}\mathcal{C}_{N,M}e^{(l)} \\ &= \mathcal{R}_{\infty M}\mathcal{C}_{N,M}\mathcal{P}_{\infty M}\epsilon^{(l)}, \qquad l \geq 0. \end{aligned}$$

Consider the case $l = 0$ and note that $\epsilon^{(0)}$ can be chosen arbitrarily. Then we see that the action of $C_{N,M}$ and $\mathcal{R}_{\infty M}\mathcal{C}_{N,M}\mathcal{P}_{\infty M}$ is the same on all $\epsilon^{(0)} \in \mathbb{R}^{M_v}$ and thus

$$C_{N,M} = \mathcal{R}_{\infty M}\mathcal{C}_{N,M}\mathcal{P}_{M,\infty}. \tag{A.20}$$

To show (A.1), use

$$\|C_{N,M}\| \le \|\mathcal{R}_{\infty M}\mathcal{C}_{N,M}\mathcal{P}_{\infty M}\| \le \frac{5}{3}\|\mathcal{C}_{N,M}\|. \tag{A.21}$$

The approximate surface case. When the surface S is replaced by $\tilde{S}$, then the matrix elements in (A.10) over S are replaced by the corresponding integrals over $\tilde{S}$. Let $\tilde{C}_{N,M}$ denote the matrix in (A.17) when using $\tilde{S}$ (actually using $\tilde{S} \equiv \tilde{S}_N$ when defining K_N and using $\tilde{S}_M$ when defining K_M). The result (A.1) is shown by a perturbation argument using

$$\|C_{N,M} - \tilde{C}_{N,M}\| = O(h_N). \tag{A.22}$$

The matrix norm used in (A.21) is the row-norm induced by $\|\cdot\|_\infty$ on $\mathbb{R}^{M_v}$.

We compare the integrals

$$\int_\sigma K(v_i^{(M)}, m_k^{(M)}(s,t))\ell_j(s,t)|(D_s m_k^{(M)} \times D_t m_k^{(M)})(s,t)|d\sigma \tag{A.23}$$

based on the exact surface S, with

$$\int_\sigma K(v_i^{(M)}, \tilde{m}_k^{(M)}(s,t))\ell_j(s,t)|(D_s \tilde{m}_k^{(M)} \times D_t \tilde{m}_k^{(M)})(s,t)|d\sigma, \tag{A.24}$$

which are based on the approximate surface $\tilde{S}$; we consider the corresponding integrals for the coarse mesh parameter N. The kernel function is, of course, the double layer function

$$K(P, Q) = \frac{\partial}{\partial v_Q}\left[\frac{1}{|P - Q|}\right].$$

The measurement of the error in (A.24) and its effect on the row norm of K_M and K_N is based on a slight modification of a derivation given in the thesis of Chien [14]. Using this thesis, it can be shown that

$$\|K_M - \tilde{K}_M\| = O(h_M), \qquad \|K_N - \tilde{K}_N\| = O(h_N). \tag{A.25}$$

Chien [14, §3.6] looked at the approximation of the solid angle on the surface of S at the nodes of $\mathcal{V}_M$. As this is a very involved derivation, based on asymptotic expansions of the error in powers of h, we omit it here. Using (A.25), the assertion (A.22) follows easily and also the assertion (A.1).

REFERENCES

[1] P. ANSELONE, *Collectively Compact Operator Approximation Theory*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[2] K. ATKINSON, *Iterative variants of the Nyström method for the numerical solution of integral equations*, Numer. Math., 22 (1973), pp. 17–31.

[3] ———, *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1976.

[4] K. ATKINSON, *The numerical solution of Laplace's equation in three dimensions* II, in Numerical Treatment of Integral Equations, J. Albrecht and L. Collatz, eds., Birkhäuser, Basel, 1980, pp. 1–23.

[5] ———, *The numerical solution of Laplace's equation in three dimensions*, SIAM J. Numer. Anal., 19 (1982), pp. 263–274.

[6] ———, *Piecewise polynomial collocation for integral equations on surfaces in three dimensions*, J. Integral Equations Appl., 9 (1985), pp. 25–48.

[7] ———, *Solving integral equations on surfaces in space*, in Constructive Methods for the Practical Treatment of Integral Equations, G. Hämmerlin and K. Hoffman, eds., Birkhäuser, Basel, 1985, pp. 20–43.

[8] ———, *Algorithm* 629 : *An integral equation program for Laplace's equation in three dimensions*, ACM Trans. Math. Software, 11 (1985), pp. 85–96.

[9] ———, *An empirical study of the numerical solution of integral equations on surfaces in* $\mathbb{R}^3$, Reports on Computational Mathematics #1, Dept. of Mathematics, University of Iowa, Iowa City, 1989.

[10] ———, *A survey of boundary integral equation methods for the numerical solution of Laplace's equation in three dimensions*, in Numerical Solution of Integral Equations, M. Golberg, ed., Plenum Press, New York, 1990, pp. 1–34.

[11] K. ATKINSON AND D. CHIEN, *Piecewise polynomial collocation for boundary integral equations*, SIAM J. Sci. Comput., to appear.

[12] K. ATKINSON AND I. GRAHAM, *An iterative variant of the Nyström method for boundary integral equations on non-smooth boundaries*, in The Mathematics of Finite Elements and Applications, John Whiteman, ed., Academic Press, London, 1988, pp. 297–304.

[13] ———, *Iterative solution of linear systems arising from the boundary integral method*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 694–722.

[14] D. CHIEN, *Piecewise Polynomial Collocation for Integral Equations on Surfaces in Three Dimensions*, Ph. D. thesis, University of Iowa, Iowa City, 1991.

[15] ———, *Piecewise polynomial collocation for integral equations with a smooth kernel on surfaces in three dimensions*, Integral Equations Appl., 5 (1993), pp. 315–344.

[16] J. ELSCHNER, *The double layer potential operator over polyhedral domains, II: spline Galerkin methods*, Math. Methods Appl. Sci., 15 (1992), pp. 23–37.

[17] J. GUERMOND, *Numerical quadrature for layer potentials over curved domains in* $\mathbb{R}^3$, SIAM J. Numer. Anal., 29 (1992), pp. 1347–1369.

[18] N. GÜNTER, *Potential Theory*, Ungar, New York, 1967.

[19] W. HACKBUSCH, *Die schnelle Auflösung der Fredholmschen Integralgleichungen zweiter Art*, Beiträge Numer. Math., 9 (1981), pp. 47–62.

[20] ———, *Multi-grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[21] W. HACKBUSCH AND Z. NOWAK, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.

[22] F. HEBEKER, *Efficient boundary element methods for three-dimensional exterior viscous flows*, Numer. Meth. Partial, 2 (1986), pp. 273–297.

[23] ———, *On the numerical treatment of viscous flows against bodies with corners and edges by boundary element and multigrid methods*, Numer. Math., 52 (1988), pp. 81–99.

[24] R. KRESS, *Linear Integral Equations*, Springer-Verlag, Berlin, 1990.

[25] J. MANDEL, *On multilevel iterative methods for integral equations of the second kind and related problems*, Numer. Math., 46 (1985), pp. 147–157.

[26] S. MIKHLIN, *Mathematical Physics: An Advanced Course*, North-Holland, Amsterdam, the Netherlands, 1970.

[27] J. NEDELEC, *Curved finite element methods for the solution of singular integral equations on surfaces in* $\mathbb{R}^3$, Computer Meth. Appl. Mech. Engrg., 8 (1976), pp. 61–80.

[28] Z. NOWAK, *Use of the multigrid method for Laplacian problems in three dimensions*, in Multigrid Methods, W. Hackbusch and U. Trottenberg, eds., Lecture Notes in Math. #960, Springer-Verlag, 1982, pp. 576–598.

[29] T. PETERSDORFF AND E. STEPHAN, *Decomposition in edge and corner singularities for the solution of the Dirichlet problem of the Laplacian in a polyhedron*, Math. Nachr., 149 (1990), pp. 71–104.

[30] J. RADON, *Über die Randwertaufgaben beim logarithmischen Potential*, Sber. Akad. Wiss. Wien, 128 (1919), Abt. IIa, pp. 1123–1167.

[31] V. ROKHLIN, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.

[32] H. SCHIPPERS, *Multigrid methods in boundary integral equations*, Numer. Math., 46 (1985), pp. 351–363.

[33] ———, *Multigrid methods in boundary element calculations*, in Boundary Elements IX: Vol. 1 — Mathematical and Computational Aspects, C. Brebbia, W. Wendland, and G. Huhn, eds., Springer-Verlag, New York, Berlin, 1987, pp. 475–492.

[34] C. SCHWAB AND W. WENDLAND, *On numerical cubatures of singular surface integrals in boundary element methods*, Numerische Math. 62 (1992), pp. 343–369.

[35] A. STROUD, *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[36] W. WENDLAND, *Die Behandlung von Randwertaufgaben im* $\mathbb{R}^3$ *mit Hilfe von Einfach und Doppelschichtpotentialen*, Numer. Math., 11 (1968), pp. 380–404.

[37] ———, *Asymptotic accuracy and convergence for point collocation methods*, in Topics in Boundary Element Research, Vol. II: Time-Dependent and Vibration Problems, C. Brebbia, ed., Springer-Verlag, Berlin, 1985.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1105–1110, September 1994

A FAST METHOD FOR THE NUMERICAL EVALUATION OF CONTINUOUS FOURIER AND LAPLACE TRANSFORMS*

DAVID H. BAILEY† AND PAUL N. SWARZTRAUBER‡

Abstract. The fast Fourier transform (FFT) is often used to compute numerical approximations to continuous Fourier and Laplace transforms. However, a straightforward application of the FFT to these problems often requires a large FFT to be performed, even though most of the input data to this FFT may be zero and only a small fraction of the output data may be of interest. In this note, the "fractional Fourier transform," previously developed by the authors, is applied to this problem with a substantial savings in computation.

Key words. fast Fourier transform, fractional Fourier transform, numerical integration

AMS subject classifications. 65T20, 65T10, 65Y20

1. Introduction. The continuous Fourier transform (CFT) of a function $f(t)$ (which may have real or complex values) and its inverse will be defined here as

$$F[f](x) = \int_{-\infty}^{\infty} f(t)e^{-itx}dt, \tag{1}$$

$$F^{-1}[f](x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(t)e^{itx}dt, \tag{2}$$

where i is the imaginary unit. The discrete Fourier transform (DFT) and the inverse DFT of an n-long sequence z (which may have real or complex values) will be defined here as

$$D_k(z) = \sum_{j=0}^{n-1} z_j e^{-2\pi ijk/m}, \tag{3}$$

$$D_k^{-1}(z) = \frac{1}{m}\sum_{j=0}^{n-1} z_j e^{2\pi ijk/m}. \tag{4}$$

Straightforward evaluation of the DFT using these formulas is expensive, even when the exponential factors are precomputed. This cost can be greatly reduced by employing one of the variants of the fast Fourier transform (FFT) algorithm [1], [2], [5], [6].

The methods developed in this paper are equally applicable to the numerical evaluation of continuous Laplace transforms, and the approach is the same as for Fourier transforms. For brevity, this exposition will focus on evaluating continuous Fourier transforms.

2. A conventional scheme for evaluating CFTs with FFTs. There are of course a number of advanced techniques for the approximate numerical evaluation of definite integrals [4]. Unfortunately, most of these techniques deal with evaluating only one integral, rather than a large number of integrals, each with a different integrand. Evaluating the CFT falls in this latter category, since one usually requires the values of this integral for a large range of x.

The FFT can be effectively applied to this problem as follows. Let us assume that $f(t)$ is zero outside the interval $(-a/2, a/2)$. Let $\beta = a/m$ be the interval in t for the m input values of $f(t)$, which are assumed centered at zero, where m is even. To be specific, the abscissas for the input data are $t_j = (j - m/2)\beta$, $0 \le j < m$. The abscissas for the output data set will

*Received by the editors November 9, 1992; accepted for publication (in revised form) July 30, 1993.

†Numerical Aerodynamic Simulation (NAS) Systems Division at National Aeronautic and Space Administration, Moffett Field, California 94035-1000 (dbailey@nas.nasa.gov).

‡National Center for Atmospheric Research, Boulder, Colorado 80307 (pauls@near.ucar.edu).

be defined as $x_k = 2\pi(k - m/2)/a = 2\pi(k - m/2)/(m\beta)$, $0 \le k < m$. This definition of x_k will be explained later. Then we can write

$$F(x_k) = \int_{-\infty}^{\infty} f(t)e^{-itx_k}dt \tag{5}$$

$$= \int_{-a/2}^{a/2} f(t)e^{-itx_k}dt \tag{6}$$

$$\approx \sum_{j=0}^{m-1} f(t_j)e^{-it_jx_k}\beta \tag{7}$$

$$= \beta \sum_{j=0}^{m-1} f(t_j)e^{-2\pi i(j-m/2)(k-m/2)/m} \tag{8}$$

$$= \beta e^{\pi i(k-m/2)} \sum_{j=0}^{m-1} f(t_j)e^{\pi ij}e^{-2\pi ijk/m} \tag{9}$$

$$= (-1)^k \beta D_k[\{(-1)^j f(t_j)\}], \qquad 0 \le k < m. \tag{10}$$

The DFT indicated in (10) can of course be rapidly evaluated using an FFT.

It is now clear why x_k was defined as above—this is necessary for the expression (9) to be in the form of a DFT. In other words, the interval (i.e., resolution) of the output results of this procedure is fixed at the value $2\pi/(m\beta)$ as soon as one specifies the number m of input values and their interval β.

Let us assume that comparable intervals are required in t and x to obtain accurate results. Then one must have $\beta \approx 2\pi/(m\beta)$, or in other words $m \approx 2\pi/\beta^2$. From this observation it is clear that if one wishes to obtain accurate, high-resolution results using this procedure, then it may be necessary to set m very large, perhaps much larger than the actual size of the nonzero input data set. Another way of saying this is that the input data $f(t_j)$ may need to be padded on both sides with many zeros to obtain the desired resolution in the output data.

A specific example will illustrate these issues. Let us consider the problem of numerically computing the Fourier transform of the Gaussian probability density function

$$f(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}. \tag{11}$$

It is well known that the Fourier transform of $f(t)$ is

$$F(x) = e^{-x^2/2}. \tag{12}$$

Note that outside the interval $[-10, 10]$, we have $f(t) < 7.7 \times 10^{-23}$ and $F(x) < 1.9 \times 10^{-22}$.

Suppose now that one wishes to compute the Fourier transform using the above FFT scheme with a resolution of $\sqrt{2\pi}/256$ in both t and x. One can do this provided that one sets $m = 65,536$, since in that case $\beta = 2\pi/(m\beta)$. However, both the input and output data then span the interval $[-m\beta/2, m\beta/2]$, or approximately $[-320.8, 320.8]$.

In short, as a consequence of the fundamentally inflexible nature of the FFT when applied to this problem, it turns out that a large fraction of the input data to the FFT used to evaluate (10) are considerably smaller than the machine "epsilon" of most computers, even when using double precision arithmetic. Also, only a small fraction (the central values) of the output of this FFT are of interest. Thus we seek a more flexible scheme for problems of this sort, one that can produce the central section of the results, using only the central nonzero input data, with a savings of computation.

3. The Fractional Fourier Transform (FRFT). We will employ here a generalization of the DFT that has been termed the FRFT [3]. It is defined on the m-long complex sequence x as

$$G_k(x,\alpha) = \sum_{j=0}^{m-1} x_j e^{-2\pi i j k \alpha}. \tag{13}$$

The parameter α is not restricted to rational numbers and in fact may be any complex number. Note that the ordinary DFT and its inverse are special cases of the FRFT.

A fortunate fact is that the FRFT admits computation by means of a fast algorithm, one with complexity comparable to the FFT. In fact, it employs the FFT in a crucial step. This algorithm can be stated as follows. Define the $2m$-long sequences y and z as

$$y_j = x_j e^{-\pi i j^2 \alpha}, \qquad 0 \le j < m, \tag{14}$$

$$y_j = 0, \qquad m \le j < 2m, \tag{15}$$

$$z_j = e^{\pi i j^2 \alpha}, \qquad 0 \le j < m, \tag{16}$$

$$z_j = e^{\pi i (j-2m)^2 \alpha}, \qquad m \le j < 2m. \tag{17}$$

It is then shown in [3] that

$$G_k(x,\alpha) = e^{-\pi i k^2 \alpha} D_k^{-1}[\{D_j(y) D_j(z)\}], \qquad 0 \le k < m. \tag{18}$$

The remaining m results of the final inverse DFT are discarded. Element-wise complex multiplication is implied in the expression $D_j(y)D_j(z)$. These three DFTs can of course be efficiently computed using $2m$-point FFTs.

To compute a different m-long segment $G_{k+s}(x,\alpha)$, $0 \le k < m$, it is necessary to slightly modify the above procedure. In this case z is as follows:

$$z_j = e^{\pi i (j+s)^2 \alpha}, \qquad 0 \le j < m, \tag{19}$$

$$z_j = e^{\pi i (j+s-2m)^2 \alpha}, \qquad m \le j < 2m. \tag{20}$$

Then we have

$$G_{k+s} = e^{-\pi i (k+s)^2 \alpha} D_k^{-1}[\{D_j(y) D_j(z)\}], \qquad 0 \le k < m. \tag{21}$$

Note that the exponential factors in (14)–(21) can be precomputed. Furthermore, the DFT of the z sequence can also be precomputed. Thus the cost of an m-point FRFT is only about four times the cost of an m-point FFT.

4. Computing CFTs with FRFTs. One of the applications of FRFTs mentioned in [3] is computing DFTs of sparse sequences, i.e., sequences that are mostly zero. It is particularly effective when only a small portion of the input data values are nonzero, and only a small portion of the output values are required. Thus, the FRFT is very well suited for the problem at hand.

As before, we will assume that $f(t)$ is zero outside the interval $(-a/2, a/2)$, and that $\beta = a/m$ is the interval of the m input values of $f(t)$, which will be assumed centered at zero; i.e., $t_j = (j - m/2)\beta$, $0 \le j < m$. We will now define γ to be the interval desired for the output data, so that they are given by $x_k = (k - m/2)\gamma$, $0 \le k < m$. Set $\delta = \beta\gamma/(2\pi)$. Then we can write

$$F(x_k) = \int_{-\infty}^{\infty} f(t) e^{-i t x_k} dt \tag{22}$$

$$= \int_{-a/2}^{a/2} f(t)e^{-itx_k}dt \tag{23}$$

$$\approx \sum_{j=0}^{m-1} f(t_j)e^{-it_j x_k}\beta \tag{24}$$

$$= \beta \sum_{j=0}^{m-1} f(t_j)e^{-i(j-m/2)(k-m/2)\beta\gamma} \tag{25}$$

$$= \beta e^{\pi i(k-m/2)m\delta} \sum_{j=0}^{m-1} f(t_j)e^{\pi ijm\delta}e^{-2\pi ijk\delta} \tag{26}$$

$$= \beta e^{\pi i(k-m/2)m\delta} G_k[\{f(t_j)e^{\pi ijm\delta}\}, \delta], \qquad 0 \le k < m. \tag{27}$$

We now have an economical means of computing the central m results of this transform. Additional m-long segments of results can be obtained by applying the more general form of the FRFT given at the end of the previous section.

5. Implementation example. Collecting the results we have obtained so far, we have

$$F(x_k) = \int_{-\infty}^{\infty} f(t)e^{-itx_k}dt \tag{28}$$

$$\approx \beta e^{\pi i(k-m/2)m\delta} G_k[\{f(t_j)e^{\pi ijm\delta}\}, \delta] \tag{29}$$

$$= \beta e^{\pi i[(k-m/2)m\delta - k^2\delta]} D_k^{-1}[\{D_j(y)D_j(z)\}], \qquad 0 \le k < m, \tag{30}$$

where the $2m$-long sequences y and z are given by

$$y_j = f(t_j)e^{\pi i(jm\delta - j^2\delta)}, \qquad 0 \le j < m, \tag{31}$$

$$y_j = 0, \qquad m \le j < 2m, \tag{32}$$

$$z_j = e^{\pi ij^2\delta}, \qquad 0 \le j < m, \tag{33}$$

$$z_j = e^{\pi i(j-2m)^2\delta}, \qquad m \le j < 2m. \tag{34}$$

Let us now reconsider the problem of numerically computing the Fourier transform of the Gaussian probability density function

$$f(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/2}. \tag{35}$$

Recall that the Fourier transform of $f(t)$ is

$$F(x) = e^{-x^2/2}. \tag{36}$$

Also recall that outside the interval $[-10, 10]$, we have $f(t) < 7.7 \times 10^{-23}$ and $F(x) < 1.9 \times 10^{-22}$.

To compute the Fourier transform using the conventional FFT scheme with a resolution of $\sqrt{2\pi}/256$ in both t and x, we have $\beta = \gamma = \sqrt{2\pi}/256$ and $m = 2\pi/(\beta\gamma) = 65,536$. An implementation of this scheme requires 3.95 seconds on a SGI Personal Iris. Compared with exact values of the transform computed from the formula above, the root-mean-square (RMS) error of the central 2048 results is 1.14×10^{-16}.

Note however that only the central 2048 result values, which span the approximate interval $[-10.02, 10.02]$, are required—outside this interval both the input and output function values

can be assumed to be zero with high accuracy. Applying the FRFT scheme with $m = 2048$ yields these same results in 0.40 seconds on the Iris, fully ten times faster. The RMS error of the FRFT results is 2.96×10^{-16}. Both of these error figures are virtually at the limit of machine precision.

As an aside, while it is expected from the results of this paper that the FRFT results should be within machine precision of the FFT results, it is remarkable that both are also within machine precision of the actual values resulting from the analytic formula (36). This is in spite of the fact that the interval used in the step function approximation to the integral has the modestly small value $\beta \approx 0.01$. This surprising phenomenon evidently derives from the fact that step function approximations to integrals converge to the exact results very rapidly provided that the function $f(t)$ has rapidly decreasing Fourier coefficients [4]. This condition is met by the Gaussian probability density function.

It should also be mentioned that in this specific example, one may accurately recover the original data by performing an inverse transform with the FRFT. This can be done by employing formulas (30)–(34) with δ replaced by $-\delta$. This operation works because the results of the forward transform, like the original data, can be assumed to be precisely zero outside the interval $[-10.02, 10.02]$. For other functions $f(t)$ this condition may not hold, and as a result this inversion operation would not be expected to accurately reconstruct the original data.

6. More advanced quadrature schemes. The techniques described above were formulated on the basis of a simple step-function approximation to the integral. These techniques can also be applied to a Simpson's rule approximation, as follows. Let $c_j = 4$ for odd j and $c_j = 2$ for nonzero even j. Then we have

$$F(x_k) = \int_{-\infty}^{\infty} f(t)e^{-itx_k}dt \tag{37}$$

$$= \int_{-a/2}^{a/2} f(t)e^{-itx_k}dt \tag{38}$$

$$\approx \sum_{j=0}^{m-1} c_j f(t_j)e^{-it_j x_k}\beta \tag{39}$$

$$= \beta \sum_{j=0}^{m-1} c_j f(t_j)e^{-i(j-m/2)(k-m/2)\beta\gamma} \tag{40}$$

$$= \beta e^{\pi i(k-m/2)m\delta} \sum_{j=0}^{m-1} c_j f(t_j)e^{\pi ijm\delta}e^{-2\pi ijk\delta} \tag{41}$$

$$= \beta e^{\pi i(k-m/2)m\delta} G_k[\{c_j f(t_j)e^{\pi ijm\delta}\}, \delta], \qquad 0 \le k < m. \tag{42}$$

One difficulty with using Simpson's rule to evaluate Fourier integrals is that the accuracy of the quadrature depends on the size of the product $f(t)e^{-itx}$. In particular, the accuracy may deteriorate for large values of x, due to the oscillatory nature of the integrand. For this reason some scientists prefer using Filon's method [4, p. 151] to evaluate Fourier integrals. Filon's method has the advantage that the accuracy depends only on the smoothness of $f(t)$. Again, an FRFT-based scheme can be profitably applied here, since Filon's method employs equispaced intervals. The approach is entirely similar to the above, and details will not be given here.

In general, the FRFT-based technique described above can be profitably applied to the numerical evaluation of any integral transform where (1) both the input function values $f(t_j)$

and the output transform values $F(x_k)$ are equally spaced, (2) a large fraction (more than 75%) of $f(t_j)$ are either zero or within machine tolerance of zero relative to $\max |f(t_j)|$, and (3) only a limited range of $F(x_k)$ is required.

REFERENCES

[1] D. H. BAILEY, *FFTs in external or hierarchical memory*, J. Supercomputing, 4 (1990), pp. 23–35.
[2] ———, *A high-performance FFT algorithm for vector supercomputers*, International J. Supercomputer Appl., 2 (1988), pp. 82–87.
[3] D. H. BAILEY AND P. N. SWARZTRAUBER, *The fractional Fourier transform and applications*, SIAM Rev., 33 (1991), pp. 389–404.
[4] P. J. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*, Wiley, New York, 1984.
[5] P. N. SWARZTRAUBER, *Multiprocessor FFTs*, Parallel Comput., 5 (1987), pp. 197–210.
[6] C. VAN LOAN, *Computational Frameworks for the Fast Fourier Transform*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1111–1125, September 1994

008

SPLINE INTERPOLATION AND SMOOTHING ON HYPERSPHERES*

H. J. TAIJERON†, A. G. GIBSON‡, AND C. CHANDLER§

Abstract. The authors generalize Wahba's theory [*SIAM J. Sci. Statist. Comput.*, 2 (1981), pp. 5–16] on spline interpolation and smoothing on the surface of the two-dimensional unit sphere to arbitrary dimensional hyperspheres. As a consequence, practical solutions to minimum norm interpolation and smoothing problems on hyperspheres are provided in terms of certain hyperspherical splines. In addition, Wahba's results for powers of the Laplace–Beltrami operator are extended to more general operators, and Wahba's Hilbert space of constant functions is expanded to allow more than just constant functions. Extensive curve fitting calculations are made for some two- and higher-dimensional test problems using hyperspherical harmonics and hyperspherical splines. It is found that hyperspherical splines yield better fits than hyperspherical harmonics for test functions that possess no symmetry and are not infinitely differentiable. Tests have been run using several continuous functions; satisfactory absolute errors can be obtained using only 20–50 splines. The speed of convergence of the spline approximations did not vary much with the smoothness or symmetry properties of the functions being fit.

Key words. splines on hyperspheres, hyperspherical harmonics, interpolation, smoothing

AMS subject classifications. 41A15, 41A63, 65D05, 65D10

1. Introduction.

The primary goal of this work is to develop a theory of splines on hyperspheres that provides practical solutions to interpolation and smoothing problems on the surface of hyperspheres. Our results represent a generalization of the theory of Wahba for splines on the unit circle [12] and splines on the two-dimensional sphere in three-dimensional space [13]. In [13] Wahba was motivated by an interest in the interpolation and smoothing of certain geophysical data taken at various points on the earth. Our higher-dimensional generalizations were motivated by problems in the theory of nonrelativistic multichannel quantum scattering [5]. When there are, for example, three bodies in the final state of a quantum reaction, their relative motion at a fixed energy is represented by points on a five-dimensional hypersphere in a six-dimensional space. Functions that could be used for interpolating and smoothing data could also be used as a basis in which to expand the scattering amplitudes of the theory. Such basis sets could be formed from hyperspherical harmonics [3], [10] or, more interestingly, from hyperspherical spline functions.

We have generalized Wahba's results in three basic ways. First, we have extended her theory to higher dimensions, valid on any unit hypersphere Ω_{p+2} of dimension $p+1$ in $(p+2)$-dimensional space for $p = 0, 1, 2, \ldots$. Second, we have generalized Wahba's results for powers of the Laplace–Beltrami operator to more general operators A. Third, we have modified Wahba's decomposition $\mathcal{H}_m(\Omega_3) = \{1\} \oplus \mathcal{H}_m^1(\Omega_3)$, where $\{1\}$ is the one-dimensional space of constant functions on Ω_3, and $\mathcal{H}_m^1(\Omega_3)$ is a reproducing kernel Hilbert space. Freeden [6] has previously suggested enlarging $\{1\}$ on Ω_3. In our theory the space $\{1\}$ is enlarged to become a finite- dimensional reproducing kernel Hilbert space $\mathcal{H}_m^0$ that may include more than just constant functions. This permits a more flexible combination of hyperspherical harmonics and hyperspherical splines in an interpolation or smoothing scheme.

The numerical tests of our theory were concentrated on the interpolatory fitting of functions defined on Ω_{p+2} with $p = 1$, 2, or 4. For $p = 1$ we found, numerically, that fits using

*Received by the editors April 15, 1992; accepted for publication (in revised form) August 2, 1993. This research was supported in part by National Science Foundation grants PHY-9104459 and INT-880898.

†Mathematics and Science Division, University of Guam, Mangilao, Guam 96913.

‡Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico 87131 (archie@math.unm.edu). During the preparation of this manuscript, this author was supported by the Australian Research Council and the University of New South Wales.

§Department of Physics and Astronomy, University of New Mexico, Albuquerque, New Mexico 87131 (chandler@triton.unm.edu).

only spherical harmonics were considerably better for symmetric infinitely differentiable functions than those with hyperspherical splines. The fits with splines, on the other hand, were better for functions that had little or no symmetry or were not infinitely differentiable. In higher dimensions splines continued to fit our test functions reasonably well, and often these fits were superior to those with hyperspherical harmonics, even for infinitely differentiable functions. We also found that, in general, the addition of a few hyperspherical harmonics to the space {1} gave improved approximations only for the smooth functions which could be approximated better by hyperspherical harmonics.

Our tests using several continuous positive functions on Ω_{p+2} with $p = 1, 2$, or 4 showed that we could obtain satisfactory absolute errors by using only 20–50 data points. The speed of convergence did not vary much with the smoothness or symmetry properties of the function being fit. However, this was not the case with hyperspherical harmonics. These functions were sensitive to the smoothness and symmetry properties of the function being approximated.

Another feature of hyperspherical splines is that the corresponding interpolatory scheme is robust. The location of the data points (knots) may be essentially random, and the unique solution of the interpolation problem will always pass through all of the data points. On the other hand, a random selection of data points may lead to ill-conditioned systems of equations for a fitting scheme that is based on hyperspherical harmonics. In addition, we were successful in finding an algorithm for selecting hyperspherical splines by iteratively choosing one additional data point where it is most needed, but we were unsuccessful in finding an automatic optimum way of improving fits with hyperspherical harmonics.

Section 2 is devoted to generalizing Wahba's theory of solving minimum-norm interpolation and smoothing problems to higher-dimensional hyperspheres. Reproducing kernel Hilbert space methods are used to find the solutions to these problems by invoking two general lemmas of Kimeldorf and Wahba [9]. The reproducing kernels defined by infinite sums in §2 appear to be summable only for the $p = 0$ circle case. Consequently, for $p > 0$ we modify these kernels to obtain closed-form expressions by using a different, but equivalent, norm in the Hilbert space. This modification is analogous to the one done in [13, §3]. The modified reproducing kernels have proved to take considerably less time to calculate than the original kernels.

In §3 we present a summary of our numerical methods and results. We compare hyperspherical spline approximations with hyperspherical harmonic approximations on the surface of the unit hyperspheres in three-, four-, and six-dimensional space. In our comparisons we use three types of functions. The first is infinitely differentiable and symmetric, and the second function is infinitely differentiable but not symmetric. The third is continuous, has a point at which it is not differentiable, and has no symmetry.

Most of the results of this paper are contained in the 1989 Ph.D. dissertation of Taijeron [11].

2. Theory of splines on hyperspheres.

2.1. The reproducing kernel Hilbert space $\mathcal{H}_A(\Omega_{p+2})$. Let p be a nonnegative integer, and let Ω_{p+2} denote the $(p+1)$-dimensional unit hypersphere in $(p+2)$-dimensional space. Let L_{p+2} denote the Laplace–Beltrami operator in (p+2) dimensions [3], with L_2 the special case defined by $L_2 \equiv d^2/d\phi^2, 0 \le \phi < 2\pi$. Let $h(\ell, p) \equiv (2\ell + p)(\ell + p - 1)!/(\ell! p!)$, for ℓ and p not both zero, and $h(0, 0) \equiv 1$. For $p > 0$, let $S_{\ell m}(\zeta), \zeta \in \Omega_{p+2}, \ell = 0, 1, 2, \ldots, m = 1, 2, \ldots, h(\ell, p)$, denote the $h(\ell, p)$ normalized real hyperspherical harmonics of order ℓ in $(p+2)$ dimensions [3]. For $p = 0$, let $S_{01}(\zeta) \equiv (2\pi)^{-1/2}$, and let $S_{\ell 1}(\zeta) \equiv \pi^{-1/2} \cos \ell\phi$, and $S_{\ell 2}(\zeta) \equiv \pi^{-1/2} \sin \ell\phi$, for $\ell = 1, 2, \ldots$, where ϕ is the polar angle of the unit vector $\zeta \in \Omega_2$

(circle). Then, for any integer $p \geq 0$,

$$-L_{p+2}S_{\ell m} = \ell(\ell+p)S_{\ell m}, \tag{1}$$

and the eigenvalue $\lambda_\ell \equiv \ell(\ell+p)$ of $-L_{p+2}$ is of multiplicity $h(\ell, p)$. $\{S_{\ell m}\}$ forms a complete set of orthonormal functions on Ω_{p+2}, and any real-valued function $u \in \mathcal{L}_2(\Omega_{p+2})$ has an unique expansion $u(\zeta) = \sum_{\ell=0}^{\infty}\sum_{m=1}^{h(\ell,p)} u_{\ell m}S_{\ell m}(\zeta)$, with $\|u\|^2 = \sum_{\ell=0}^{\infty}\sum_{m=1}^{h(\ell,p)} |u_{\ell m}|^2 < \infty$.

Let $\omega_{p+2} \equiv 2\pi^{(1+p/2)}/\Gamma(1+\frac{1}{2}p)$, with Γ the Gamma function, denote the total surface area of Ω_{p+2}. For $p > 0$, let $C_\ell^{p/2}$ denote the Gegenbauer polynomial of degree ℓ and order $\frac{1}{2}p$. The value of $C_\ell^{p/2}(1)$ is given by $C_\ell^{p/2}(1) = (\ell+p-1)!/[\ell!(p-1)!]$. Let $\zeta \cdot \eta$ denote the scalar (dot) product in $\mathbb{R}^{p+2}$ of the unit vectors ζ and η, and let θ denote the angle between ζ and η. The following identity then holds for all $\ell = 0, 1, 2, \ldots$:

$$\frac{\omega_{p+2}}{h(\ell,p)}\sum_{m=1}^{h(\ell,p)} S_{\ell m}(\zeta)S_{\ell m}(\eta) = \left\{ \begin{array}{ll} C_\ell^{p/2}(\zeta\cdot\eta)/C_\ell^{p/2}(1)\,, & \text{for } p>0 \\ \cos\ell\theta\,, & \text{for } p=0 \end{array} \right\}. \tag{2}$$

For $p > 0$, (2) is the well-known Addition Theorem (cf. [3, §11.4]). For $p = 0$, it is a trigonometric identity.

For a given sequence $\{a_\ell\}_{\ell=0}^{\infty}$, we define the vector space (the domain of A)

$$\mathcal{H}_A(\Omega_{p+2}) \equiv \left\{ u \in \mathcal{L}_2(\Omega_{p+2}) : \sum_{\ell=0}^{\infty}\sum_{m=1}^{h(\ell,p)} |a_\ell|^2|u_{\ell m}|^2 < \infty \right\}, \tag{3}$$

and the operator $A : \mathcal{H}_A(\Omega_{p+2}) \to \mathcal{L}_2(\Omega_{p+2})$ by

$$Au(\zeta) \equiv \sum_{\ell=0}^{\infty}\sum_{m=1}^{h(\ell,p)} a_\ell u_{\ell m}S_{\ell m}(\zeta)\,. \tag{4}$$

To solve the interpolation and smoothing problems of the next section, we will apply the general lemmas for curve fitting problems given in Lemmas 3.1 and 5.1 of [9]. We will need to express $\mathcal{H}_A(\Omega_{p+2})$ as a direct sum of two subspaces, where each subspace incorporates the $(p+2)$-dimensional spherical harmonics in its construction. We will also need to display reproducing kernels for $\mathcal{H}_A(\Omega_{p+2})$ and its subspaces, that is, we must show that these spaces are reproducing kernel Hilbert spaces (RKHS).

Let $\mathcal{H}_{A,\ell}(\Omega_{p+2}) \equiv \text{Span}\{S_{\ell m}\}_{m=1}^{h(\ell,p)}$, and let P_ℓ be the orthogonal projection operator that maps $\mathcal{H}_A(\Omega_{p+2})$ onto the eigenspace $\mathcal{H}_{A,\ell}(\Omega_{p+2})$ of A with eigenvalue a_ℓ. Let $\mathbb{N} \equiv \{0, 1, 2, 3, \ldots\}$, and let $\mathbb{N}_0^A \equiv \{\ell \in \mathbb{N} : A(P_\ell u) = 0, \text{ for every } u \in \mathcal{H}_A(\Omega_{p+2})\}$, where $\mathbb{N}_0^A$ is assumed to be a finite set. Then,

$$\mathcal{H}_A(\Omega_{p+2}) = H_A^0(\Omega_{p+2}) \bigoplus \mathcal{H}_A^1(\Omega_{p+2})\,, \tag{5}$$

where

$$\mathcal{H}_A^0(\Omega_{p+2}) \equiv \bigoplus_{\ell\in\mathbb{N}_0^A} \mathcal{H}_{A,\ell}(\Omega_{p+2})\,, \qquad \mathcal{H}_A^1(\Omega_{p+2}) \equiv \bigoplus_{\ell\notin\mathbb{N}_0^A} \mathcal{H}_{A,\ell}(\Omega_{p+2})\,, \tag{6}$$

are, respectively, the null space of A and its orthogonal complement.

Let $u = u_0 + u_1$, and $v = v_0 + v_1$, with $u_0, v_0 \in \mathcal{H}_A^0(\Omega_{p+2})$, and $u_1, v_1 \in \mathcal{H}_A^1(\Omega_{p+2})$. Then $\langle u_0, v_0\rangle_A^0 \equiv \sum_{\ell\in\mathbb{N}_0^A}\sum_{m=1}^{h(\ell,p)}(u_0)_{\ell m}(v_0)_{\ell m}$, $\langle u_1, v_1\rangle_A^1 \equiv \int_{\Omega_{p+2}}[Au_1][Av_1]d\Omega_{p+2} = \sum_{\ell\notin\mathbb{N}_0^A}\sum_{m=1}^{h(\ell,p)}|a_\ell|^2(u_1)_{\ell m}(v_1)_{\ell m}$, and $\langle u, v\rangle_A \equiv \langle u_0, v_0\rangle_A^0 + \langle u_1, v_1\rangle_A^1$ are inner products in $\mathcal{H}_A^0(\Omega_{p+2})$, $\mathcal{H}_A^1(\Omega_{p+2})$, and $\mathcal{H}_A(\Omega_{p_2})$, respectively. It is straightforward to show that $\mathcal{H}_A^0(\Omega_{p+2})$, $\mathcal{H}_A^1(\Omega_{p+2})$, and $\mathcal{H}_A(\Omega_{p+2})$ are Hilbert spaces with respective norms $||u_0||_A^0 \equiv (\langle u_0, u_0\rangle_A^0)^{1/2}$, $||u_1||_A^1 \equiv (\langle u_1, u_1\rangle_A^1)^{1/2}$, and $||u||_A = (\langle u, u\rangle_A)^{1/2}$.

To show that $\mathcal{H}_A^0(\Omega_{p+2})$, $\mathcal{H}_A^1(\Omega_{p+2})$, and $\mathcal{H}_A(\Omega_{p+2})$ are RKHS, we need to find reproducing kernels for these spaces. Before we do this, we make two assumptions.

ASSUMPTION A. *Let $p \in \mathbb{N}$ be fixed, and assume that $\mathcal{H}_A^0(\Omega_{p+2})$ is a finite-dimensional Hilbert space, i.e., $\mathbb{N}_0^A$ is a finite set. For future reference, let $\mathbb{N}_0^A = \{\ell_1, \ell_2, \ldots, \ell_k\}$, and $h_i \equiv h(\ell_i, p)$.*

ASSUMPTION B. *Assume not only that $\sum_{\ell=0}^{\infty}\sum_{m=1}^{h(\ell,p)}|a_\ell|^2|u_{\ell m}|^2 < \infty$, for $u \in \mathcal{H}_A(\Omega_{p+2})$, but also assume that $\sum_{\ell\notin\mathbb{N}_0^A}|a_\ell|^{-2}h(\ell, p) < \infty$. In other words, assume that a_ℓ increases at least as fast as $\ell^{(p+\epsilon+1)/2}$, for some $\epsilon > 0$, as $\ell \to \infty$.*

For $\zeta, \eta \in \Omega_{p+2}$, we define $K_0(\zeta, \eta)$, $K_1(\zeta, \eta)$, and $K(\zeta, \eta)$, respectively, as follows:

$$K_0(\zeta, \eta) \equiv \sum_{i=1}^{k}\sum_{m=1}^{h(\ell,p)} S_{\ell_i m}(\zeta)S_{\ell_i m}(\eta), \tag{7}$$

$$K_1(\zeta, \eta) \equiv \sum_{\ell\notin\mathbb{N}_0^A}\sum_{m=1}^{h(\ell,p)} |a_\ell|^{-2}S_{\ell m}(\zeta)S_{\ell m}(\eta), \tag{8}$$

and

$$K(\zeta, \eta) \equiv K_0(\zeta, \eta) + K_1(\zeta, \eta). \tag{9}$$

LEMMA 2.1. *Let Assumptions* A *and* B *be true. Then K_0, K_1, and K are reproducing kernels for $\mathcal{H}_A^0(\Omega_{p+2})$, $\mathcal{H}_A^1(\Omega_{p+2})$, and $\mathcal{H}_A(\Omega_{p+2})$, respectively, and $\mathcal{H}_A^0(\Omega_{p+2})$, $\mathcal{H}_A^1(\Omega_{p+2})$, and $\mathcal{H}_A(\Omega_{p+2})$ are* RKHS.

Proof. For each $\zeta \in \Omega_{p+2}$, $K_0(\zeta, \eta) \in \mathcal{H}_A^0(\Omega_{p+2})$ as a function of η, and for each $u_0 \in \mathcal{H}_A^0(\Omega_{p+2})$,

$$\langle K_0(\zeta, \cdot), u_0\rangle_A^0 = \sum_{i=1}^{k}\sum_{m=1}^{h(\ell,p)} S_{\ell_i m}(\zeta)(u_0)_{\ell_i m} = u_0(\zeta)\,. \tag{10}$$

Hence, K_0 is a reproducing kernel, and $\mathcal{H}_A^0(\Omega_{p+2})$ is an RKHS. For $K_1(\zeta, \eta)$ as a function of η, we have

$$(\|\ K_1\ \|_A^1)^2 = \sum_{\ell\notin\mathbb{N}_0^A}|a_\ell|^{-2}\sum_{m=1}^{h(\ell,p)}|S_{\ell m}(\zeta)|^2 = \frac{1}{\omega_{p+2}}\sum_{\ell\notin\mathbb{N}_0^A}|a_\ell|^{-2}h(\ell, p), \tag{11}$$

where the last equality follows from (2). Thus, $K_1(\zeta, \eta) \in \mathcal{H}_A^1(\Omega_{p+2})$ as a function of η by Assumption B, and for each $u_1 \in \mathcal{H}_A^1(\Omega_{p+2})$,

$$\langle K_1(\zeta, \cdot), u_1\rangle_A^1 = \sum_{\ell\notin\mathbb{N}_0^A}\sum_{m=1}^{h(\ell,p)}|a_\ell|^2[|a_\ell|^{-2}S_{\ell m}(\zeta)](u_1)_{\ell m} = u_1(\zeta)\,. \tag{12}$$

Therefore, K_1 is a reproducing kernel and $\mathcal{H}_A^1(\Omega_{p+2})$ is an RKHS. That K is a reproducing kernel and $\mathcal{H}_A(\Omega_{p+2})$ is an RKHS then follow from (9), (10), and (12). □

2.2. The interpolation and smoothing problems. In this subsection we will construct hyperspherical spline functions that solve the following interpolation and smoothing problems.

PROBLEM A (An interpolation problem). *Find a* $u \in \mathcal{H}_A(\Omega_{p+2})$ *to minimize*

$$(||u||_A^1)^2 = \int_{\Omega_{p+2}} |Au|^2 d\Omega_{p+2}\,, \tag{13}$$

subject to the constraints

$$u(\zeta_i) = z_i, \quad i = 1, 2, \ldots, n, \tag{14}$$

where A *is the operator given in* (4), $\zeta_i \in \Omega_{p+2}$, *and* $z_i \in \mathbb{R}$, *for* $i = 1, 2, \ldots, n$, *with* $\mathbb{R}$ *denoting the set of real numbers.*

PROBLEM B (A smoothing problem). *Find a* $u \in \mathcal{H}_A(\Omega_{p+2})$ *to minimize*

$$\frac{1}{n}\sum_{i=1}^{n}[u(\zeta_i) - z_i]^2 + \lambda(||u||_A^1)^2\,, \tag{15}$$

where $\lambda > 0$, $\zeta_i \in \Omega_{p+2}$, *and* $z_i \in \mathbb{R}$, *for* $i = 1, 2, \ldots, n$.

2.3. The general theory. Our most general result will need the following assumption.

ASSUMPTION C. *Assume that* $\zeta_1, \zeta_2, \ldots, \zeta_n$ *are* n *distinct data points (knots) on* Ω_{p+2} *chosen so that* (i) $\{K_0(\zeta_j, \eta)\}_{j=1}^n$ *spans* $\mathcal{H}_A^0(\Omega_{p+2})$, *and* (ii) $\{K_1(\zeta_j, \eta)\}_{j=1}^n$ *is a linearly independent set in* $\mathcal{H}_A^1(\Omega_{p+2})$, *where* $n \geq m_0$, *with* $m_0 \equiv \sum_{i=1}^k h(\ell_i, p)$.

THEOREM 2.2. *Let* $\lambda \geq 0$, *let Assumptions* A *and* B *be satisfied, and let* $\{\zeta_1, \zeta_2, \ldots, \zeta_n\}$ *satisfy Assumption* C. *Let* $[a_{ij}]$ *denote the matrix with elements* a_{ij}, *and let* $'$ *denote the transpose of a matrix or vector. Let* $K_{n,\lambda}$ *be the symmetric* $n \times n$ *matrix defined by*

$$K_{n,\lambda} \equiv [K_1(\zeta_i, \zeta_j)] + n\lambda I, \tag{16}$$

where I *is the identity matrix. Let* S_0 *be the* $n \times m_0$ *matrix defined by*

$$S_0 \equiv [S_{\ell_i m}(\zeta_j)]', \tag{17}$$

for $i = 1, 2, \ldots, k$, $m = 1, 2, \ldots, h_i$, *and* $j = 1, 2, \ldots, n \geq m_0$. *Define*

$$\vec{z} \equiv (z_1, z_2, \ldots, z_n)', \tag{18}$$

$$\vec{s}\,(\eta) \equiv (S_{\ell_1 1}(\eta), \ldots, S_{\ell_1 h_1}(\eta), S_{\ell_2 1}(\eta), \ldots, S_{\ell_2 h_2}(\eta), \ldots, S_{\ell_k 1}(\eta), \ldots, S_{\ell_k h_k}(\eta))', \tag{19}$$

and

$$\vec{k}(\eta) \equiv (K_1(\zeta_1, \eta), K_1(\zeta_2, \eta), \ldots, K_1(\zeta_n, \eta))', \tag{20}$$

where $z_j \in \mathbb{R}$. *Let* $u_{A,n,\lambda}$ *be defined by*

$$u_{A,n,\lambda}(\eta) \equiv \vec{k}(\eta) \cdot \vec{c} + \vec{s}\,(\eta) \cdot \vec{d}\,, \tag{21}$$

where the n*-dimensional vector* $\vec{c}$ *and the* m_0*-dimensional vector* $\vec{d}$ *are defined by*

$$\vec{c} \equiv K_{n,\lambda}^{-1}(I - S_0 X)\vec{z}\,, \qquad \vec{d} \equiv X\vec{z}\,, \tag{22}$$

with X the $m_0 \times n$ matrix

$$X \equiv (S_0' K_{n,\lambda}^{-1} S_0)^{-1} S_0' K_{n,\lambda}^{-1} . \tag{23}$$

Then $u_{A,n,0}(\eta)$ is the solution for Problem A, *and $u_{A,n,\lambda}(\eta)$ for $\lambda > 0$ is the solution for Problem* B. *Furthermore,* (22) *for $\vec{c}$ and $\vec{d}$ are equivalent to the following system of equations:*

$$K_{n,\lambda}\vec{c} + S_0\vec{d} = \vec{z} \quad \text{and} \quad S_0'\vec{c} = \vec{0} . \tag{24}$$

Proof. Assumption C(ii) implies that the $n \times n$ matrix $[K_1(\zeta_j, \zeta_i)]$ is positive definite and invertible (cf. [2, p. 13]). It follows that $K_{n,\lambda}$ is also an invertible $n \times n$ matrix for all $\lambda \geq 0$. We also note that for $\lambda > 0$, the $n \times n$ matrix $\frac{1}{n\lambda} I$ is positive definite. Assumption C(i) and (7) imply that the range of S_0 is $\mathcal{H}_A^0(\Omega_{p+2})$, and, therefore, the rank of the matrix S_0 is m_0. Let P_0 and P_1 be the orthogonal projections of $\mathcal{H}_A(\Omega_{p+2})$ onto $\mathcal{H}_A^0(\Omega_{p+2})$ and $\mathcal{H}_A^1(\Omega_{p+2})$, respectively. For $j = 1, 2, \ldots, n \geq m_0$, we then have that $P_0 K(\zeta_j, \eta) = K_0(\zeta_j, \eta)$ and $P_1 K(\zeta_j, \eta) = K_1(\zeta_j, \eta)$.

For $\lambda > 0$, we now apply [9, Lem. 5.1] by, respectively, identifying $K_1(\zeta_i, \cdot)$, $\vec{s}$, $\vec{k}$, $K_{n,\lambda}$, S_0', and $\frac{1}{n\lambda} I$ with ψ_i, $\vec{\omega}$, $\vec{\eta}$, M, U, and S in [9]. The result is that the solution $u_{A,n,\lambda}$ (u^{Δ} in [9]) of Problem B is given by (21) with $\vec{c}$ and $\vec{d}$ defined by (22) and (23). For $\lambda = 0$, [9, Lem. 3.1] similarly implies that $u_{A,n,0}$ is the solution for Problem A.

If $\vec{c}$ and $\vec{d}$ defined by (22) and (23) are substituted into (24), then these equations are easily seen to be satisfied. Conversely, let $\vec{c}$ and $\vec{d}$ be defined by (24). The first equation in (24) gives

$$\vec{c} = K_{n,\lambda}^{-1}(\vec{z} - S_0\vec{d}) . \tag{25}$$

Substitution of (25) into the second equation in (24) yields $\vec{d} = X\vec{z}$, and this equation combined with (25) gives (22). □

In practice, (24) provides a numerically more efficient algorithm for computing $\vec{c}$ and $\vec{d}$ than do (22) and (23), and in all of our computations we have used (24). We now consider the possibility of removing Assumptions A–C from Theorem 2.2. Toward this end we present the following lemma and theorem.

LEMMA 2.3. *Assume that Assumptions* A *and* B *are true. In addition, suppose there is a positive constant c and a positive integer r such that $|a_\ell| \leq c\lambda_\ell^r$, for each $\ell = 0, 1, \ldots$, where $\lambda_\ell \equiv \ell(\ell + p)$. Then $\{K(\zeta_j, \eta)\}_{j=1}^n$ is a linearly independent set in $\mathcal{H}_A(\Omega_{p+2})$ for all distinct $\zeta_1, \zeta_2, \ldots, \zeta_n \in \Omega_{p+2}$.*

Proof. Consider the function ϕ defined on $\mathbb{R}^{p+2}$ by

$$\phi(\vec{x}) \equiv \left\{ \begin{array}{ll} \exp\left(-|\vec{x}|^2/[1 - |\vec{x}|^2]\right), & \text{for } |\vec{x}| < 1 \\ 0, & \text{for } |\vec{x}| \geq 1 \end{array} \right\} , \tag{26}$$

and let $u_i(\eta) \equiv \phi([\eta - \zeta_i]/\varepsilon)$, for $i = 1, 2, \ldots, n$, and some ε in the interval $(0, \varepsilon_0]$, with $\varepsilon_0 \equiv \min_{i \neq j} |\zeta_i - \zeta_j|/2$, $i, j = 1, 2, \ldots, n$. Then u_i is an infinitely differentiable function in $\mathcal{L}_2(\Omega_{p+2})$ that vanishes outside a ball of radius ε with center at ζ_i and satisfies $u_i(\zeta_j) = \delta_{ij}$, where δ_{ij} denotes the Kronecker delta. Furthermore, $Au_i \in \mathcal{L}_2(\Omega_{p+2})$ since

$$(\|Au_i\|_{\mathcal{L}_2})^2 = \sum_{\ell=0}^{\infty} \sum_{m=1}^{h(\ell,p)} |a_\ell|^2 |(u_i)_{\ell m}|^2 \leq c^2 \|(-L_{p+2})^r u_i\|_{\mathcal{L}_2}^2 < \infty , \tag{27}$$

where the last inequality holds because u_i is infinitely differentiable. It follows that $u_i \in \mathcal{H}_A(\Omega_{p+2})$. Taking the $\mathcal{H}_A(\Omega_{p+2})$ inner product of $\sum_{j=1}^n \alpha_j K(\zeta_j, \eta) = 0$ with u_i gives

$$0 = \sum_{j=1}^n \alpha_j \langle K(\zeta_j, \cdot), u_i \rangle_A = \sum_{j=1}^n \alpha_j u_i(\zeta_j) = \alpha_i \,. \tag{28}$$

This proves the lemma. □

THEOREM 2.4. *Let $p \in \mathbb{N}$ be fixed, and suppose that $|a_\ell| = c[\ell(\ell + p)]^{\nu/2}$ for some constants c and ν satisfying $c > 0$, and $\nu > (p + 1)/2$, and for all $\ell = 0, 1, 2, \ldots$. Then Assumptions* A *and* B *are satisfied, and Assumption* C *is satisfied whenever the knots ζ_j, $j = 1, 2, \ldots, n$, are distinct.*

Proof. Since $\mathbb{N}_0^A = \{0\}$, $\mathcal{H}_A^0(\Omega_{p+2})$ is the one-dimensional space of constant functions. Also, $m_0 = 1$, and $K_0(\zeta_j, \eta) = 1$, for all $j = 1, 2, \ldots, n$. Assumptions A and C(i) are obviously satisfied. Assumption B is satisfied since $\nu > (p+1)/2$ implies $\sum_{\ell \neq 0} |a_\ell|^{-2} h(\ell, p) < \infty$. Lemma 2.3 is satisfied since $|a_\ell| = c\lambda_\ell^{\nu/2} \leq c\lambda_\ell^r$ for any integer $r \geq \nu/2$.

Now suppose that $\sum_{j=1}^n \alpha_j K_1(\zeta_j, \eta) = 0$. Then

$$\sum_{j=1}^n \alpha_j K(\zeta_j, \eta) = \sum_{j=1}^n \alpha_j K_0(\zeta_j, \eta) + \sum_{j=1}^n \alpha_j K_1(\zeta_j, \eta) = a_n, \tag{29}$$

where $a_n \equiv \sum_{j=1}^n \alpha_j$. Using (28) of Lemma 2.3, the $\mathcal{H}_A(\Omega_{p+2})$ inner product of (29) with the vector u_i defined in Lemma 2.3 gives

$$\alpha_i = a_n \langle 1, u_i \rangle_A, \tag{30}$$

for all $i = 1, 2, \ldots, n$. Summing this equation over i yields

$$\left[1 - \sum_{i=1}^n \langle 1, u_i \rangle_A\right] a_n = 0. \tag{31}$$

Now consider the integrals $\langle 1, u_i \rangle_A = \int u_i d\Omega_{p+2}$. Let θ denote the angle between η and ζ_i. Then, $d\Omega_{p+2} = \sin^p \theta d\theta d\Omega_{p+1}$ (cf. [3, p. 233]), and $\sin(\theta/2) = |\eta - \zeta_i|/2$. Consequently,

$$\langle 1, u_i \rangle_A = \omega_{p+1} \int_0^{2 \arcsin(\varepsilon/2)} \sin^p \theta \exp\left\{\frac{-\sin^2(\theta/2)}{(\varepsilon/2)^2 - \sin^2(\theta/2)}\right\}, \tag{32}$$

for all $i = 1, 2, \ldots, n$. Here we define $d\Omega_1 \equiv 1$, and $\omega_1 \equiv 1$, when $p = 0$. It follows that

$$\left|\sum_{i=1}^n \langle 1, u_i \rangle_A\right| \leq 2n\omega_{p+1} \arcsin\frac{\varepsilon}{2}, \tag{33}$$

which can be made less than 1 by choosing ε sufficiently small. For such a choice of ε the coefficient of a_n in (31) is nonzero, with the implication that $a_n = 0$. However, then the right-hand side of (29) is zero, and $\alpha_i = 0$, $i = 1, 2, \ldots, n$, follows from Lemma 2.3. This proves Assumption C(ii) and the theorem. □

Remark 1. For the case $p = 0$ and $a_\ell \equiv \ell^\nu$, $\nu \geq 1$, the solution $u_{A,n,\lambda}(\eta)$, with $\lambda > 0$, given by Theorems 2.2 and 2.4 coincides with that previously obtained for Problem B by Wahba [12, (2.2)].

Remark 2. In [13] Wahba considers the case $p = 1$ and $a_\ell \equiv [\ell(\ell+1)]^{\nu/2}$, $\nu \geq 2$. Since this a_ℓ satisfies Theorem 2.4, we see that [13, Thm 1] is a special case of our Theorems 2.2 and 2.4.

Remark 3. In general, the linear independence of the set $\{K(\zeta_j, \eta)\}_{j=1}^n$ does *not* imply Assumption C(i). Since Assumption C(i) is true if and only if the rank of the matrix S_0 is m_0, it follows from a simple modification of the proof of Lemma 6 of [10] that *there exist* knots $\zeta_1, \zeta_2, \ldots, \zeta_n$ on Ω_{p+2} for which Assumption C(i) is satisfied. Clearly, the set of knots ζ_j, $j = 1, 2, \ldots, n$, must be sufficiently large and carefully chosen so that the rank of the matrix S_0 is m_0. In particular, if $m_0 > 1$, then ζ_j should not be chosen exactly on the poles or equators of the spheres.

Remark 4. In this paper we have taken $\mathcal{H}_A(\Omega_{p+2})$ to be a real Hilbert space so that we may invoke the lemmas in [9]. Applications to quantum mechanics would require a complex Hilbert space version of the theory, and we believe that the modifications necessary to establish a complex Hilbert space version are straightforward.

Unfortunately for computational purposes, the reproducing kernel $K_1(\zeta, \eta)$ defined in (8) is in the form of an infinite series. The only case that we know this series to be summable is when $p = 0$ and $a_\ell = \ell^\nu$, with ν a positive integer. In this case (2) and (8) give

$$K_1(\theta) \equiv K_1(\zeta, \eta) = \frac{1}{2\pi} \sum_{\ell=1}^{\infty} \frac{2\cos \ell\theta}{\ell^{2\nu}} , \tag{34}$$

where θ is the angle between ζ and η, and this infinite series sums to the periodic extension of

$$K_1(\theta) = \frac{(-1)^{\nu-1}(2\pi)^{2\nu-1}}{(2\nu)!} B_{2\nu}(\theta/2\pi) , \qquad 0 \le \theta < 2\pi , \tag{35}$$

where $B_{2\nu}$ are Bernoulli polynomials of degree 2ν (cf. [7], [12] and [8, (9.622)]). The solution $u_{A,n,\lambda}(\eta)$ given by (21) is then the well-known periodic interpolating ($\lambda = 0$) or smoothing ($\lambda > 0$) polynomial spline of degree $2\nu - 1$ and continuity class $C^{2\nu-2}[0, 2\pi]$. The vanishing of the coefficient of $\theta^{2\nu}$ is a consequence of the second equation in (24). For example, when $\nu = 1$, (35) is

$$K_1(\theta) = \frac{(\theta - \pi)^2}{4\pi} - \frac{\pi}{12} , \qquad 0 \le \theta < 2\pi , \tag{36}$$

with $K_1(\theta \pm 2k\pi) \equiv K_1(\theta)$, for $k = 1, 2, \ldots$, and $u_{A,n,0}(\phi)$ are the continuous piecewise linear interpolating polynomials. In other words, $u_{A,n,0}(\phi)$ is the solution of the child's game of connecting n dots by straight lines on a piece of paper and enforcing periodicity at the ends. In the general case, Theorem 2.2 gives the corresponding solution to either Problem A or Problem B for data defined on the surface of a hypersphere of arbitrary dimension, but with the proviso that the minimum continuity of the spline solution must satisfy $\nu > (p + 1)/2$.

For $p > 0$, it appears that no closed-form expression for the series in (8) exists that is convenient for computations. In the next subsection we will define modified reproducing kernels $R_1(\zeta, \eta)$ in a manner analogous to that of [13]. These new kernels will have closed form expressions, with the consequence that numerical computations are much faster.

2.4. The modified theory. In generalizing Wahba's modified spline theory to higher dimensions, we consider only the case $p \in \mathbb{N}$ with $p > 0$, and

$$a_\ell \equiv [\ell(\ell + p)]^{\nu/2} , \tag{37}$$

for all $\ell = 0, 1, 2, \ldots$, where ν is an integer or half integer satisfying $\nu > (p + 1)/2$. Theorem 2.4 holds in this case. The RKHS $\mathcal{H}_A^0(\Omega_{p+2})$ is $\mathcal{H}_A^0(\Omega_{p+2}) = \{1\}$. When ν is an integer the RKHS $\mathcal{H}_A^1(\Omega_{p+2})$ is equal to $\{u \in \mathcal{H}_A(\Omega_{p+2}) : u(\zeta) = \sum_{\ell=1}^{\infty} \sum_{m=1}^{h(\ell,p)} (u)_{\ell m} S_{\ell m}(\zeta)$, with $u, u^{(1)}, u^{(2)}, \ldots, u^{(\nu-1)}$ absolutely continuous$\}$, where $u^{(k)}$ denotes a kth-order partial

derivative. By (8) and (2), the reproducing kernel $K_1(\zeta, \eta)$ with respect to the norm $\|\cdot\|_A^1$ becomes

$$K_1(\zeta, \eta) = \frac{1}{p\omega_{p+2}} \sum_{\ell=1}^{\infty} \frac{2\ell + p}{[\ell(\ell + p)]^{\nu}} C_\ell^{(p/2)}(\zeta \cdot \eta) . \tag{38}$$

To construct the analogue of Wahba's modified splines in higher dimensions, we define for $u_1 \in \mathcal{H}_A^1(\Omega_{p+2})$,

$$\|u_1\|_B^1 \equiv \left[\sum_{\ell=1}^{\infty} \sum_{m=1}^{h(\ell,p)} |g_\ell|^2 |(u_1)_{\ell m}|^2 \right]^{1/2} , \tag{39}$$

where

$$g_\ell \equiv \left[\left(\ell + \frac{p}{2} \right) (\ell + 1)(\ell + 2) \cdots (\ell + 2\nu - 1) \right]^{1/2} , \tag{40}$$

with 2ν any integer satisfying $2\nu > p + 1$. It is not difficult to show [11] that the new norm $\|\cdot\|_B^1$ is topologically equivalent to $\|\cdot\|_A^1$ in the sense that there exist positive real numbers a and b, $a < b$, such that

$$a\|u_1\|_A^1 \le \|u_1\|_B^1 \le b\|u_1\|_A^1 . \tag{41}$$

A reproducing kernel for $\mathcal{H}_A^1(\Omega_{p+2})$ with respect to the norm $\|\cdot\|_B^1$ is

$$\begin{aligned} R_1(\zeta, \eta) &\equiv \sum_{\ell=1}^{\infty} \sum_{m=1}^{h(\ell,p)} |g_\ell|^{-2} S_{\ell m}(\zeta) S_{\ell m}(\eta) \\ &= \frac{2}{pw_{p+2}} \sum_{\ell=1}^{\infty} \frac{C_\ell^{(p/2)}(\zeta \cdot \eta)}{(\ell + 1)(\ell + 2) \cdots (\ell + 2\nu - 1)} , \end{aligned} \tag{42}$$

where the last equality is a consequence of (2). It follows that $\mathcal{H}_A^1(\Omega_{p+2})$ is a RKHS with respect to the norm $\|\cdot\|_B^1$.

We note that R_1 is a function of only $z \equiv \zeta \cdot \eta$. To express $R_1(\zeta, \eta)$ in a closed form that is convenient for computations, we apply the following two equations:

$$\frac{1}{r!} \int_0^1 (1 - h)^r h^\ell dh = \frac{1}{(\ell + 1)(\ell + 2) \cdots (\ell + r + 1)} , \tag{43}$$

for $r = 0, 1, 2, \ldots$, and

$$\sum_{\ell=1}^{\infty} h^\ell C_\ell^{p/2}(z) = (1 - 2hz + h^2)^{-p/2} - 1 , \tag{44}$$

for $|h| < 1$. Equation (43) is just the repeated application of [8, (2.151)], and (44) is taken from [1, Table 22.9.3]. Applying (43) and (44) to (42), $R_1(\zeta, \eta)$ can be expressed as

$$R_1(\zeta, \eta) = \frac{2}{pw_{p+2}} \left[\frac{q(2\nu - 2, p, \zeta \cdot \eta)}{(2\nu - 2)!} - \frac{1}{(2\nu - 1)!} \right] , \tag{45}$$

where

$$q(r, p, z) \equiv \int_0^1 (1 - h)^r (1 - 2hz + h^2)^{-p/2} dh. \tag{46}$$

TABLE 1
Values of the integrals $q(r, p, z)$ for selected values of r and p.

$q(1,1,z)$	$-S+L+1$
$q(2,1,z)$	$[-L(3z-1)+3S(z-1)+4-3z]/2$
$q(3,1,z)$	$[(15z^2-27z+14)-S(z-1)(15z-7)+3L(z-1)(5z+1)]/6$
$q(4,1,z)$	$[(-105z^3+240z^2-161z+32)-3L(z-1)(35z^2-10z-13)+5S(z-1)^2(21z+1)]/24$
$q(2,2,z)$	$-2Tz-N+1$
$q(3,2,z)$	$2T(z-1)(2z+1)+N(2z-1)+(5-4z)/2$
$q(4,2,z)$	$-4T(z-1)(2z^2-1)-4N(z-1)z+(12z^2-21z+10)/3$
$q(5,2,z)$	$4T(z-1)^2(4z^2+2z-1)+2N(z-1)(4z^2-2z-1)-[(4z-1)(24z^2-48z+23)]/12$
$q(4,4,z)$	$-2N+[-2T(2z^2+2z-1)+2z]/(z+1)$
$q(5,4,z)$	$2N(3z-2)+[2T(z-1)(6z^2+8z+1)+(-12z^2+5z+9)]/[2(z+1)]$
$q(6,4,z)$	$[(48z^3-48z^2-35z+37)-12T(z-1)(8z^3+6z^2-6z-3)]/[3(z+1)]$ $-4N(z-1)(4z-1)$
$q(7,4,z)$	$[(-480z^4+744z^3+176z^2-645+211)+48T(z-1)^2(20z^3+24z^2-3z-6)]/[12(z+1)]$ $+2N(z-1)(20z^2-16z-1)$

In Table 1 we display the values of the integrals $q(r, p, z)$ for specific values of r and p. The letters S, N, L, and T represent the following functions of z:

$$S = \sqrt{2-2z}\,, \tag{47}$$

$$N = (1-z)\log(2-2z), \tag{48}$$

$$L = (1-z)\log\left(\frac{\sqrt{2}}{\sqrt{1-z}}+1\right), \tag{49}$$

$$T = \left(\frac{\sqrt{1-z}}{\sqrt{1+z}}\right)\arctan\frac{\sqrt{1+z}}{\sqrt{1-z}}\,. \tag{50}$$

The closed-form expressions in Table 1 were generated by a MACSYMA symbolic manipulation program. We note that the formulas for $p = 1$ are the same as those in [13].

We summarize the results of this subsection in the following theorem.

THEOREM 2.5. *Let a_ℓ be defined by* (37), *and let $\zeta_j \in \Omega_{p+2}$, $j = 1, 2, \ldots, n$, be distinct. Then, the solution $u_{B,n,0}(\eta)$ to Problem* A *with $(||u||_A^1)^2$ replaced by $(||u||_B^1)^2$, and the solution $u_{B,n,\lambda}(\eta)$, $\lambda > 0$, to Problem* B *with $(||u||_A^1)^2$ replaced by $(||u||_B^1)^2$, is given by*

$$u_{B,n,\lambda}(\eta) \equiv \vec{r}(\eta)\cdot\vec{c} + \vec{s}\,(\eta)\cdot\vec{d}\,, \tag{51}$$

where $\lambda \geq 0$, and $\vec{c}$ and $\vec{d}$ are determined by the system of equations

$$R_{n,\lambda}\vec{c} + S_0\vec{d} = \vec{z}\,, \qquad S_0'\vec{c} = \vec{0}\,, \tag{52}$$

with S_0, $\vec{z}$, and $\vec{s}$ defined by (17)–(19), *respectively, and $R_{n,\lambda}$ and $\vec{r}$ defined, respectively, by* (16) *and* (20) *with K_1 replaced by R_1, and $R_1(\zeta, \eta)$ defined by* (45) *and* (46).

In the next section we will use Theorems 2.2 and 2.5 to fit functions defined on Ω_{p+2}, for $p = 1$, 2, and 4.

3. Numerical computations.

3.1. Methods used for computations. In most of our computations we have chosen the a_ℓ in (3) and (4) to be of the form of (37). However, in a few computations (cf. Tables 9 and 10) we have also used the form

$$a_\ell = [\ell(\ell+p)(\ell-1)(\ell-2)\cdots(\ell-L_s)]^{\nu/2}, \tag{53}$$

with $L_s \geq 1$, where L_s, is the highest-order hyperspherical harmonic used in the computation. Then $\mathbb{N}_0^A = \{0, 1, \ldots, L_s\}$. In this case the hypotheses of Lemma 2.3 are satisfied, but the hypotheses of Theorem 2.4 are not satisfied. We have tried various values of ν and have found in most cases that the best results are obtained with $\nu = 1 + (p+1)/2$.

Our tests showed that $K_1(\zeta, \eta)$ could be approximated to at least ten significant digits of accuracy by truncating the infinite sum defining $K_1(\zeta, \eta)$ (cf. (38) or (2) and (8)) to 40–50 terms. The method used to compute the coefficients in the truncation of $K_1(\zeta, \eta)$ uses the recurrence relations arising in the telescoping orthogonal expansions for the Gegenbauer polynomials (cf. [4], [11]). The computation of the modified reproducing kernel $R_1(\zeta, \eta)$ was easily done once $q(r, p, z)$ given by (46) was computed. Selected values of $q(r, p, z)$ for $p = 1, 2$, and 4 appear in Table 1.

The basic problem is to approximate a given function $u(\eta)$ defined on Ω_{p+2} using hyperspherical harmonics and/or hyperspherical splines. Using $m_0 \equiv \sum_{\ell=0}^{L_s} h(\ell, p)$ hyperspherical harmonics and $n \geq m_0$ hyperspherical splines (i.e., n data points), the problem is to find coefficients $d_{\ell m}$ and c_j such that

$$u(\eta) \simeq \sum_{\ell=0}^{L_s} \sum_{m=1}^{h(\ell,p)} d_{\ell m} S_{\ell m}(\eta) + \sum_{j=1}^{n} c_j W(\zeta_j, \eta), \tag{54}$$

where $W(\zeta_j, \eta)$ denotes either $K_1(\zeta_j, \eta)$ or $R_1(\zeta_j, \eta)$, and ζ_j for $j = 1, 2, \ldots, n$ are distinct knots on Ω_{p+2}. The approximation $\simeq$ in (54) may be in the sense of either Problem A or Problem B. By Theorem 2.2 or 2.5 the solution is obtained by solving the system of equations

$$W_{n,\lambda}\vec{c} + S_0\vec{d} = \vec{z} \quad \text{and} \quad S_0'\vec{c} = \vec{0}, \tag{55}$$

where $S_0 \equiv [S_{\ell m}(\zeta_j)]'$, $\vec{z} \equiv (z_1, z_2, \ldots, z_n)'$, with $z_i \in \mathbb{R}$, and $W_{n,\lambda} \equiv [W(\zeta_j, \zeta_i)] + n\lambda I$, with $\lambda = 0$ for Problem A, and $\lambda > 0$ for Problem B. The linear algebraic systems in (55) were solved using LINPACK subroutines.

For comparison purposes, we have also considered the problem of approximating $u(\eta)$ using only m_0 hyperspherical harmonics, that is, with $c_j = 0$, $j = 1, 2, \ldots, n$, in (54). In this case we chose $n \geq m_0$ collocation points η_i, $i = 1, 2, \ldots, n$, on Ω_{p+2} in such a way that the rank of the $n \times m_0$ matrix $S_0 \equiv [S_{\ell m}(\eta_i)]'$ was m_0. The system of equations $S_0\vec{d} = \vec{z}$, where $\vec{z}$ is the vector with components $u(\eta_i)$, was then solved using LINPACK subroutines.

The maximum and average absolute errors were evaluated over a grid obtained by dividing ϕ into n_ϕ equal subintervals, and each of the angles θ_i, $i = 1, 2, \ldots, p$, into n_θ equal subintervals, and then counting all coinciding points only once. For $p = 1$ we used $n_\phi = 21$ and $n_\theta = 27$. For $p = 2$ we used $n_\phi = n_\theta = 19$, and for $p = 4$ we used $n_\phi = 8$ and $n_\theta = 7$. Our tests showed that increasing the number of grid points did not significantly affect the results.

When $L_s = 0$, the most successful procedure that we have found in the selection of knots on Ω_{p+2} is to first solve (55) using the set of knots suggested by Abramowitz and Stegun ([1, p. 894]). In particular, the initial set of knots used for $p = 1$ are either $\{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1)\}$ or $\{(\pm 1, 0, 0), (0, \pm 1, 0), (0, 0, \pm 1), (\pm\sqrt{1/2}, \pm\sqrt{1/2}, 0), (\pm\sqrt{1/2}, 0, \pm\sqrt{1/2}),$

$(0, \pm\sqrt{1/2}, \pm\sqrt{1/2})\}$. Analogous initial sets of knots were used for $p > 1$. After fitting is done using these knots (A & S knots), a search over the grid points is made for the maximum absolute error and its coordinates on Ω_{p+2}. A new knot is then added at the place where the maximum absolute error occurs, and the procedure is repeated until a reasonable result is obtained. Our numerical experience showed that shifting the initial knots or changing their number did not significantly affect the eventual results.

The procedure of first using the A & S knots and then adding the knot where the maximum absolute error occurs repeatedly until a reasonable result is obtained did not do so well when $L_s > 0$. The matrix S_0 quickly became ill-conditioned using this method. Much better results were obtained by shifting the initial A & S knots as indicated in Remark 3.

3.2. Examples used for computations. In testing the hyperspherical spline approximations and comparing them with hyperspherical harmonics, we have used functions of three basic types. For $\zeta \in \Omega_{p+2}$, $p = 1, 2, 4$, the first two functions F_1 and F_2 were defined, respectively, by

$$F_1(\zeta) \equiv \frac{1}{2 + \cos(\zeta \cdot a_1)}, \qquad F_2(\zeta) \equiv \sum_{j=1}^{5} \frac{1}{2 + \zeta \cdot a_j}, \tag{56}$$

where $a_j \in \Omega_{p+2}$, $j = 1, 2, 3, 4, 5$, are arbitrarily chosen distinct fixed vectors [11]. The third function G was defined for $(\vec{\theta}, \phi) \in \Omega_{p+2}$ by

$$G(\vec{\theta}, \phi) \equiv \frac{2}{2 + p^{-1} \sum_{i=1}^{p} \theta_i + \phi \sin(\phi/2) \prod_{i=1}^{p} \sin\theta_i}. \tag{57}$$

All three functions are positive. In particular, the ranges are $0.333 < F_1(\zeta) \leq 1$, $1.666 < F_2(\zeta) \leq 5$, and $0.272 < G(\vec{\theta}, \phi) \leq 1$, for $p = 1, 2, 4$. Functions F_1 and F_2 have continuous partial derivatives of all orders. The function F_1 depends only on the angle θ between ζ and a_1, and is, therefore, symmetric under rotations about the axis defined by a_1. The function F_2 does not have this symmetry. The function G is continuous but has discontinuous directional derivatives at the poles. It does not have any symmetry. When $p = 1$, we have taken the constant vectors $a_j = (\theta_j, \phi_j)$ to be $a_1 = (0.3333\pi, 0.1667\pi)$, $a_2 = (0.1234\pi, 0.9876\pi)$, $a_3 = (0.9999\pi, 0.0054\pi)$, $a_4 = (0.5134\pi, 0.4988\pi)$, and $a_5 = (0.7500\pi, 1.7890\pi)$, where θ and ϕ are the usual polar angles.

3.3. Summary of results. We have compared the maximum absolute errors and the average absolute errors when solving Problem A using the original hyperspherical splines K_1 versus the modified splines R_1 for $p = 1, 2, 4$. As a general rule, we have found that the errors using the R_1 splines vary from one to three times those obtained using the K_1 splines. One of the poorest showings for the modified splines was for the function F_1 with $p = 1$, and this case is shown in Table 2. For the function G with $p = 1$, the R_1 splines performed almost as well as the K_1 splines. Our computations using the K_1 splines took approximately 30 times as much computing time as those using the R_1 splines, and, with the exception of Tables 9 and 10, we have used the R_1 splines for the remainder of the calculations reported in this paper.

In Tables 3–5 we compare the Problem A errors obtained using $N = m_0$ spherical harmonics versus $N = n$ modified hyperspherical splines for the functions F_1, F_2, and G, respectively. We note that the spherical harmonics did exceptionally well for function F_1, which is rotationally symmetric about the line a_1. This is not surprising because this problem is equivalent to fitting with respect to the variable $z \equiv \langle \zeta, a_1 \rangle$ using Legendre polynomials. To a lesser extent, spherical harmonics seem also to be sensitive to the smoothness of the function being fit, and indeed the fit for the function F_2 is better than that for the function G.

On the other hand, the spline fits do not seem to be affected so much by the smoothness or symmetry properties of the functions being fit. Using 50 R_1 splines yields approximations for each function that have a maximum (average) absolute error within 0.02 (0.01). For the functions F_1 and F_2 the maximum (average) absolute errors are within 1.2% (0.5%) of the minimum values of the corresponding functions, and for G the maximum (average) absolute error is within 6.4% (3.4%) of the minimum value of G.

TABLE 2

A comparison of maximum absolute errors and average absolute errors when fitting function F_1 using n K_1 splines versus using n R_1 splines with $v = 2.0$ for $p = 1$.

	K_1 Splines		R_1 Splines	
n	Max. abs. err.	Ave. abs. err.	Max. abs. err.	Ave. abs. err.
10	0.0162	0.0065	0.0192	0.0083
20	0.0064	0.0021	0.0103	0.0039
30	0.0030	0.0012	0.0071	0.0027
40	0.0023	0.0008	0.0043	0.0018
50	0.0016	0.0006	0.0032	0.0013

TABLE 3

A comparison of maximum absolute errors and average absolute errors for spherical harmonic fits versus R_1 spline fits with $v = 2.0$ using function F_1 for $p = 1$. ($N = m_0$ = number of spherical harmonics or $N = n$ = number of hyperspherical splines.)

	Maximum absolute error		Average absolute error	
N	Sph. harm.	R_1 Splines	Sph. harm.	R_1 Splines
10	0.0022	0.0192	0.0006	0.0083
20	0.0013	0.0103	0.0002	0.0039
30	0.4403E–04	0.0071	0.1056E–04	0.0027
50	0.6239E–05	0.0032	0.6034E–06	0.0013

TABLE 4

A comparison of maximum absolute errors and average absolute errors for spherical harmonic fits versus R_1 spline fits with $v = 2.0$ using function F_2 for $p = 1$.

	Maximum absolute error		Average absolute error	
N	Sph. harm.	R_1 Splines	Sph. harm.	R_1 Splines
10	0.1906	0.1450	0.0525	0.0488
20	0.1554	0.0636	0.0356	0.0255
30	0.0422	0.0398	0.0099	0.0165
50	0.0048	0.0193	0.0005	0.0081

Tables 6 and 7 give the corresponding comparison for the function F_2 in four-dimensional ($p = 2$) and six-dimensional ($p = 4$) space, respectively. We note that the hyperspherical splines are now out performing the spherical harmonics even for the infinitely differentiable function F_2, and the higher the dimension of the space the more the advantage. As expected, however, the absolute errors increase as the value of p increases. In Table 8 we see that hyperspherical splines are yielding reasonable results for the more difficult function G when $p = 4$.

Finally, we have numerically tested the possibility of combining hyperspherical harmonics with hyperspherical splines. In particular, we have used the a_ℓ defined in (53) for $L_s = 1$ and 2, and $p = 1$ and 2. Tables 9 and 10 show our $p = 1$ Problem A computations for the functions F_2 and G, respectively. In these computations we used n of Wahba's original K_1 splines plus the appropriate $S_{\ell m}$ spherical harmonics corresponding to $L_s = 0, 1, 2$. These computations using K_1 splines were very time consuming and not as stable as our previous R_1 spline computations. However, we see that the addition of the three $\ell = 1$, and to a lesser extent the five $\ell = 2$, spherical harmonics appears to be beneficial for

TABLE 5

A comparison of maximum absolute errors and average absolute errors for spherical harmonic fits versus R_1 spline fits with $\nu = 1.5$ using function G for $p = 1$.

	Maximum absolute error		Average absolute error	
N	Sph. harm.	R_1 Splines	Sph. harm.	R_1 Splines
15	0.3306	0.0609	0.0763	0.0285
22	0.2686	0.0401	0.0652	0.0180
30	0.3953	0.0307	0.0856	0.0132
51	0.3460	0.0173	0.0472	0.0091

TABLE 6

A comparison of maximum absolute errors and average absolute errors for hyperspherical harmonic fits versus R_1 hyperspherical spline fits with $\nu = 2.5$ using function F_2 for $p = 2$.

	Maximum absolute error		Average absolute error	
N	Sph. harm.	R_1 Splines	Sph. harm.	R_1 Splines
12	0.4581	0.1145	0.1359	0.0310
22	0.2821	0.0471	0.0476	0.0158
30	0.4632	0.0375	0.1022	0.0126
40	0.0462	0.0273	0.0092	0.0089
50	0.0546	0.0176	0.0107	0.0070
58	0.0381	0.0155	0.0070	0.0054

TABLE 7

A comparison of maximum absolute errors and average absolute errors for hyperspherical harmonic fits versus R_1 hyperspherical spline fits with $\nu = 3.5$ using function F_2 for $p = 4$.

	Maximum absolute error		Average absolute error	
N	Sph. harm.	R_1 Splines	Sph. harm.	R_1 Splines
15	0.5810	0.1528	0.1521	0.0458
20	0.5327	0.1008	0.1178	0.0350
30	0.4637	0.0636	0.0813	0.0269
40	0.2779	0.0503	0.0663	0.0225
50	0.2681	0.0413	0.0463	0.0176
60	0.2315	0.0372	0.0416	0.0152

TABLE 8

Computed values of maximum absolute errors and average absolute errors when fitting function G using n R_1 hyperspherical splines with $\nu = 3.5$ for $p = 4$.

n	Max. abs. error R_1 Splines	Ave. abs. error R_1 Splines
100	0.1034	0.0334
110	0.0971	0.0320

TABLE 9

A comparison of maximum absolute errors and average absolute errors using 1 ($\ell = 0$) spherical harmonic plus n K_1 splines versus 4 ($\ell = 0, 1$) spherical harmonics plus n K_1 splines versus 9 ($\ell = 0, 1, 2$) spherical harmonics plus n K_1 splines using function F_2 with $\nu = 2.0$ for $p = 1$.

	Maximum absolute errors			Average absolute errors		
n	$1S + nK_1$	$4S + nK_1$	$9S + nK_1$	$1S + nK_1$	$4S + nK_1$	$9S + nK_1$
18	0.0348	0.0126	0.0495	0.0105	0.0023	0.0132
20	0.0300	0.0059	0.0230	0.0094	0.0016	0.0054
22	0.0279	0.0055	0.0085	0.0080	0.0015	0.0022
24	0.0226	0.0044	0.0063	0.0069	0.0011	0.0017
26	0.0213	0.0036	0.0059	0.0066	0.0007	0.0014

TABLE 10

A comparison of maximum absolute errors and average absolute errors using 1 ($\ell = 0$) spherical harmonic plus n K_1 splines versus 4 ($\ell = 0, 1$) spherical harmonics plus n K_1 splines versus 9 spherical harmonics ($\ell = 0, 1, 2$) plus n K_1 splines using function G with $\nu = 2.0$ for $p = 1$.

	Maximum absolute errors			Average absolute errors		
n	$1S + nK_1$	$4S + nK_1$	$9S + nK_1$	$1S + nK_1$	$4S + nK_1$	$9S + nK_1$
10	0.0903	0.4648	—	0.0248	0.1348	—
20	0.0328	0.2346	0.3358	0.0120	0.0565	0.0663
30	0.0312	0.0957	0.1255	0.0101	0.0256	0.0351
40	0.0179	0.0698	0.2112	0.0063	0.0115	0.0476

function F_2 but not for function G. In general, we found that if hyperspherical harmonics fit a function reasonably well, then the addition of hyperspherical harmonics to our hyperspherical spline fits showed an improvement. But functions like G, which lack smoothness and symmetry, seem to be better fit using only hyperspherical spline functions.

Acknowledgments. The authors thank the referees and G. Wahba for several valuable suggestions which have helped us to improve the manuscript. The authors have also benefited from discussions about this work with Gy. Bencze, B. S. Bertram, G. H. Berthold, P. Doleschall, B. Fisk, S. P. Huestis, Z. C. Kuruoğlu, G. W. Pletsch, I. H. Sloan, S. L. Steinberg, D. Sulsky, T. G. Trucano, M. Trummer, and A. J. Waters.

REFERENCES

[1] M. ABRAMOWITZ AND I. STEGUN, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Applied Mathematical Series, Vol. 55, National Bureau of Standards, Gaithersburg, MD, 1964.

[2] N. ARONSZAJN, *Theory of reproducing kernels*, Amer. Math. Soc. Transl. Ser. 2, 68 (1950), pp. 337–404.

[3] BATEMAN MANUSCRIPT PROJECT, *Higher Transcendental Functions*, Vol. 2, A. Erdélyi, ed., McGraw-Hill, New York, 1953.

[4] P. BECKMANN, *Orthogonal Polynomials for Engineers and Physicists*, The Golem Press, Boulder, CO, 1975.

[5] C. CHANDLER AND A. G. GIBSON, *N-body quantum scattering theory in two Hilbert spaces. V. Computation strategy*, J. Math. Phys., 30 (1989), pp. 1533–1544.

[6] W. FREEDEN, *Spherical spline interpolation-basic theory and computational aspects*, J. Comput. Appl. Math., 3 (1984), pp. 367–375.

[7] M. GOLOMB, *Approximation by periodic spline interpolants on uniform meshes*, J. Approx. Theory, 1 (1968), pp. 26–65.

[8] I. S. GRADSHTEYN AND I. M. RYZHIK, *Table of Integrals, Series, and Products*, Academic Press, San Diego, CA, 1980.

[9] G. KIMELDORF AND G. WAHBA, *Some results on Tchebycheffian spline functions*, J. Math. Anal. Appl., 33 (1971), pp. 82–95.

[10] C. MÜLLER, *Spherical Harmonics*, Lecture Notes in Math., Vol. 17, Springer-Verlag, New York, 1966, pp. 1–20.

[11] H. J. TAIJERON, *Splines on Hyperspheres*, Ph.D. Dissertation, Dept. Mathematics and Statistics, University of New Mexico, 1989.

[12] G. WAHBA, *Smoothing noisy data by spline functions*, Numer. Math., 24 (1975), pp. 383–393.

[13] ———, *Spline interpolation and smoothing on the sphere*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 5–16; 3 (1982), pp. 385–386.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1126–1133, September 1994

009

MONOTONIC SMOOTHING SPLINES FITTED BY CROSS VALIDATION*

S. N. WOOD†

Abstract. A practical method for calculating monotonic cubic smoothing splines is given. Linear sufficient conditions for monotonicity are employed, and the spline coefficients are obtained using quadratic programming. The method enables efficient cross-validation estimates of the smoothing parameter to be made and confidence intervals to be calculated for the resulting spline. The results are easy to extend to histogram data.

Key words. smoothing spline, spline, monotonic, cross validation

AMS subject classifications. 65D07, 65D10

1. Introduction. There are a number of practical spline based methods for monotonic interpolation [1], [3], [5], [6], [9], [10], [12]–[14], [17], [18], [22], [30], but monotonic smoothing splines have received less attention. Unconstrained smoothing splines are useful tools for data analysis principally because of the cross-validation methods pioneered in [4] for choosing the amount of smoothing appropriate for any dataset and the further work in [27] on estimating confidence intervals associated with the resulting spline even when the variance of the original data is unknown. Similar results are not available for monotonic smoothing splines [24]. A cross-validation method for linearly constrained thin-plate splines has been developed in [25], but monotonicity or confidence intervals were not considered. Subsequent work [20], [15] on monotonically constrained cubic smoothing splines requires the level of smoothing to be chosen by eye, whilst [7] is concerned with efficient algorithms for more restrictive constraints involving convexity and does not address the choice of smoothing parameter. Confidence intervals are not considered in [15] or [7]. Ramsay [20] plots confidence intervals on one figure, but does not discuss their calculation or properties.

The practical methods for calculating monotonic smoothing splines proposed in [15] and [20] involve constraints on the parameters of B-splines, which yield sufficient (but not necessary) conditions for monotonicity. In this note I instead use the piecewise polynomial representation of a spline and some sufficient conditions for monotonicity first used in [14] for monotonic interpolation with piecewise cubics. This yields a quadratic programming problem with linear constraints, and by considering some properties of the active set method for positive-definite quadratic programming given in [11] it is possible to extend the cross-validation and confidence interval methods of [4] and [27] to monotonic cubic smoothing splines. A demonstration of the technique applied to artificial data is given at the end of this article.

2. The method. The set of n "knots" $\{(x_i, a_i)\}$ such that $x_i < x_{i+1}$ for $1 \leq i \leq n-1$ can be interpolated by a curve $s(x)$ defined by

$$s(x) = s_i(x) = a_i\phi_{0i}(x) + a_{i+1}\phi_{1i} + c_i\gamma_{0i}(x) + c_{i+1}\gamma_{1i}(x) \quad \text{for } x_i \leq x \leq x_{i+1},$$

where the basis functions ϕ and γ are defined in Table 1; $c_i = s''(x_i)$, $a_i = s(x_i)$, and $s_i(x)$ is a section of cubic polynomial. The c_i's can be obtained from the a_i's by requiring continuity of $s(x)$, $s'(x)$, and $s''(x)$ at each x_i, and setting $c_1 = c_n = 0$, this yields the matrix system

$$\mathbf{Dc} = \mathbf{Ha},$$

where $\mathbf{D}$ and $\mathbf{H}$ are defined in Table 1, $\mathbf{a} = (a_1, a_2, \ldots, a_n)^{\mathrm{T}}$ and $\mathbf{c} = (c_2, c_3, \ldots, c_{n-1})^T$. The interpolant calculated in this manner is a cubic spline.

*Received by the editors August 14, 1991; accepted for publication (in revised form) August 2, 1993.

†Natural Environment Research Council (NERC) Centre for Population Biology, Imperial College, Silwood Park, Ascot, Berkshire SL5 7PY, United Kingdom (snw@ic.ac.uk).

TABLE 1
Definitions for a cubic spline $h_j = x_{j+1} - x_j$.

Definitions of **H** and **D**. All elements are 0 except the following.		
$H_{j,j} = 1/h_j$ $D_{j,j} = (h_j + h_{j+1})/3$	$H_{j,j+1} = -(1/h_j + 1/h_{j+1})$	$H_{j,j+2} = 1/h_{j+1}$ $1 \le j \le n-2$
$D_{j,j+1} = h_{j+1}/6$	$D_{j+1,j} = h_{j+1}/6$	$1 \le j \le n-3$
Definitions of **P** and **U**. All elements are 0 except the following.		
$P_{j,j} = 2$		$1 \le j \le n$
$P_{j,j-1} = h_j/(h_j + h_{j+1})$ $U_{j,j+1} = 3P_{j,j+1}/h_j$	$P_{j,j+1} = 1 - P_{j,j-1}$ $U_{j,j} = -(U_{j,j+1} + U_{j,j-1})$	$U_{j,j-1} = -3P_{j,j-1}/h_{j-1}$ $2 \le j \le n-1$
$P_{1,2} = 1$ $P_{n,n-1} = 1$	$U_{1,1} = -3/h_1$ $U_{n,n-1} = -3/h_{n-1}$	$U_{1,2} = -U_{1,1}$ $U_{n,n} = -U_{n,n-1}$
Basis functions for a cubic spline.		
$\phi_{0j}(x) = (x_{j+1} - x)/h_j$ $\phi_{1j} = (x - x_j)/h_j$	$\gamma_{0j}(x) = \{\phi_{0j}(x)^3 - \phi_{0j}(x)\}h_j^2/6$ $\gamma_{1j}(x) = \{\phi_{1j}(x)^3 - \phi_{1j}(x)\}h_j^2/6$	

If **y** is a vector of observations, which we wish to approximate with the spline $s(x)$, then we seek **a** the vector of the values of the spline at the knots such that

$$\int_{x_1}^{x_n} [s''(x)]^2 dx + \frac{\lambda}{n}\|\mathbf{W}(\mathbf{a} - \mathbf{y})\|^2 \tag{1}$$

is minimised. λ is the "smoothing parameter," **W** is a diagonal matrix of observation weights, and $\|\cdot\|$ is the standard Euclidian norm. The resulting curve is a cubic smoothing spline. Extensive discussion of spline theory is given in [2], [8], [23], [28], and at a more introductory level in [16]. It is easy to show that (1) is equivalent to minimising

$$\frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{G}_\lambda\mathbf{a} + \mathbf{c}_\lambda^{\mathrm{T}}\mathbf{a}, \tag{2}$$

where $\mathbf{c}_\lambda^{\mathrm{T}} = -2\frac{\lambda}{n}\mathbf{y}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}$ and $\mathbf{G}_\lambda = 2[\mathbf{H}^{\mathrm{T}}\mathbf{D}^{-1}\mathbf{H} + \frac{\lambda}{n}\mathbf{W}^{\mathrm{T}}\mathbf{W}]$. $\mathbf{G}_\lambda$ is clearly nonnegative definite.

The minimisation of (2) will not, in general, give rise to an $s(x)$, which is monotonic, but conditions for monotonicity of a cubic over an interval $[x_i, x_{i+1}]$ are well known and can be expressed in terms of β and δ where $\beta = s'(x_{i+1})/\Delta_i$, $\delta = s'(x_i)/\Delta_i$, and $\Delta_i = (a_{i+1} - a_i)/(x_{i+1} - x_i)$. The necessary and sufficient conditions for monotonicity are that δ and β should lie within the region of Fig. 1, defined by the union of A, B, and C. These nonlinear constraints are used in several shape-preserving interpolation schemes, notably [30], [9], and [1]. Minimisation of (2) subject to these constraints requires quadratic programming with nonlinear constraints. This is not an attractive proposition (see [11, Chap. 6]) and instead I propose to follow [14] in using the linear sufficient conditions for monotonicity $0 \le \beta \le 3$ and $0 \le \delta \le 3$; that is β and δ lie within region A of Fig. 1. If $\mathbf{b} = (s'(x_1), s'(x_2), \ldots, s'(x_n))^{\mathrm{T}}$ then, for a cubic spline, **b** can be written as a linear transformation of **a**: $\mathbf{b} = \mathbf{B}\mathbf{a}$, where $\mathbf{B} = \mathbf{P}^{-1}\mathbf{U}$ and **P** and **U** are defined in Table 1. Writing $\mathbf{B}_i$ for the ith row of **B**, it is now possible to write the sufficient conditions for monotonicity over $[x_i, x_{i+1}]$ in terms of **a**:

$$3(a_{i+1} - a_i)/h_i - \mathbf{B}_i\mathbf{a} \ge 0, \quad 3(a_{i+1} - a_i)/h_i - \mathbf{B}_{i+1}\mathbf{a} \ge 0,$$

$$\mathbf{B}_i\mathbf{a} \ge 0, \quad \mathbf{B}_{i+1}\mathbf{a} \ge 0 \quad \text{and} \quad a_{i+1} - a_i \ge 0$$

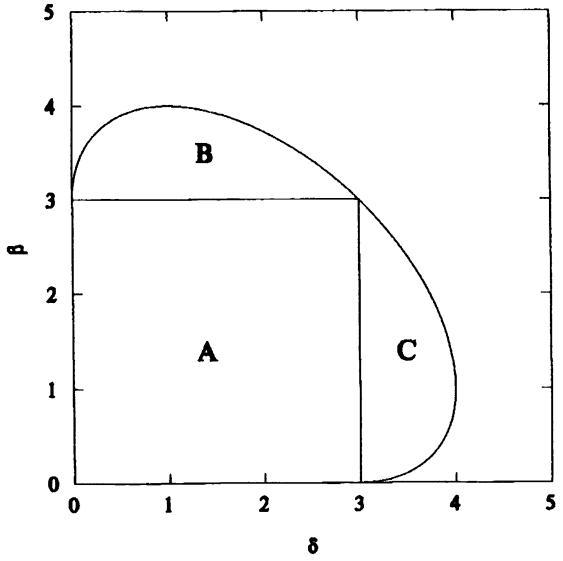


FIG. 1. *The monotonicity region for a section of a cubic spline.* A *is the subregion used for this work.* β *and* δ *are defined in* §2.

for $s(x)$ nondecreasing and

$$\mathbf{B}_i\mathbf{a} - 3(a_{i+1} - a_i)/h_i \geq 0, \quad \mathbf{B}_{i+1}\mathbf{a} - 3(a_{i+1} - a_i)/h_i \geq 0,$$

$$\mathbf{B}_i\mathbf{a} \leq 0, \quad \mathbf{B}_{i+1}\mathbf{a} \leq 0, \quad \text{and} \quad a_i - a_{i+1} \geq 0$$

for $s(x)$ nonincreasing, where $h_i = x_{i+1} - x_i$. Clearly, the sufficient conditions for monotonicity of $s(x)$ over $[x_1, x_n]$ can be written as

$$\mathbf{Ca} \geq [\mathbf{0}], \tag{3}$$

where $[\mathbf{0}]$ is a vector of zeros. Additional absolute constraints on any a_i are trivial to add to this constraint set. It is fairly easy to show that these constraints are equivalent to those used in [15] (which are apparently similar to those used in [20]). Note also that any straight line sloping up or down, as appropriate, will satisfy the monotonicity constraints.

Minimisation of (2) subject to (3) is a positive definite quadratic programming problem, which will be solved using the algorithm given in [11]. To extend cross-validation methods from unconstrainted to constrained cubic splines some features of this algorithm must be considered.

The method starts with a feasible vector $\mathbf{a}$, which violates none of the inequality constraints in (3) but which may be constrained to satisfy t of the constraints as equality constraints. These t exact constraints are termed an "Active set" and can be expressed $\mathbf{C}_A\mathbf{a} \geq [\mathbf{0}]$. The algorithm is straightforward.

The $t \times n$ matrix $\mathbf{C}_A$ is factorized $\mathbf{C}_A\mathbf{Q} = [\mathbf{0}, \mathbf{T}]$, where $\mathbf{T}$ is a $t \times t$ matrix such that $T_{i,j} = 0$ if $i + j < n$, $\mathbf{0}$ is an appropriate matrix of zeros, and $\mathbf{Q}$ is an $n \times n$ orthogonal matrix constructed from the product of the identity matrix and a series of Householder matrices applied to the right. The first $n - t$ columns of $\mathbf{Q}$ define a matrix $\mathbf{Z}$, the columns of which form a basis for the null space of $\mathbf{C}_A$, the space in which any movement is possible whilst still satisfying the active set as equality constraints. The remaining columns of $\mathbf{Q}$ form the basis

$\mathbf{\Omega}$ of the space orthogonal to $\mathbf{Z}$: this is the "range space" of the active set. Given $\mathbf{Z}$ a search direction $\mathbf{p}$ can be found that is guaranteed not to violate the active constraints. The search direction $\mathbf{p}$ comes from the solution of $\mathbf{Z}^{\mathrm{T}}\mathbf{G}_\lambda\mathbf{Z}\mathbf{p}_z = -\mathbf{Z}^{\mathrm{T}}\mathbf{g}$, where $\mathbf{p} = \mathbf{Z}\mathbf{p}_z$ and $\mathbf{g} = \mathbf{G}_\lambda\mathbf{a} + \mathbf{c}_\lambda$. The vector $\mathbf{a}^* = \mathbf{a} + \mathbf{p}$ is the position of the minimum of the quadratic problem within the null space of $\mathbf{C}_A$, but if a step from $\mathbf{a}$ to $\mathbf{a}^*$ would violate one or more of the inactive constraints, then a shorter step must be taken to the constraint nearest $\mathbf{a}$ and that constraint must be included in an updated active set matrix $\mathbf{C}_A$. The algorithm then proceeds from the beginning with the new $\mathbf{C}_A$ and $\mathbf{a}$. If the solution has reached a constrained minimum $\mathbf{a}^*$, then it is necessary to find out whether all the active constraints are currently required. This is achieved by evaluating the Lagrange multipliers associated with each active constraint. If no constraints need to be deleted from $\mathbf{C}_A$, then the problem has been solved. Extensive computational details are given in [11]. What matters here are the spaces defined by $\mathbf{\Omega}$ and $\mathbf{Z}$, which will be used in the method of choosing λ.

Assume a data model $y_i = f(x_i) \pm \varepsilon_i$, where $f(x)$ is the underlying but unknown smooth function and the ε_i's are error terms such that $\mathbf{E}(\varepsilon_i) = 0$, $\mathbf{E}(\varepsilon_i\varepsilon_i) = w_i^{-2}\sigma^2$, and $\mathbf{E}(\varepsilon_i\varepsilon_i) = 0$ if $i \neq j$. Let $s_\lambda(x)$ be the monotonic smoothing spline fitted to $\mathbf{y}$ by solving (2) subject to (3). The spline $s_\lambda(x)$ is intended to be an approximation of $f(x)$. A reasonable measure of the success of this approximation is provided by

$$R(\lambda) = \sum_{i=1}^{n} w_i^2[f(x_i) - s_\lambda(x_i)]^2/n = \|\mathbf{W}(\mathbf{f} - \mathbf{a}^*)\|^2/n, \tag{4}$$

where $W_{i,i} = w_i$ and $W_{i,j} = 0$ if $i \neq j$. The term λ should be chosen to minimize the expected value of $R(\lambda)$. In the unconstrained case it has been suggested that λ should be chosen so that $\|\mathbf{a}^* - \mathbf{y}\|^2/n = \sigma^2$, [21], but in [26] it is shown that this leads to oversmoothing and a practical estimator of $R(\lambda)$ is derived. Reference [26] uses the fact that unconstrained smoothing with spline functions can be summarised by a linear equation of the form $\mathbf{a} = \mathbf{A}\mathbf{y}$, where $\mathbf{A}$ is called the "influence matrix." In general, such an equation cannot be constructed for the constrained spline function, since the constraints also enter the solution. However, it is possible to estimate λ by minimisation of an estimator of $R(\lambda)$ for any particular active set $\mathbf{C}_A$. As mentioned above the null space of $\mathbf{C}_A$, $\mathbf{Z}$ has a complementary "range" space $\mathbf{\Omega}$ and an initial estimate $\mathbf{a}^0$ of the solution $\mathbf{a}^*$ of the spline fitting problem can therefore be written $\mathbf{a}^0 = \mathbf{\Omega}\mathbf{a}_\Omega^0 + \mathbf{Z}\mathbf{a}_Z^0$. The solution $\mathbf{a}^*$ is therefore $\mathbf{a}^* = \mathbf{K}\mathbf{a}_\Omega^0 + \tilde{\mathbf{A}}\mathbf{y}$, where $\mathbf{K} = (\mathbf{I} - \mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{G}_\lambda\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{G}_\lambda)\mathbf{\Omega}$ and $\tilde{\mathbf{A}} = 2\frac{\lambda}{n}\mathbf{Z}(\mathbf{Z}^{\mathrm{T}}\mathbf{G}_\lambda\mathbf{Z})^{-1}\mathbf{Z}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}$. The attractive feature of this representation is that $\mathbf{a}_\Omega^0$ is uniquely determined by the constraints in the active set, unlike $\mathbf{a}^0$. It is now easy to show that for a given $\mathbf{C}_A$, minimisation of the $\mathbf{E}\{R(\lambda)\}$ can be achieved by minimisation of

$$\tilde{R}(\lambda) = \|\mathbf{W}(\mathbf{a}^* - \mathbf{y})\|^2/n + 2\sigma^2\mathrm{Tr}(\tilde{\mathbf{A}})/n \tag{5}$$

(cf. [4, Eqn. (1.8)]). So one way of finding the value of λ is to search for the turning point of (5), which is consistent with the active set implied by such a turning point. This method is adopted in the work reported below.

In the unconstrained case, [4] shows that the value of λ minimising the expected value of (4) can be estimated without prior knowledge of σ using the process of generalized cross validation. If $s_\lambda^{[k]}(x)$ is the spline function calculated using every point in the data vector $\mathbf{y}$ except y_k, then

$$V(\lambda) = \sum_{k=1}^{n} u_k(s_\lambda^{[k]}(x) - y_k)^2/n$$

is taken as a measure of the badness of λ (u_k is a weight compensating for uneven mesh, etc.). So λ is chosen by minimisation of $V(\lambda)$. Once again the presence of the constraints in the

solution of the monotonic spline problem precludes direct use of the results in [4], but by using intermediate results from the paper it is easy to show that for a given active set minimisation of:

$$\tilde{V}(\lambda) = n\|\mathbf{W}(\mathbf{a}^* - \mathbf{y})\|^2/[\mathrm{Tr}(\mathbf{I} - \tilde{\mathbf{A}})]^2 \tag{6}$$

is equivalent to minimisation of $V(\lambda)$ with respect to λ. Equation (6) can be used in the same way as (5) to obtain an estimate of λ. $\tilde{\mathbf{A}}$ has the same role in choosing the smoothing parameter for the monotonic spline that $\mathbf{A}$ has for unconstrained spline smoothing. Minimisation of (6) should be compared with the method for constrained multivariate thin-plate splines given in [25].

$\tilde{\mathbf{A}}$ can be used to extend the results on confidence intervals for unconstrained spline smoothing given in [27] to the constrained case. For the spline fitted by cross validation, an estimate of σ^2, $\sigma_v^2 = \|\mathbf{W}(\mathbf{a}^* - \mathbf{y})\|^2/\mathrm{Tr}(\mathbf{I} - \tilde{\mathbf{A}})$, can be obtained. Ninety-five percent "confidence intervals," $p_v = 2\sigma_v\sqrt{\mathrm{Tr}(\tilde{\mathbf{A}}[\mathbf{W}^\mathrm{T}\mathbf{W}]^{-1})/n}$ or $p_R = 2\sigma\sqrt{\mathrm{Tr}(\tilde{\mathbf{A}}[\mathbf{W}^\mathrm{T}\mathbf{W}]^{-1})/n}$, can also be calculated depending on whether or not σ is known a priori. Finally note that all the results given above can easily be applied to area approximating splines as used in [29], for example.

3. Some examples. The three monotonic relationships, $f_i(x)$, shown in Fig. 2 were "sampled" at 45 equally spaced points with three levels of additive Guassian noise: standard deviation 0.05, 0.1, and 0.2. The noise was generated using methods given in [19]. Fifty replicate sets of data were generated for each relationship at each noise level. Three splines $s_\lambda^V(x)$, $s_\lambda^R(x)$, and $s_\lambda^t(x)$, with smoothing parameters λ_V, λ_R, and λ_t, were fitted to each set of data by minimisation of $\tilde{V}(\lambda)$, $\tilde{R}(\lambda)$, and $\|\mathbf{a}^* - \mathbf{f}\|^2/n$, respectively. The spline $s_\lambda^t(x)$ is closest to the function from which the data was sampled and enables the efficacy of $s_\lambda^V(x)$ and $s_\lambda^R(x)$ to be examined. For the spline fitted by cross validation, σ_V was calculated and 95% "confidence intervals," were produced for both $s_\lambda^V(x)$ and $s_\lambda^R(x)$. In all cases $\mathbf{W}$ was set to the identity matrix.

Several test statistics were calculated for each replicate as follows.

(i) XE is the proportion extra mean square deviation from $f_i(x)$ of $s_\lambda^V(x)$ and $s_\lambda^R(x)$ in comparison to the mean square deviation of $s_\lambda^t(x)$ from $f_i(x)$: this is intended to see how well the spline recaptures the true underlying function, in comparison with the best achievable.

(ii) RM measures the closeness of the estimated smoothing parameter to the one that would have led to the most accurate reconstruction of the underlying function. RM is the ratio of $\log(\lambda_V)$ or $\log(\lambda_R)$ to $\log(\lambda_t)$.

(iii) CIC, the percentage of $f_i(x)$ within the region the region defined by $s_\lambda^{V/R}(x) \pm p_{V/R}$, tests the effectiveness of the confidence intervals.

Means and standard deviations for these quantities are given in Table 2 for each treatment. Table 2 also records the mean number of constraints active at the solution K and the mean estimated value of σ, σ_V. Figure 2 shows three randomly selected examples from the set of simulations.

The estimates of σ, σ_V, are excellent, although the confidence intervals obtained from them are slightly too narrow when calculated from the minimisation of (6) and are slightly too wide when calculated by minimisation of (5). The deterioration in the ability of the splines $s_\lambda^V(x)$ and $s_\lambda^R(x)$ to recapture the underlying relationship $f(x)$ as the level of noise increases is unsurprising. If σ is large enough, then minimisation of (5) or (6) is bound to produce a straight line whatever the form of $f(x)$. This tendency for the method to produce a smoother relationship as the noise level becomes very large also explains the systematic tendency to underestimate λ and, therefore, k relative to $s_\lambda^t(x)$.

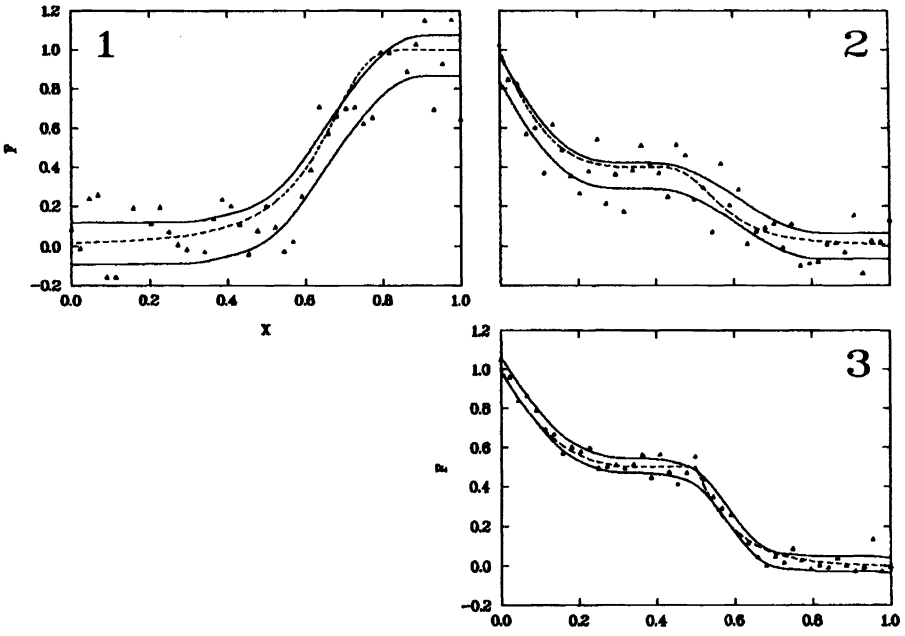


FIG. 2. *The dotted lines show the functions $f_i(x)$ used to test the spline fitting algorithms, as detailed in* §3. *The numbers in the corner of each graph are the function numbers of Table* 2. *The continuous lines show* 95% *confidence regions for a spline fitted to the data shown by cross validation.* σ =0.2, 0.1, *and* 0.05 *for graphs* 1, 2, *and* 3, *respectively.*

TABLE 2

A summary of the results of §3. *The figures are means over* 50 *replicates of the statistics. Figures in brackets,* (), *are standard deviations. The figures in square brackets,* [], *are the mean numbers of constraints active for* s'_λ. *The notation* $\pm$ *is used to prefix the standard error of the mean. The function,* f, *numbers refer to Fig.* 2. *Columns* $\tilde{R}$ *are for splines fitted by minimisation of* (5) *and* $\tilde{V}$ *for splines fitted by minimisation of* (6).

f	Stat	Noise level σ					
		0.05		0.1		0.2	
		$\tilde{V}$	$\tilde{R}$	$\tilde{V}$	$\tilde{R}$	$\tilde{V}$	$\tilde{R}$
1	XE	0.09(.10)	0.10(.11)	0.11(.14)	0.12(.16)	0.15(.15)	0.15(.14)
	RM	0.97(.09)	0.97(.09)	0.94(.09)	0.94(.09)	0.86(.11)	0.86(.12)
	CIC	92(7)	98(2)	94(8)	98(3)	90(11)	94(6)
	K	8	8 [9]	8	8 [11]	8	8 [12]
	σ_V	0.047 $\pm$ 0.001		0.097 $\pm$ 0.002		0.0197 $\pm$ 0.003	
2	XE	0.08(0.09)	0.09(.09)	0.15(.15)	0.12(.13)	0.31(.42)	0.29(.35)
	RM	0.95(0.08)	0.94(.07)	0.92(.11)	0.92(.10)	0.76(.61)	0.76(.61)
	CIC	92(7)	98(2)	92(8)	97(2)	82(18)	93(7)
	K	6	5 [8]	6	6 [8]	4	4 [8]
	σ_V	0.047 $\pm$ 0.001		0.101 $\pm$ 0.001		0.201 $\pm$ 0.003	
3	XE	0.09(.12)	0.10(.12)	0.12(.15)	0.12(.13)	0.18(.25)	0.15(.21)
	RM	0.93(.10)	0.92(.10)	0.92(.12)	0.92(.12)	0.74(.48)	0.83(.33)
	CIC	91(6)	99(1)	91(8)	98(3)	77(22)	89(13)
	K	8	8 [11]	7	7 [10]	5	5 [9]
	σ_V	0.049 $\pm$ 0.004		0.100 $\pm$ 0.001		0.200 $\pm$ 0.004	

4. Discussion. Unconstrained smoothing splines are useful because reliable methods exist for choosing the smoothing parameter λ. The approach described in the previous sections of this note extends this power to monotonic smoothing splines.

One possible criticism of the method presented here (or of the other published methods) is that monotonicity is imposed by unnecessarily restrictive sufficient conditions. There are two replies to this. First, Fig. 1 shows that the linear conditions cover a large proportion of the necessary parameter space for monotonicity, but this is a weak argument on its own. More persuasive is a point first made in [2, p. 248]: the smoothing spline as presented here has far more knots than are actually required to represent the noisy data to which it is fitted. Put another way, the curve $s_\lambda(x)$ is potentially far more flexible than it needs to be. This implies that it is unlikely that there is any significant advantage gained by using the full nonlinear monotonicity constraints, which will simply allow $s_\lambda(x)$ slightly more flexibility. This argument becomes increasingly dubious as errors in the data become very small (not a problem with which the author has much experience). Similarly, it is less convincing as the number of knots in a least squares spline decreases relative to the number of datapoints.

Two further practical problems are associated with monotonic smoothing splines. First, the method described here will be slow if applied to large datasets. For noisy data this could be overcome using splines with fewer knots than there are datapoints: least-squares splines. The remaining drawback is the tedious computer programming required to implement the methods, especially given that standard quadratic programming packages do not give easy access to the matrices $\mathbf{Z}$ and $\mathbf{\Omega}$. This difficulty can be overcome by writing to the author for a "C" computer program implementing the method described in this note.

Acknowledgement. Thanks to Hans Metz for several helpful discussions.

REFERENCES

[1] R. K. BEATSON AND H. WOLKOWICZ, *Post-processing piecewise cubics for monotonicity*, SIAM J. Numer. Anal., 26 (1989), pp. 480–502.
[2] C. DEBOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.
[3] P. COSTANTINI, *Co-monotone interpolating splines of arbitrary degree—a local approach*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 1026–1034.
[4] P. CRAVEN AND G. WAHBA, *Smoothing noisy data with spline functions*, Numer. Math., 31 (1979), pp. 377–403.
[5] R. DELBOURGO AND J. A. GREGORY, *C^2 quadratic spline interpolation to monotonic data*, IMA J. Numer. Anal., 3 (1983), pp. 141–152.
[6] S. DIETZE AND J. W. SCHMIDT, *Determination of shape preserving spline interpolants with minimal curvature via dual programs*, J. Approx. Theory, 52 (1988), pp. 43–57.
[7] T. ELFVING AND L.-E. ANDERSSON, *An algorithm for computing constrained smoothing spline functions*, Numer. Math., 52 (1988), pp. 583–595.
[8] R. L. EUBANK, *Spline Smoothing and Nonparametric Regression*, Marcel Dekker, New York, 1988.
[9] F. N. FRITSCH AND R. E. CARLSON, *A method for constructing local monotone piecewise cubic interpolants*, SIAM J. Numer. Anal., 17 (1980), pp. 238–246.
[10] F. N. FRITSCH AND J. BUTLAND, *A method for constructing local monotone piecewise cubic interpolants*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 300–304.
[11] P. E. GILL, W. MURRAY, AND M. H. WRIGHT, *Practical Optimization*, Academic Press, London, 1981.
[12] J. A. GREGORY, *Shape preserving rational spline interpolation*, in Rational Approximation and Interpolation, Lecture Notes in Mathematics 1105, Springer-Verlag, New York, 1984, pp. 431–441.
[13] J. A. GREGORY AND R. DELBOURGO, *Piecewise rational quadratic interpolation to monotonic data*, IMA J. Numer. Anal., 2 (1982), pp. 123–130.
[14] J. M. HYMAN, *Accurate monotonicity preserving cubic interpolation*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 645–654.
[15] C. KELLY AND J. RICE, *Monotone smoothing with application to dose-response curves and assessment of synergism*, Biometrics, 46 (1990), pp. 1071–1085.
[16] P. LANCASTER AND K. SALKAUSKAS, *Curve and Surface Fitting*, Academic Press, London, UK, 1986.
[17] D. F. MCALLISTER AND J. A. ROULIER, *An algorithm for computing a shape-preserving osculatory quadratic spline*, ACM Trans. Math. Software, 7 (1981), pp. 331–347.

[18] R. MORANDI AND P. COSTANTINI, *Piecewise monotone quadratic histosplines*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 397–406.
[19] W. H. PRESS, B. P. FLANNERY, S. A. TEUKOLSKY, AND W. T. VETTERLING, *Numerical Recipes in C*, Cambridge University Press, Cambridge, UK, 1988.
[20] J. O. RAMSAY, *Monotone regression splines in action*, Statist. Sci., 3 (1988), pp. 425–461.
[21] C. H. REINSCH, *Smoothing by spline functions*, Numer. Math., 10 (1967), pp. 177–183.
[22] M. SAKAI AND M. C. LOPEZ DE SILANES, *A simple rational spline and its application to monotonic interpolation to monotonic data*, Numer. Math., 50 (1986), pp. 171–182.
[23] 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.
[24] F. UTRERAS, *Smoothing noisy data under monotonicity constraints: existence, characterization and convergence rates*, Numer. Math., 47 (1985), pp. 611–625.
[25] M. VILLALOBOS AND G. WAHBA, *Inequality-constrained multivariate smoothing splines with application to the estimation of posterior probabilities*, J. Amer. Statist. Assoc., 82 (1987), pp. 239–248.
[26] G. WAHBA, *Smoothing noisy data with spline functions*, Numer. Math., 24 (1975), pp. 383–393.
[27] ———, *Bayesian "confidence intervals" for the cross-validated smoothing spline*, J. Roy. Statist. Soc. Sci., B, 45 (1983), pp. 133–150.
[28] ———, *Spline Methods for Observational Data*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.
[29] S. N. WOOD AND R. M. NISBET, *Estimation of Mortality Rates in Stage Structured Populations*, Lecture Notes in Biomathematics 90, Springer-Verlag, Berlin, 1991.
[30] Z. YAN, *Piecewise cubic curve fitting algorithm*, Math. Comp., 49 (1987), pp. 203–213.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1134–1144, September 1994

010

UNIFORM REFINEMENT OF A TETRAHEDRON*

MARIA ELIZABETH G. ONG†

Abstract. A uniform refinement strategy for a tetrahedron is presented. Most finite element theories are based on the assumption that the tetrahedral elements in the refinement do not degenerate. In this paper, the author presents a refinement strategy that is nondegenerate and uniform for the model tetrahedra considered and quasi uniform for arbitrary tetrahedra. It can be used to construct nested, multilevel triangulations. At level j of refinement, an arbitrary nondegenerate tetrahedron in the initial triangulation is partitioned into 2^{3j} tetrahedra of equal volume. This refinement strategy can be implemented easily by partitioning block elements instead of the more complicated tetrahedral elements. This feature makes the use of tetrahedral elements attractive in a computer code.

Key words. finite element, nondegenerate, tetrahedron, triangulation, quasi uniform, uniform refinement

AMS subject classifications. 65N30, 65N50

1. Introduction.

The finite element method is often used to obtain discretizations of second order elliptic problems. It is based on an underlying triangulation of the domain, which is typically refined to achieve higher accuracy in the discrete solution. When the domain is three dimensional, the elements are often taken to be tetrahedra. In such cases, it is imperative that the refinement of tetrahedral elements result in nondegenerate tetrahedra. In this paper, we describe a quasi-uniform and nondegenerate refinement strategy for an arbitrary tetrahedron.

We make use of the cube as a device to simplify the task of refining tetrahedra. We first describe a uniform refinement strategy for the model tetrahedra that triangulates the cube, and then we extend this refinement strategy to arbitrary tetrahedra. This leads to a multilevel refinement strategy that yields nested, quasi-uniform and nondegenerate tetrahedra.

Several studies have been done on the triangulation of an n-cube in a general context [3], [4], [8], [15]. (The spatial dimension is n.) These do not, however, cover the successive refinement of those elements that triangulate the cube. Moreover, if a cube is partitioned into small cubes such that the small cubes are contained in the larger cubes, it is not guaranteed that the elements that triangulate the small cubes are contained in the elements that triangulate the larger cubes. This inclusion property, or *nesting*, of the elements is important in many multilevel methods; e.g., multigrid methods [6], [10].

One approach to refining arbitrary tetrahedra is presented in [13], [16]. However, it fails to find an appropriate triangulation of a cube that would enable its use as a powerful tool in simplifying the task of refining arbitrary tetrahedra. This paper provides a practical and illustrative guide to refinement of tetrahedra.

We first describe a uniform refinement strategy for a model tetrahedron, which is one of the two types of tetrahedra that triangulate the cube. (These two types are reflections of each other; hence, it is sufficient to study the triangulation of one of them.) This tetrahedron is refined into eight small tetrahedra, each of which is *similar* to the model tetrahedron or its reflection. This similarity to the model tetrahedron ensures nondegeneracy in the refinement. We say the refinement is *nondegenerate* if

$$\sigma_T = \frac{h_T}{\rho_T} \leq \sigma \quad \forall\, T \in \mathcal{T}_k, \quad k = 0, 1, 2, \ldots, \tag{1.1}$$

*Received by the editors February 5, 1992; accepted for publication (in revised form) August 3, 1993.

†Department of Mathematics, University of California, San Diego, California 92093 (ong@ucsd.edu). This research was supported mainly by Air Force Office of Scientific Research grant 86-0154 and partly by National Science Foundation grant ASC85-19353, Department of Energy grant DOE-627815, National Science Foundation grant ASC90-03002, Office of Naval Research grant N00014-86-K-0691, and Army Research Office grant DAAL03-88-K-0085.

where $\mathcal{T}_k$ is the triangulation at level k of refinement, h_T is the diameter of tetrahedron T, ρ_T is the diameter of the largest sphere inscribed in T, σ_T is the measure of nondegeneracy of T, and σ is a positive constant. (The initial level of refinement is $k = 0$.) When nondegeneracy is satisfied, we have a *regular family of triangulations* according to the definition given in [2]. A single *tetrahedron* is said to be *nondegenerate* if $\sigma_T \neq \infty$. We assume that any tetrahedron that we refine is initially nondegenerate.

The resulting eight small tetrahedra also have equal volumes. Moreover, their diameters are equal. Hence, the refinement is *uniform.* That is,

$$\frac{h_{T_1}}{h_{T_2}} = 1 \quad \forall\, T_1, T_2 \in \mathcal{T}_k, \quad k = 0, 1, 2, \ldots, \tag{1.2}$$

where h_{T_1} and h_{T_2} are the diameters of any two tetrahedra T_1 and T_2, respectively, in the triangulation $\mathcal{T}_k$. The refinement becomes *quasi uniform* when applied to an arbitrary tetrahedron. That is,

$$\frac{h_{T_1}}{h_{T_2}} \leq c_0 \quad \forall\, T_1, T_2 \in \mathcal{T}_k, \quad k = 0, 1, 2, \ldots, \tag{1.3}$$

where c_0 is a positive constant.

The refinement strategy to be described also generates *nested* tetrahedra such that the triangulation $\mathcal{T}_{k+1}$ is obtained by partitioning some or all tetrahedra $T \in \mathcal{T}_k$ for $k = 0, 1, 2, \ldots$.

In §2, we describe the uniform refinement strategy for the model tetrahedron. In §3, we use the uniform refinement procedure for the model tetrahedron to show that:

(1) *Any tetrahedron T can be refined at level j into 2^{3j} tetrahedra that are equivolume and nested.*

(2) *For any tetrahedron T, there exists a refinement that is nondegenerate and quasi uniform.*

The advantages, usefulness, and limitations of this refinement strategy are discussed in §4.

2. Refinement of the model tetrahedron. We make use of the *cube* as a device in the refinement strategy for the model tetrahedron. Consider a cube with length H and with $(2^0 + 1)^3 = 8$ nodes in the initial refinement, called level 0 refinement and denoted by $\mathcal{T}_0$.

Divide the cube into six tetrahedra, as shown in Fig. 1.[1] The tetrahedra in Fig. 1 are of two types: T_1 and T_2. Note that T_2 is simply a reflection of T_1. Thus, it suffices to demonstrate the refinement strategy for only one of these two tetrahedra. The tetrahedron T_1 is the tetrahedron that we will refine.

The cube has volume $V = H^3$. The volume of a tetrahedron is given by:

$$\frac{1}{3} \times (\text{area of base}) \times \text{height}. \tag{2.4}$$

Since the six tetrahedra in the cube are either identical to or reflections of each other; each has volume $\frac{1}{6}V$.

We now describe the procedure to refine the model tetrahedron T_1 uniformly. If we position vertex 0 of the cube in Fig. 1 at the origin, $(x, y, z) = (0, 0, 0)$, then the model tetrahedron T_1 has vertices 0=(0,0,0), 1=(1,0,0), 2=(0,1,0), and 5=(1,0,1) as shown in Fig. 2. In Figs. 2–4, we denote by a_{ij} the midpoint of the edge that connects vertices i and j, and we

[1]This particular triangulation of the cube is referred to as Kuhn's triangulation [14], falls under one type of triangulation discussed in [7], and appears in [9]. The tetrahedra are called quadrirectangular tetrahedra in [3]. Other triangulations of the cube that are amenable to the refinement strategy are discussed in §4.

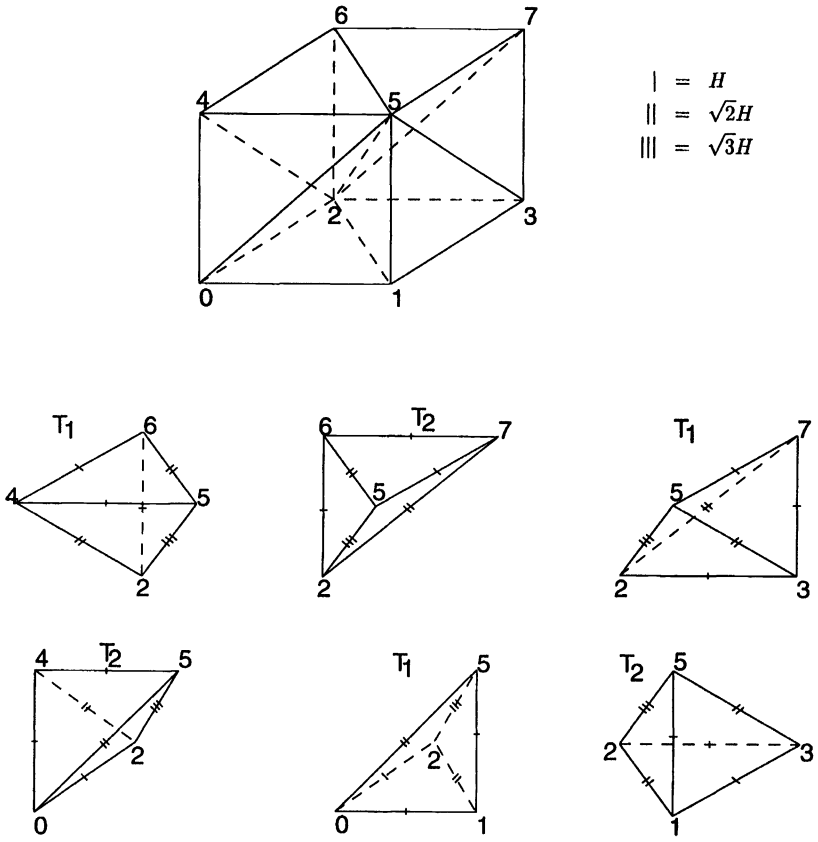


FIG. 1. *Six tetrahedra of two types, T_1 and T_2, in a cube.*

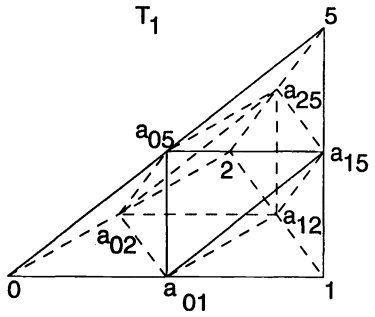


FIG. 2. *Connect midpoints of the model tetrahedron T_1.*

denote by $T_i^{(k)}$ the small tetrahedron at level k of refinement that is similar to T_i, $i = 1, 2$. This implies that T_i and $T_i^{(k)}$ have the same angles between faces and if T_i has diameter h_{T_i}, then $T_i^{(k)}$ has diameter $h_{T_i}/2^k$ with uniform refinement.

The first step is to connect the midpoints of the edges of the tetrahedron as shown in Fig. 2. This gives four tetrahedra—t_1, t_2, t_3, and t_4—formed by chopping off the four corners of the tetrahedron T_1, and an octahedron p in the middle section of T_1. See Fig. 3. In three different orientations, the octahedron p can be viewed as two skewed pyramids patched together on a common base.

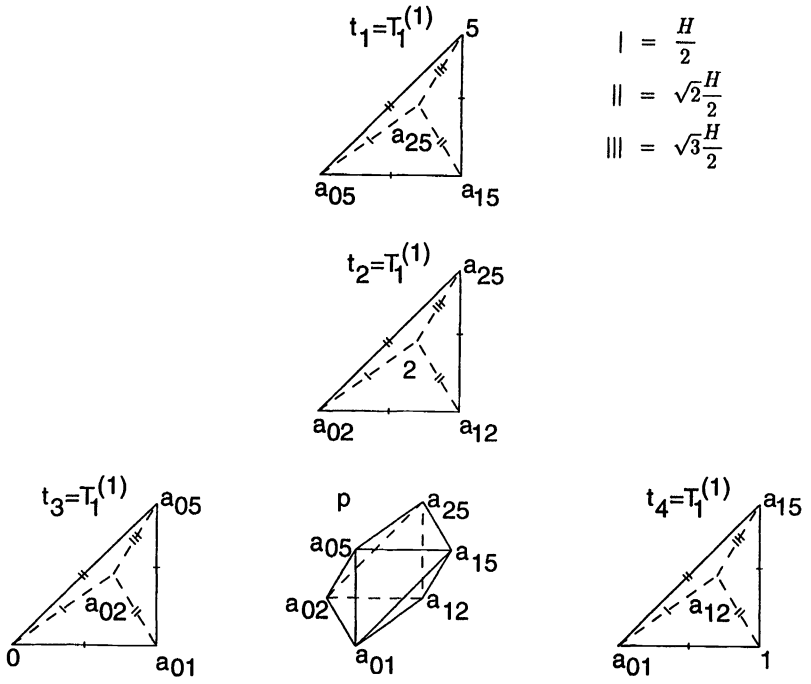


FIG. 3. *Four tetrahedra t_1, t_2, t_3, t_4, and octahedron p.*

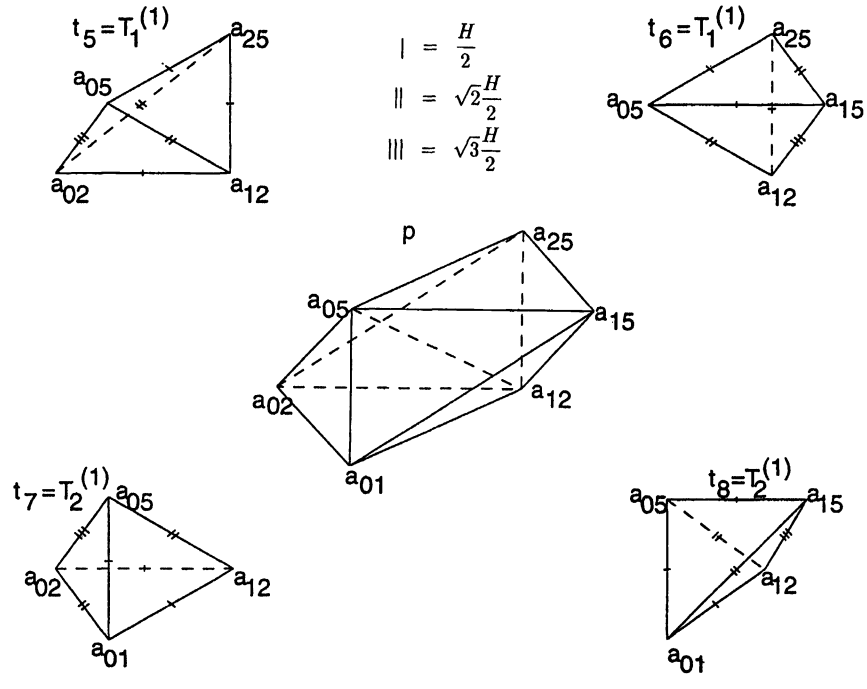


FIG. 4. *Four tetrahedra t_5, t_6, t_7, t_8 obtained from octahedron p when nodes a_{05} and a_{12} are connected to form a diagonal.*

The choice of a diagonal that connects two of the four base nodes determines the four other tetrahedra—t_5, t_6, t_7, and t_8—obtained from the octahedron p. There are three possible diagonals and hence three possible sets of four tetrahedra that can be generated from the octahedron p. The three diagonals are formed by connecting the following pairs of nodes: a_{05} and a_{12} (shown in Fig. 4), a_{25} and a_{01}, and a_{02} and a_{15}. Regardless of which diagonal is chosen, we are able to refine the tetrahedron T_1 into eight small tetrahedra. The choice of a particular diagonal will be determined when we refer back to the cube.

The eight tetrahedra for any choice of diagonal have the same volume. We show this for the case where the diagonal is formed by connecting nodes a_{05} and a_{12} as shown in Fig. 4. The model tetrahedron T_1 has diameter $h_{T_1} = \sqrt{3}H$ and volume $V_{T_1} = \frac{1}{6}V = \frac{1}{6}H^3$. Note that the four small tetrahedra t_1, t_2, t_3, t_4 shown in Fig. 3 are similar to the model tetrahedron T_1. Moreover, the tetrahedra t_5 and t_6 from the octahedron p in Fig. 4 are similar to T_1 and the tetrahedra t_7 and t_8 are similar to T_2. The only difference is that these eight small tetrahedra have diameters $\frac{\sqrt{3}H}{2}$, half that of either T_1 or T_2. Hence, each of these eight small tetrahedra has volume $\frac{1}{6}\left(\frac{H}{2}\right)^3 = \frac{1}{8}V_{T_1}$.

Observe that the four tetrahedra obtained from the octahedron p by connecting any two base nodes come in pairs. Each pair consists of either two identical tetrahedra or two tetrahedra that are reflections of each other. With this observation and using (2.4), it can be shown easily that each of the four tetrahedra in p obtained by connecting nodes a_{25} and a_{01} or nodes a_{02} and a_{15} has volume equal to $\frac{1}{8}V_{T_1}$. Hence all the eight tetrahedra making up the model tetrahedron T_1, regardless of the diagonal chosen, have the same volume.

We extend this refinement procedure to an arbitrary tetrahedron. We note that any nondegenerate tetrahedron T can be mapped to the model tetrahedron T_1; that is, for any nondegenerate tetrahedron T, there exists a *unique invertible affine mapping* [1], [2], [13]

$$\mathrm{F}_1 : \boldsymbol{x} \in \Re^3 \mapsto \mathrm{F}_1(\boldsymbol{x}) = B_1\boldsymbol{x} + \boldsymbol{b}_1, \tag{2.5}$$

where B_1 is an invertible 3×3 matrix and $\boldsymbol{b}_1$ is a vector in $\Re^3$ such that

$$\mathrm{F}(\hat{\imath}) = i. \tag{2.6}$$

Here, $\hat{\imath}$ and i are the four vertices of T_1 and corresponding vertices of T, respectively. Since midpoints are preserved by the affine mapping, it follows that

$$\mathrm{F}(\hat{m}) = m, \tag{2.7}$$

where $\hat{m}$ and m are the midpoints of T_1 and T, respectively. By property (2.7) of the affine mapping, midpoints of the model tetrahedron T_1 will map to midpoints of any nondegenerate tetrahedron T. Moreover, we have the following differential volume relations:

$$dx\,dy\,dz = |\det(B_1)|\,d\hat{x}\,d\hat{y}\,d\hat{z}, \tag{2.8}$$

where $\det(B_1)$ is the determinant of the affine mapping matrix B_1 and is constant by the property of the affine mapping. Hence, *any tetrahedron can be refined into eight tetrahedra of equal volume by connecting the midpoints of its edges.*

Recall that we have three choices of diagonals when refining any tetrahedron in the cube. We will choose a particular diagonal so that we maintain the tetrahedral structure in the refinement. In other words, if we refine the cube uniformly into eight small cubes, each of the small cubes will have a tetrahedral structure identical to that of the big cube shown in Fig. 1. We illustrate this in Fig. 5. Such a restriction forces the small tetrahedra to be *similar* to one of the two parent model tetrahedra T_1 and T_2. The diagonal chosen and the refinement of the

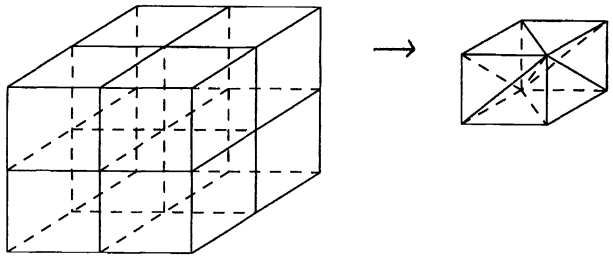


FIG. 5. *Tetrahedral structure preserved in small cubes.*

model tetrahedron T_1 are shown in Figs. 2–4. Notice the *inclusion* of the small tetrahedra in their parent tetrahedron. Thus, *the nesting condition is satisfied.*

This procedure is repeated at each level of refinement. A tetrahedron is refined into 2^{3j} tetrahedra after j levels of uniform refinement. *By the uniform refinement strategy, any tetrahedron can be refined at level j into* 2^{3j} *tetrahedra that are nested and have equal volumes.* Notice that after one level of refinement, the cube has $(2^1 + 1)^3 = 27$ nodes. In general, we have $(2^j + 1)^3 = N$ nodes after j levels of uniform refinement.

To measure *nondegeneracy of the model tetrahedra* T_1 *and* T_2, we use Zhang's formula [13] to compute the diameter ρ_T of the largest sphere inscribed in a tetrahedron T. This is given by

$$\rho_T = 6\frac{V_T}{S_T}, \tag{2.9}$$

where V_T is the volume and S_T is the surface area of tetrahedron T. The surface area S_T is the sum of the area of the triangles that make up the faces of tetrahedron T. The volume of either T_1 or T_2 is $\frac{1}{6}H^3$ and it can be calculated easily that the surface area of either tetrahedron is $(1 + \sqrt{2})H^2$. Hence,

$$\rho_{T_2} = \rho_{T_1} = \frac{H}{1 + \sqrt{2}}. \tag{2.10}$$

Since the diameter of either T_1 or T_2 is $\sqrt{3}H$, then (1.1) and (2.10) give the following measures of nondegeneracy of the model tetrahedra T_1 and T_2:

$$\sigma_{T_2} = \sigma_{T_1} = \frac{h_{T_1}}{\rho_{T_1}} = \sqrt{3}(1 + \sqrt{2}) \approx 4.18. \tag{2.11}$$

It follows that the six tetrahedra, of types T_1 and T_2, in the initial triangulation of the cube are nondegenerate.

The *uniformity of the model tetrahedra* T_1 *and* T_2 follows from the fact that T_1 and T_2 are reflections of each other, and therefore have the same diameter. Hence,

$$\frac{h_{T_i}}{h_{T_l}} = 1, \quad 1 \le i \le 2, \quad 1 \le l \le 2. \tag{2.12}$$

Thus, the six tetrahedra in the initial triangulation of the cube are uniform.

Since the small tetrahedra into which T_1 and T_2 are refined are similar to these parent tetrahedra, the measures of nondegeneracy and uniformity remain the same for all k; that is,

$$\sigma_{T_i^{(k)}} = \sigma_{T_i}, \qquad i = 1, 2, \tag{2.13}$$

and

$$\frac{h_{T_i^{(k)}}}{h_{T_l^{(k)}}} = \frac{h_{T_i}}{h_{T_l}} = 1, \quad 1 \le i \le 2, \quad 1 \le l \le 2. \tag{2.14}$$

(Recall that after k levels of refinement, $k = 1, 2, \ldots$, we denote by $T_i^{(k)}$ the small tetrahedra similar to T_i, $i = 1, 2$.) Because of the properties given in (2.13) and (2.14), it follows that *the uniform refinement strategy for the model tetrahedra in the cube is nondegenerate and uniform.*

In the next section we show that when the uniform refinement strategy is applied to any nondegenerate tetrahedron, the resulting tetrahedra in the refinement are nondegenerate and quasi uniform.

3. Extensions to arbitrary tetrahedra. In the previous section, we described a uniform and nondegenerate refinement of the model tetrahedra T_1 and T_2. We have shown that with uniform refinement, any tetrahedron can be refined at level j into 2^{3j} tetrahedra that are equivolume and nested. We now show that *for any nondegenerate tetrahedron T, there exists a refinement that is nondegenerate and quasi uniform.*

We denote by F_i the affine mapping that maps the model tetrahedron T_i, $i = 1, 2$, to an arbitrary nondegenerate tetrahedron T:

$$T = \mathrm{F}_i(T_i) : \boldsymbol{x} \in T_i \mapsto \mathrm{F}_i(\boldsymbol{x}) = B_i\boldsymbol{x} + \boldsymbol{b}_i \in T. \tag{3.15}$$

We proceed to prove the following theorem, which shows nondegeneracy of the refinement of T into 2^{3k} small tetrahedra at level k, where T is the image of any of the model tetrahedra T_i, $i = 1, 2$. This theorem is similar to Theorem 2.2.8 in Zhang's thesis [13].

THEOREM 3.1. *For any nondegenerate tetrahedron T with*

$$\sigma_T \le c_0,$$

where σ_T is given by (1.1), *there exists a refinement of T into 2^{3k} small tetrahedra $\{T_{s,\ell}^{(k)}\}_{\ell=1}^{2^{3k}}$ for levels $k = 1, 2, \ldots$ such that*[2]

$$\sigma_{T_{s,\ell}^{(k)}} \le c_0\sigma_{T_1}^2, \quad \ell = 1, \ldots, 2^{3k} \quad \textit{for all } k.$$

Proof. We prove the case where T is the image of the model tetrahedron T_1 shown in Fig. 2, where we have the following affine mapping from (3.15):

$$T = \mathrm{F}_1(T_1) : \boldsymbol{x} \in T_1 \mapsto \mathrm{F}_1(\boldsymbol{x}) = B_1\boldsymbol{x} + \boldsymbol{b}_1 \in T. \tag{3.16}$$

(The case where T is the image of the model tetrahedron T_2 is proved similarly.) We use the relations found in Theorem 3.1.3 in Ciarlet [2], namely,

$$\|B_1\| \le \frac{h_T}{\rho_{T_1}}, \qquad \|B_1^{-1}\| \le \frac{h_{T_1}}{\rho_T}, \tag{3.17}$$

where $\|\cdot\|$ denotes the Euclidean norm.

Let T and T_1 be refined into 2^{3k} tetrahedra $\{T_{s,\ell}^{(k)}\}_{\ell=1}^{2^{3k}}$ and $\{T_{1:s,\ell}^{(k)}\}_{\ell=1}^{2^{3k}}$, respectively, at level k. In the case where the small tetrahedron $T_{1:s,l}^{(k)}$ is similar to T_i, $i = 1$ or 2, we have

$$\begin{aligned} h_{T_{1:s,l}^{(k)}} &= \frac{h_{T_i}}{2^k}, \\ \rho_{T_{1:s,l}^{(k)}} &= \frac{\rho_{T_i}}{2^k}. \end{aligned} \tag{3.18}$$

[2]The subscript s is used to denote *small*.

If $T_{s,\ell}^{(k)}$ is the image of $T_{1:s,\ell}^{(k)}$, which is similar to T_i, we have

$$\begin{aligned}\sigma_{T_{s,\ell}^{(k)}} = \frac{h_{T_{s,\ell}^{(k)}}}{\rho_{T_{s,\ell}^{(k)}}} &\le \frac{\|B_1\| h_{T_{1:s,\ell}^{(k)}}}{\|B_1^{-1}\|^{-1} \rho_{T_{1:s,\ell}^{(k)}}} \\ &\le \frac{h_T}{\rho_{T_1}} \frac{h_{T_1}}{\rho_T} \frac{h_{T_i}}{\rho_{T_i}} \\ &= \sigma_T \sigma_{T_1} \sigma_{T_i}.\end{aligned} \tag{3.19}$$

Since $\sigma_{T_1} = \sigma_{T_2}$, we obtain the desired result. □

In the following theorem, we show that *for any nondegenerate tetrahedron T, there exists a quasi-uniform refinement of T into 2^{3k} tetrahedra at level k for $k = 1, 2, \ldots$.*

THEOREM 3.2. *For any nondegenerate tetrahedron T with*

$$\sigma_T \le c_0$$

there exists a refinement of T into 2^{3k} small tetrahedra $\{T_{s,\ell}^{(k)}\}_{\ell=1}^{2^{3k}}$ for levels $k = 1, 2, \ldots$, such that

$$\frac{h_{T_{s,m}^{(k)}}}{h_{T_{s,n}^{(k)}}} \le c_0 \sigma_{T_1}, \quad m, n = 1, \ldots, 2^{3k} \quad \text{for all } k.$$

Proof. Let T be the image of the model tetrahedron T_1, where the affine mapping is given in (3.16) and the relations in (3.17) hold. (The case where T is the image of T_2 is proved similarly.) Let T be refined into 2^{3k} tetrahedra $\{T_{s,\ell}^{(k)}\}_{\ell=1}^{2^{3k}}$ and let T_1 be refined into 2^{3k} tetrahedra $\{T_{1:s,\ell}^{(k)}\}_{\ell=1}^{2^{3k}}$ at level k. Let $T_{s,m}^{(k)}$ and $T_{s,n}^{(k)}$ be the image of $T_{1:s,m}^{(k)}$ and $T_{1:s,n}^{(k)}$, respectively, where $m, n \in \{1, \ldots, 2^{3k}\}$. Let $T_{1:s,m}^{(k)}$ and $T_{1:s,n}^{(k)}$ be similar to the model tetrahedra T_i and T_l, respectively, where $i, l = 1, 2$, so that $T_{1:s,m}^{(k)} = T_i^{(k)}$ and $T_{1:s,n}^{(k)} = T_l^{(k)}$. Using (2.14), (3.17), and (3.18), we have for all k

$$\begin{aligned}\frac{h_{T_{s,m}^{(k)}}}{h_{T_{s,n}^{(k)}}} \le \frac{\|B_1\| h_{T_{1:s,m}^{(k)}}}{\|B_1^{-1}\|^{-1} h_{T_{1:s,n}^{(k)}}} &= \frac{\|B_1\| h_{T_i^{(k)}}}{\|B_1^{-1}\|^{-1} h_{T_l^{(k)}}} \\ &\le \frac{h_T}{\rho_{T_1}} \frac{h_{T_1}}{\rho_T} \frac{h_{T_i^{(k)}}}{h_{T_l^{(k)}}} \\ &\le \sigma_T \sigma_{T_1} \\ &\le c_0 \sigma_{T_1} \qquad m, n = 1, \ldots, 2^{3k}. \quad \square\end{aligned}$$

COROLLARY 3.1. *At the initial refinement $\mathcal{T}_0$, let the polygonal domain Ω be partitioned into blocks, each containing six tetrahedra, where each block is the image of the cube containing six tetrahedra of two model types T_1 and T_2 shown in Fig. 1. If*

$$\sigma_T \le c_0 \qquad \forall\, T \in \mathcal{T}_0,$$

then there exists a refinement wherein each $T \in \mathcal{T}_0$ is refined into 2^{3k} tetrahedra $\{T_{s,l}^{(k)}\}_{l=1}^{2^{3k}}$ at level $k = 1, 2, \ldots$, such that

$$\begin{aligned}&\sigma_{T_{s,l}^{(k)}} \le c_0 \sigma_{T_1}^2, \qquad l = 1, \ldots, 2^{3k}, \\ &\frac{h_{T_{s,m}^{(k)}}}{h_{T_{s,n}^{(k)}}} \le c_0 \sigma_{T_1}, \qquad m, n = 1, \ldots, 2^{3k}\end{aligned} \tag{3.20}$$

for all $T \in \mathcal{T}_0$ and all levels k. Here, $T_{s,1}^{(0)} = T \in \mathcal{T}_0$, which is not refined.

Proof. This follows from Theorems 3.1 and 3.2. □

4. Applications and limitations. We have described a uniform refinement strategy for any nondegenerate tetrahedron T that results in a triangulation that is *nested* and generates, at level k, 2^{3k} tetrahedra that are *equivolume*. The refinement strategy, which employs the model cube containing six tetrahedra of two types—T_1 and T_2—is *quasi uniform* and *nondegenerate*. (Though refinement of the six tetrahedra in the cube generates small tetrahedra similar to the initial six tetrahedra, this is not true of the refinement of arbitrary tetrahedra.) This strategy applied to the two model tetrahedra happens to satisfy Zhang's rule of choosing the shortest interior edge [13], that is, the diagonal with the shortest length, even though the rule is not invoked in choosing the diagonal. Recall that the strategy chooses the diagonal so that the tetrahedral structure in the cubes is preserved. This strategy applied to an arbitrary tetrahedron also satisfies Zhang's rule of labeled edge subdivision [13], where the diagonal connects the midpoints of the labeled edge (arbitrarily chosen at the initial refinement and determined by the rule thereafter) and the edge opposite it. This implies that *the uniform refinement strategy applied to an arbitrary tetrahedron will generate at most six types of tetrahedra during successive refinement.* This holds true for any initial labeling of the arbitrary tetrahedron and, therefore, for any affine mapping from the model tetrahedron to the arbitrary tetrahedron.

The refinement strategy is *easy to implement and can be automated.* A polygonal domain Ω can be partitioned into blocks, each block partitioned into eight blocks at each level of refinement. When partitioning the block, the midpoints of the six tetrahedra in the block that are images of the six model tetrahedra in the cube must be used. At the final level of refinement, say level j, six tetrahedra will be fitted into each of the 8^j blocks in the same manner as six model tetrahedra were fitted into the model cube. This achieves the same triangulation as when each of the initial blocks in the domain Ω are fitted with six tetrahedra and the tetrahedra refined according to the uniform refinement strategy. This can be seen easily by assembling the refined model tetrahedra into a cube.

The refinement strategy has *applications to finite element methods* in general. A typical concern in using tetrahedral elements is the generation of degenerate tetrahedra in the refinement, which leads to poorly conditioned systems of equations [5], [12]. The nondegeneracy of the refinement strategy avoids this problem.

The refinement yields a *nested triangulation* that can be used for multigrid and other nested multilevel methods [6], [10]. The optimal operation count of $O(N)$, where N is the number of unknowns, is likewise obtained. This is shown easily by summing over all levels the number of unknowns or nodes in the cube, which is the same number of nodes in the blocks making up the domain Ω. That is,

$$\sum_{k=0}^{j}(2^k + 1)^3 = O(N)$$

since $(2^j + 1)^3 = N$.

There are some *limitations* to this refinement strategy. First, the *nondegeneracy* and *quasi-uniform bounds* obtained using the model tetrahedra in the cube are *not optimal.* A refinement using the model tetrahedra in [13] produces smaller bounds. However, the tetrahedra in [13] cannot be used to triangulate a cube, a tool that greatly simplifies the refinement of tetrahedra.

Second, the *coefficient matrix* associated with the tetrahedral refinement is *not relatively sparse.* There can be as many as 15 nonzeros in each row of the matrix as opposed to 7 nonzeros in the usual 7-point discretization of a second order differential operator in three dimensions. Figure 6 shows the connectivity of 14 nodes to a center node in a cube partitioned into six tetrahedra of types T_1 and T_2 and refined according to the uniform refinement strategy.

An interesting question is how many ways can we triangulate a cube into tetrahedra and which of these triangulations is amenable to the uniform refinement strategy. The refinement

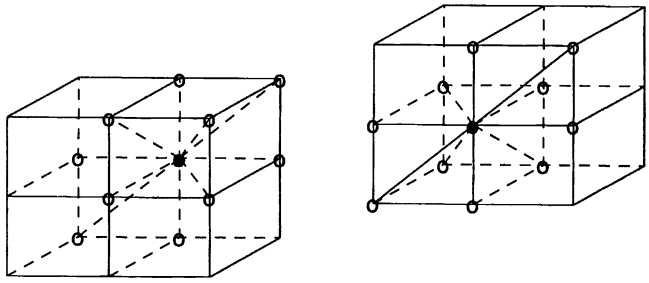


FIG. 6. *Connectivity of a center node.*

strategy applied to a cube generates tetrahedra that are nested, equivolume, similar to the model tetrahedra, and conforming so that the tetrahedral elements match at block interfaces. Moreover, small cubes into which a parent cube is partitioned have the same tetrahedral structure as the parent cube.

A theorem in [7] states that there exist exactly ten essentially different vertex-true triangulations of a cube into tetrahedra. A vertex-true triangulation requires that each vertex of a tetrahedron in the cube is a vertex of the cube. These triangulations are classified into six cases depending on the number k_m of tetrahedra with $m = 0, 1, 2, 3$ external faces on the cube. The triangulation of the cube presented here is the case with $k_0 = 0, k_1 = 0, k_2 = 6$, and $k_3 = 0$ since there are six tetrahedra (of types T_1 and T_2) with two faces on the cube. Another triangulation of the cube that is amenable to the refinement strategy is the case with $k_0 = 0, k_1 = 2, k_2 = 2$, and $k_3 = 2$ as shown in Fig. 7. This triangulation has two tetrahedra with one face on the cube, two tetrahedra (types T_1 and T_2) with two faces on the cube, and two tetrahedra with three faces on the cube. Another realization of this case is shown in Fig. 8 and appears in [14] and is basically a different choice of diagonal. This case is studied in detail in [11]. In all three cases represented by Figs. 1, 7, and 8, a key thing to note is that the diagonals on parallel faces of the cube align. This guarantees that the tetrahedra match at block interfaces. Again, uniform refinement strategies that have the properties of the uniform refinement strategy discussed in the previous sections can be defined for the cases represented in Fig. 7 and 8. These strategies will be nondegenerate and quasi uniform. The tetrahedra generated from the refinement will be equivolume and nested. The small cubes generated from the refinement will have the same tetrahedral structure as the parent cube. This refinement strategy can also be automated by working with the cubes.

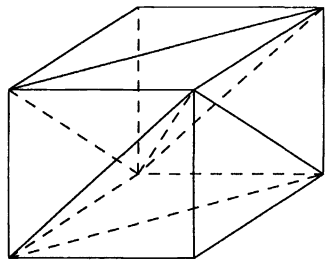


FIG. 7. *Triangulation of a cube,* $k_0 = 0, k_1 = 2, k_2 = 2, k_3 = 2$.

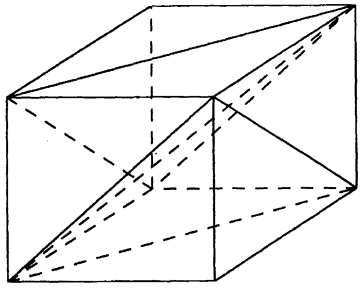


FIG. 8. *Triangulation of a cube,* $k_0 = 0, k_1 = 2, k_2 = 2, k_3 = 2$.

Acknowledgement. I wish to thank the referees for their valuable suggestions.

REFERENCES

[1] E. B. BECKER, G. F. CAREY AND J. T. ODEN *Finite Elements, An Introduction*, Vol. I, Prentice Hall, Englewood Cliffs, NJ, 1981.

[2] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland Publishing Company, New York, 1978.

[3] H. S. M. COXETER, *Regular Polytopes*, The Macmillan Company, New York, Second ed., 1963.

[4] W. A. DAHMEN AND C. A. MICCHELLI, *On the Linear Independence of Multivariate B-Splines* I. *Triangulations of Simploids*, SIAM J. Numer. Anal., 19 (1982), pp. 993–1012.

[5] I. FRIED, *Condition of finite element matrices generated from nonuniform meshes*, AIAA J., 10 (1972), pp. 219–221.

[6] W. HACKBUSCH, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[7] J. BÖHM, *Some problems of triangulating polytopes in Euclidean d-space*, *Forschungsergebnisse* N/88/17, Sektion Mathematik der Friedrich-Sciller-Universität Jena, Germany, 1988.

[8] H. W. KUHN, *Some combinatorial lemmas in topology*, IBM J. Res. Develop., 45 (1960), pp. 518–524.

[9] P. S. MARA, *Triangulation for the cube*, J. Combin. Theory Ser. A, 20 (1976), pp. 170–177.

[10] S. F. MCCORMICK, ED., *Multigrid Methods*, in Frontiers in Applied Mathematics, Vol. 3, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987.

[11] M. E. G. ONG, *Hierarchical Basis Preconditioners for Second Order Elliptic Problems in Three Dimensions*, Ph.D. thesis, Dept. of Applied Mathematics, University of Washington, October 1989; Tech. Report 89-3, Dept. of Applied Mathematics, University of Washington, 1989.

[12] G. STRANG AND G. J. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[13] S. ZHANG, *Multi-level Iterative Techniques*, Ph.D. thesis, Dept. of Mathematics, The Pennsylvania State University, University Park, PA, August 1988.

[14] O. C. ZIENKIEWICZ, *The Finite Element Method*, McGraw-Hill Book Company (UK) Limited, London, 1977.

[15] M. J. TODD, *The Computation of Fixed Points and Applications*, Lecture Notes in Economics and Mathematical Systems 124, Springer-Verlag, Berlin, 1976.

[16] ———, *Optimal dissection of simplices*, SIAM J. Appl. Math., 34 (1978), pp. 792–803.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1145–1173, September 1994

011

THE LAGUERRE ITERATION IN SOLVING THE SYMMETRIC TRIDIAGONAL EIGENPROBLEM, REVISITED *

T. Y. LI† AND ZHONGGANG ZENG‡

Abstract. This paper presents an algorithm for the eigenvalue problem of symmetric tridiagonal matrices. The algorithm employs the determinant evaluation, split-and-merge strategy, and the Laguerre iteration. The method directly evaluates eigenvalues and uses inverse iteration as an option when eigenvectors are needed. This algorithm combines the advantages of existing algorithms such as QR, bisection/multisection, and Cuppen's divide-and-conquer method. It is fully parallel and competitive in speed with the most efficient QR algorithm in serial mode. On the other hand, the algorithm is as accurate as any standard algorithm for the symmetric tridiagonal eigenproblem and enjoys the flexibility in evaluating partial spectrum.

Key words. eigenvalue, Laguerre's iteration, symmetric tridiagonal matrix

AMS subject classification. 65F15

1. Introduction.

For a symmetric tridiagonal matrix T with nonzero subdiagonal entries, the eigenvalues of T or the zeros of its characteristic polynomial

$$f(\lambda) = \det[T - \lambda I] \tag{1}$$

are all real and simple. Therefore, the globally convergent Laguerre iteration

$$L_{\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{(n-1)\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}} \tag{2}$$

with cubic convergence rate for solving a polynomial equation with real and simple zeros seems perfectly suitable for finding eigenvalues of T. This method was mentioned many years ago in several places (e.g., see Wilkinson [28, pp. 443–445]). There is also literature about the Laguerre iteration for nonsymmetric eigenvalue problems [23], [27]. However, a serious investigation of the practicality of the method, such as efficiency and accuracy, and an attempt to efficiently implement the algorithm have not yet been done for the symmetric tridiagonal eigenproblem. The purpose of this paper is to revisit this method with an intensive examination of the theory and practicality and make an effort to fully implement the algorithm efficiently. The proposed algorithm evaluates eigenvalues of a symmetric tridiagonal matrix without computing its eigenvectors. In case eigenvectors are also needed, the inverse iteration can be applied [14].

First, to use the Laguerre iteration (2) for finding zeros of the polynomial f in (1), one needs to evaluate f, f', and f''. Those values can be efficiently evaluated by certain three-term recurrence relations (see §2). It is well known that those three-term recurrences may suffer from a severe underflow-overflow problem. In §2, we propose an alternative scheme, which can virtually avoid underflow-overflow problems. The accuracy of our scheme will also be analyzed.

To find some or all of the eigenvalues of T by the Laguerre iteration, a set of proper starting points is essential. For this purpose, our split-merge process proposed in §3, similar

*Received by the editors October 14, 1992; accepted for publication (in revised form) August 10, 1993. This research was supported in part by the National Science Foundation under grant CCR-9024840.

†Department of Mathematics, Michigan State University, East Lansing, Michigan 48824-1027 (li@math.msu.edu).

‡Department of Mathematics, Michigan State University, East Lansing, Michigan 48824-1027 (zeng@math.msu.edu).

to Cuppen's divide-and-conquer strategy [4], [7], [11], [25], provides a close matrix $\hat{T}$ of T, whose eigenvalues constitute an excellent set of starting points. These starting points can lead to desired eigenvalues of T with about 2–3 steps of the Laguerre iteration in general.

The Laguerre iteration converges only linearly when it is used to approximate zeros with multiplicity $r > 1$. Although all the eigenvalues of T are simple, some of them may be indistinguishable numerically. In the actual computation, the Laguerre iteration also converges linearly in this case as if those eigenvalues, say, r of them, were a multiple eigenvalue with multiplicity r. Slow convergence also occurs when a group of r eigenvalues are relatively close to each other compared to the distance from the starting point of the iteration. In all those situations, the modified Laguerre iteration

$$L_{r\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{\frac{n-r}{r}\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}}$$

can be used to speed up the process and maintain the cubic convergence rate. The behavior of the modified Laguerre iteration will be studied in §2.2. The estimation of the number r, which plays an important role in practice, can be obtained from the information of the closeness of the eigenvalues of the matrix $\hat{T}$ in our split-merge process, and an algorithm is given in Fig. 3.

Homotopy continuation methods have recently been developed for algebraic eigenvalue problems, and very encouraging results have been obtained for both the symmetric and non-symmetric cases [12], [16], [17], [19]–[21], [29]. While our algorithm does not employ the homotopy continuation method directly, the design of our algorithm is critically based on the consideration of the structure of the eigenvalue curves of the homotopy

$$h(t, \lambda) = \det[(1-t)\hat{T} + tT - \lambda I].$$

The symmetric tridiagonal eigenvalue problem has been intensively studied, and hence forms an excellent testing ground for newly developed algorithms. There are quite a few reliable algorithms for this problem with various features, such as QR (or QL) [8], D&C (Cuppen's divide-and-conquer) [4], [7], and B/M (bisection/multisection) [13], [22]. A detailed comparison of these algorithms can be found in [5]. In summary, the QR algorithm is believed to be the most efficient algorithm on serial machines for matrices of moderate sizes. D&C is an excellent parallel algorithm. B/M, which is also parallelizable, leads in accuracy and possesses the flexibility in the evaluation of partial spectrum. When only eigenvalues are needed, both QR and B/M are able to evaluate eigenvalues without computing eigenvectors. The promise of our algorithm is that it contains most of the advantages of these algorithms:

(i) the same parallel structure as D&C but less memory contention;

(ii) the same accuracy as B/M;

(iii) the same flexibility as B/M in evaluating partial spectrum;

(iv) the same advantage as QR and B/M in separating the evaluation of eigenvalues and eigenvectors.

In §5, comprehensive numerical results comparing our algorithm with these methods on a substantial variety of types of matrices are presented. It appears that, with our stopping criterion, our algorithm achieves the best accuracy in evaluating eigenvalues. The speed is competitive with the QR algorithm in serial mode and leads B/M as well as D&C by a considerable margin. When only a small fraction of the total number of eigenvalues are in demand, our algorithm can take full advantage of this situation by reducing the execution time to a similar fraction.

Modern scientific computing is marked by the advent of vector and parallel computers and by the search for algorithms that are to a large extent parallel in nature. An important advantage

of our algorithm is its natural parallelism, in the sense that each eigenvalue is computed totally independently of the others. Besides, our algorithm has a great capacity for vectorization. These considerations will be discussed in §6 and an intensive experiment will be reported in a separate article [26].

The code of our algorithm is available electronically from Zhonggang Zeng.

2. The Laguerre iteration.

2.1. Evaluation of the determinant and its accuracy. Let T be a symmetric tridiagonal matrix of the form

$$T = [\beta_{i-1}, \alpha_i, \beta_i] = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix}. \tag{3}$$

Without loss of generality, we may assume T is unreduced; that is, $\beta_i \neq 0$, $i = 1, \ldots, n-1$. For unreduced T, the characteristic

$$f(\lambda) \equiv \det(T - \lambda I) \tag{4}$$

is a polynomial with only *real* and *simple* zeros ([28, p. 300]). Let

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n$$

be the zeros of f. Set $\lambda_0 = -\infty$ and $\lambda_{n+1} = +\infty$. For $x \in (\lambda_0, \lambda_{n+1})$, the Laguerre iterates $L_+(x)$ and $L_-(x)$ are defined as follows:

$$L_\pm(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{(n-1)\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}}. \tag{5}$$

The following result is known ([28, pp. 443–445]).

PROPOSITION 2.1. *For $x \in (\lambda_m, \lambda_{m+1})$, $m = 0, 1, \ldots, n$, the following results hold:*

1. $\lambda_m < L_-(x) < x < L_+(x) < \lambda_{m+1}$.
2. *There are constants c_+ and c_- such that*

$$|L_+(x) - \lambda_{m+1}| \leq c_+ |x - \lambda_{m+1}|^3 \quad \text{if } x \text{ is close to } \lambda_{m+1},$$

$$|L_-(x) - \lambda_m| \leq c_- |x - \lambda_m|^3 \quad \text{if } x \text{ is close to } \lambda_m.$$

From this proposition, if we let

$$x_+^{(k)} = L_+^k(x) \equiv \overbrace{L_+(L_+(\cdots L_+}^{k}(x)\cdots)) \quad \text{and}$$

$$x_-^{(k)} = L_-^k(x) \equiv \overbrace{L_-(L_-(\cdots L_-}^{k}(x)\cdots)),$$

then

$$\lambda_m \longleftarrow \cdots x_-^{(2)} < x_-^{(1)} < x < x_+^{(1)} < x_+^{(2)} \cdots \longrightarrow \lambda_{m+1}.$$

The two sequences $x_+^{(k)}$ and $x_-^{(k)}$ converge monotonically to λ_m and λ_{m+1}, respectively, in asymptotically cubic rate. That is, they eventually enter certain neighborhoods of their corresponding limits and then exhibit a cubic convergence rate.

To use Laguerre's iteration (5) for finding zeros of $f(\lambda)$ in (4), it is necessary to evaluate f, f', and f'' efficiently with satisfactory accuracy. It is well known that the characteristic polynomial $f(\lambda) \equiv \det(T - \lambda I)$ of $T = [\beta_{i-1}, \alpha_i, \beta_i]$ and its derivatives with respect to λ can be evaluated by three-term recurrences ([28, p. 423]):

$$\text{(6)} \qquad \begin{cases} \rho_0 = 1, \quad \rho_1 = \alpha_1 - \lambda, \\ \rho_i = (\alpha_i - \lambda)\rho_{i-1} - \beta_{i-1}^2 \rho_{i-2}, \qquad i = 2, 3, \ldots, n; \end{cases}$$

$$\text{(7)} \qquad \begin{cases} \rho_0' = 0, \quad \rho_1' = -1, \\ \rho_i' = (\alpha_i - \lambda)\rho_{i-1}' - \rho_{i-1} - \beta_{i-1}^2 \rho_{i-2}', \qquad i = 2, 3, \ldots, n; \end{cases}$$

$$\text{(8)} \qquad \begin{cases} \rho_0'' = 0, \quad \rho_1'' = 0, \\ \rho_i'' = (\alpha_i - \lambda)\rho_{i-1}'' - 2\rho_{i-1}' - \beta_{i-1}^2 \rho_{i-2}'', \qquad i = 2, 3, \ldots, n; \end{cases}$$

and

$$f(\lambda) = \rho_n, \qquad f'(\lambda) = \rho_n', \qquad f''(\lambda) = \rho_n''.$$

However, these recurrences may suffer from a severe underflow-overflow problem and require constant testing and scaling. We thus propose the following alternative scheme, which scales ρ_i by ρ_{i-1}. Let

$$\xi_i = \frac{\rho_i}{\rho_{i-1}}, \qquad i = 1, 2, \ldots, n.$$

Dividing both sides of (6) by ρ_{i-1} yields

$$\text{(9)} \qquad \begin{cases} \xi_1 = \alpha_1 - \lambda, \\ \xi_i = \alpha_i - \lambda - \dfrac{\beta_{i-1}^2}{\xi_{i-1}}, \quad i = 2, 3, \ldots, n. \end{cases}$$

We then scale ρ_i' and ρ_i'' by ρ_i by letting

$$\eta_i = -\frac{\rho_i'}{\rho_i} \quad \text{and} \quad \zeta_i = \frac{\rho_i''}{\rho_i}, \qquad i = 0, 1, \ldots, n.$$

Divide (7) and (8) by ρ_i and consider the obvious relation

$$\rho_i = \prod_{j=1}^{i} \xi_j.$$

We obtain the following scaled recurrences

$$\text{(10)} \qquad \begin{cases} \eta_0 = 0, \qquad \eta_1 = \dfrac{1}{\xi_1}, \\ \eta_i = \dfrac{1}{\xi_i}\left[(\alpha_i - \lambda)\eta_{i-1} + 1 - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}}\right)\eta_{i-2}\right], \qquad i = 2, 3, \ldots, n; \end{cases}$$

$$(11)\qquad \begin{cases} \zeta_0 = 0, \qquad \zeta_1 = 0, \\ \zeta_i = \dfrac{1}{\xi_i}\left[(\alpha_i - \lambda)\zeta_{i-1} + 2\eta_{i-1} - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}}\right)\zeta_{i-2}\right], \qquad i = 2, 3, \ldots, n; \end{cases}$$

and

$$-\frac{f'(\lambda)}{f(\lambda)} = \eta_n, \qquad \frac{f''(\lambda)}{f(\lambda)} = \zeta_n.$$

These are what one really needs in (5). Because of its self scaling, this process can avoid underflow-overflow problems except in extraordinarily unusual cases. However, the algorithm may break down if $\xi_i = 0$ for some $1 \le i \le n$. To prevent this, the following guardian is designed:

- if $\xi_1 = 0$ (i.e., $\alpha_1 = \lambda$), set $\xi_1 = \beta_1^2\varepsilon^2$;
- if $\xi_i = 0$, $i > 1$, set $\xi_i = (\beta_{i-1}^2\varepsilon^2)/\xi_{i-1}$;

where ε is the machine precision. In case of $\xi_i = 0$ ($i > 1$), the above remedy amounts to a small relative perturbation of β_{i-1}. More precisely, we simply replace β_{i-1} by $\beta_{i-1}(1-\varepsilon^2/2)$. For $\lambda = \alpha_1$, a perturbation $\beta_1^2\varepsilon^2$ is introduced upon α_1.

A determinant evaluation algorithm DETEVL (see Fig. 1) is formulated accordingly to the recurrences (9), (10), and (11). When ξ_i, $i = 1, \ldots, n$ are known, the Sturm sequence is available ([24, p. 47]). Thus, as a by-product, DETEVL also evaluates the number of eigenvalues of T that are less than λ. This number is denoted by $\kappa(\lambda)$.

Algorithm DETEVL:

input: $T = [\beta_{i-1}, \alpha_i, \beta_i]$, λ

output: $-\dfrac{f'(\lambda)}{f(\lambda)} = \eta_n$, $\dfrac{f''(\lambda)}{f(\lambda)} = \zeta_n$ (where $f(\lambda) \equiv \det(T - \lambda I)$) and $\kappa(\lambda) =$ *neg_count* (=the number of eigenvalues of T less than λ)

begin DETEVL

$\xi_1 = \alpha_1 - \lambda$

if $\xi_1 = 0$ then $\xi_1 = \beta_1^2\varepsilon^2$

$\eta_0 = 0$, $\eta_1 = \dfrac{1}{\xi_1}$, $\zeta_0 = 0$, $\zeta_1 = 0$, *neg_count* $= 0$

if $\xi_1 < 0$ then *neg_count* $= 1$

for $i = 2 : n$

$\xi_i = (\alpha_i - \lambda) - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}}\right)$

if $\xi_i = 0$ then $\xi_i = \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}}\right)\varepsilon^2$

if $\xi_i < 0$ then *neg_count* = *neg_count* $+1$

$\eta_i = \dfrac{1}{\xi_i}\left[(\alpha_i - \lambda)\eta_{i-1} + 1 - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}}\right)\eta_{i-2}\right]$

$\zeta_i = \dfrac{1}{\xi_i}\left[(\alpha_i - \lambda)\zeta_{i-1} + 2\eta_{i-1} - \left(\dfrac{\beta_{i-1}^2}{\xi_{i-1}}\right)\zeta_{i-2}\right]$

end for

end DETEVL

FIG. 1. *Algorithm* DETEVL.

To analyze the accuracy of this algorithm, we first assume the usual model of arithmetic:

$$fl(x \circ y) = (x \circ y) \cdot (1 + e), \tag{12}$$

where $\circ$ is one of the operations $+$, $-$, $\times$, and $/$; $fl(*)$ is the floating point result of $*$; and $|e| \leq \varepsilon$ is the machine precision. Then, from (9),

$$fl(\xi_1) = (\alpha_1 - \lambda)(1 + e_{11}) \quad \text{with } |e_{11}| < \varepsilon$$

and, for $i = 2, \ldots, n$,

$$\begin{aligned} fl(\xi_i) &= fl\left((\alpha_i - \lambda) - \frac{\beta_{i-1}^2}{fl(\xi_{i-1})}\right) \\ &= \left((\alpha_i - \lambda)(1 + e_{i1}) - \left(\frac{(\beta_{i-1}^2)(1 + e_{i3})}{fl(\xi_{i-1})}\right)(1 + e_{i4})\right)(1 + e_{i2}), \end{aligned}$$

where $|e_{im}| \leq \varepsilon$ for $i = 2, \ldots, n$ and $m = 1, 2, 3, 4$. So,

$$\frac{fl(\xi_1)}{1 + e_{11}} = \alpha_1 - \lambda$$

and, for $i = 2, \ldots, n$,

$$\frac{fl(\xi_i)}{(1 + e_{i1})(1 + e_{i2})} = (\alpha_i - \lambda) - \frac{\beta_{i-1}^2 \frac{(1+e_{i3})(1+e_{i4})}{(1+e_{i1})(1+e_{i-1,1})(1+e_{i-1,2})}}{\frac{fl(\xi_{i-1})}{(1+e_{i-1,1})(1+e_{i-1,2})}}.$$

Let

$$\varsigma_1 = \frac{fl(\xi_1)}{1 + e_{11}} \quad \text{and} \quad \varsigma_i = \frac{fl(\xi_i)}{(1 + e_{i1})(1 + e_{i2})}, \quad i = 2, \ldots, n.$$

Then

$$\begin{cases} \varsigma_1 = \alpha_1 - \lambda, \\ \varsigma_i = \alpha_i - \lambda - \frac{[\beta_{i-1}(1+e_{i-1})]^2}{\varsigma_{i-1}}, \qquad i = 2, 3, \ldots, n, \end{cases} \tag{13}$$

where

$$1 + \varepsilon_{i-1} = \left[\frac{(1 + e_{i3})(1 + e_{i4})}{(1 + e_{i1})(1 + e_{i-1,1})(1 + e_{i-1,2})}\right]^{1/2}. \tag{14}$$

With (9) in mind, the relations in (13) provide the recurrence for evaluating the determinant of the matrix

$$\begin{pmatrix} \alpha_1 - \lambda & \beta_1(1 + \varepsilon_1) & & & \\ \beta_1(1 + \varepsilon_1) & \alpha_2 - \lambda & \beta_2(1 + \varepsilon_2) & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2}(1 + \varepsilon_{n-2}) & \alpha_{n-1} - \lambda & \beta_{n-1}(1 + \varepsilon_{n-1}) \\ & & & \beta_{n-1}(1 + \varepsilon_{n-1}) & \alpha_n - \lambda \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix} + \begin{pmatrix} 0 & \beta_1\varepsilon_1 & & \\ \beta_1\varepsilon_1 & 0 & \ddots & \\ & \ddots & \ddots & \beta_{n-1}\varepsilon_{n-1} \\ & & \beta_{n-1}\varepsilon_{n-1} & 0 \end{pmatrix} - \lambda I$$

$$= (T + E) - \lambda I$$

with

$$E = \begin{pmatrix} 0 & \beta_1\varepsilon_1 & & \\ \beta_1\varepsilon_1 & 0 & \ddots & \\ & \ddots & \ddots & \beta_{n-1}\varepsilon_{n-1} \\ & & \beta_{n-1}\varepsilon_{n-1} & 0 \end{pmatrix}.$$

It follows from (14) that

$$-2.5\varepsilon + O(\varepsilon^2) \le \varepsilon_i \le 2.5\varepsilon + O(\varepsilon^2) \tag{15}$$

for $i = 1, \ldots, n$. If $\lambda = \alpha_1$, namely, the breakdown occurs at the beginning, then an additional backward error $\beta_1^2\varepsilon^2$ is added to the (1, 1) entry of E. A breakdown at the ith step makes a further $O(\varepsilon^2)$ perturbation on β_{i-1}.

Now, since $\prod_{i=1}^{n} \varsigma_i = \det[(T + E) - \lambda I]$, so,

$$fl[f(\lambda)] = fl[\det(T - \lambda I)] = fl\left[\prod_{i=1}^{n} fl(\xi_i)\right] = fl\left[\prod_{i=1}^{n} \varsigma_i(1 + e_{i1})(1 + e_{i2})\right]$$

$$= \prod_{i=1}^{n} \varsigma_i(1 + e_{i1})(1 + e_{i2})(1 + e_{i5}) = \det[(T + E) - \lambda I](1 + \gamma),$$

where

$$1 + \gamma = \prod_{i=1}^{n}(1 + e_{i1})(1 + e_{i2})(1 + e_{i5})$$

with $e_{12} = e_{15} = 0$ and $|e_{i5}| \le \varepsilon$, $i = 2, \ldots, n$. It can be seen easily that

$$-(3n - 2)\varepsilon + O(\varepsilon^2) \le \gamma \le (3n - 2)\varepsilon + O(\varepsilon^2).$$

By the Weyl inequality ([3, p. 34] or Lemma 6.1 in §6.1), if $\breve{\lambda}_1 < \breve{\lambda}_2 < \cdots < \breve{\lambda}_n$ are zeros of $\breve{f}(\lambda) \equiv (1 + \gamma)\det[(T + E) - \lambda]$, then

$$|\breve{\lambda}_i - \lambda_i| \le \|E\|_2 \le 2.5\varepsilon\left[\max_j(|\beta_j| + |\beta_{j+1}|)\right] + O(\varepsilon^2) \qquad i = 1, 2, \ldots, n,$$

where λ_i's are eigenvalues of T. Because the exact value of $\breve{\lambda}_i$ is generally not obtainable, the actual accuracy of the algorithm at λ_i, with $O(\varepsilon^2)$ omitted, can be

$$|fl(\breve{\lambda}_i) - \lambda_i| \le |\breve{\lambda}_i - \lambda_i| + |fl(\breve{\lambda}_i) - \breve{\lambda}_i| \le \frac{5\varepsilon}{2}\max_j(|\beta_j| + |\beta_{j+1}|) + |\lambda_i|\varepsilon. \tag{16}$$

The error analysis above, combined with the perturbation theory of Demmel and Kahan ([6, Thm. 2, p. 875]), leads to an important extension of our algorithm in the evaluation of singular values of bidiagonal matrices. Based on this algorithm and a hybrid strategy, the singular values of bidiagonal matrices can be evaluated within a tiny relative error [18].

Remark 2.2. Wilkinson [28, p. 303] analyzed the three-term recurrence (6) and proved that $\rho_n = \hat{f}(\lambda)$ where $\hat{f}(\lambda)$ is the exact characteristic polynomial of $S + \delta S$ with

$$\delta S = \begin{bmatrix} \delta\alpha_1 & \delta\beta_1 & & \\ \delta\beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \delta\beta_{n-1} \\ & & \delta\beta_{n-1} & \delta\alpha_n \end{bmatrix}$$

such that $|\delta\alpha_i| \leq 3.01\varepsilon(|\alpha_i| + |\lambda|)$ and $|\delta\beta_i| \leq 1.51\varepsilon|\beta_i|$. Apparently, the modified recurrence (9) can have an improved error bound (15). The same argument can also be applied to the bisection method and a similar error analysis can be found in [15].

2.2. Pathologically close eigenvalues and the modified Laguerre iteration. The Laguerre iteration converges only linearly to multiple zeros of a polynomial (see [28, p. 445]). Although an unreduced symmetric tridiagonal matrix can have only simple eigenvalues, some of them may be too close to be distinguishable numerically (e.g., Wilkinson matrices W_{21}^+ [28, pp. 308–309]). In this situation, the Laguerre iteration converges with an apparently linear rate in actual computing as if the cluster of r eigenvalues were a multiple eigenvalue with multiplicity r.

For integer $0 < r \leq n$, the modified Laguerre iterates L_{r+} and L_{r-} defined by

$$L_{r\pm}(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{\frac{n-r}{r}\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}} \tag{17}$$

produce two sequences

$$x_{r+}^{(k)} = L_{r+}^k(x) \equiv \overbrace{L_{r+}(L_{r+}(\cdots L_{r+}}^{k}(x)\cdots)),$$

$$x_{r-}^{(k)} = L_{r-}^k(x) \equiv \overbrace{L_{r-}(L_{r-}(\cdots L_{r-}}^{k}(x)\cdots)).$$

When x^* is a r-fold zero of f, if $x < x^*$ and there is no zeros of f in (x, x^*), then the sequence $x_{r+}^{(k)}$, $k = 1, 2, \ldots$ converges increasingly to x^* in cubic rate ([28, p. 445]). Similarly, if $x^* < x$ and there is no zeros of f in (x^*, x), then $x_{r-}^{(k)}$, $k = 1, 2, \ldots$ converges decreasingly to x^* in cubic rate. Numerical evidence shows that they also converge cubically when pathologically close zeros occur. For instance, the Wilkinson W_{21}^+ has two very close eigenvalues $\lambda_{20} = 10.74619418290332$ and $\lambda_{21} = 10.74619418290339$ with a difference of magnitude 10^{-14}. When starting from 10.0 targeting λ_{20}, the standard Laguerre iteration L_+ and the modified Laguerre iteration L_{2+} have a fundamental difference in performance, as shown below.

	L_+^k			L_{2+}^k	
k	$\lambda_{20} - x_+^{(k)} \approx$		k	$\lambda_{20} - x_{2+}^{(k)} \approx$	
1	0.7	linear	1	0.7	cubic
2	0.3	convergence	2	0.1	convergence
3	0.1		3	0.0004	
4	0.03		4	0.000000000008	
5	0.01		5	-0.00000000000004	overshoot
6	0.003				
7	0.0008				
8	0.0002				
9	0.00007				
10	0.00002				
11	0.000006				
12	0.000002				
13	0.0000005				
14	0.0000001				
15	0.00000004				
16	0.00000001				
17	0.000000003				
18	0.000000001				
19	0.0000000003				
20	0.00000000008				
21	0.00000000002				
22	0.000000000007				
23	0.000000000002				
24	0.0000000000005				
25	0.0000000000001				
26	0.00000000000002				

The behavior of the modified Laguerre iterates (17) can be seen in the following theorem.

THEOREM 2.3. *Let* $\lambda_1 < \lambda_2 < \cdots < \lambda_n$ *be zeros of* $f(\lambda) = \det(T - \lambda I)$. *Then, for* $x \in (\lambda_m, \lambda_{m+1})$ *and any positive integer* $r < n$, *we have*

$$\lambda_m < x < L_+(x) < L_{2+}(x) < \cdots < L_{r+}(x) < \lambda_{m+r} \quad \text{if} \quad -\frac{f'(x)}{f(x)} > 0, \tag{18}$$

$$\lambda_{m-r+1} < L_{r-}(x) < \cdots < L_{2-}(x) < L_-(x) < x < \lambda_{m+1} \quad \text{if} \quad -\frac{f'(x)}{f(x)} < 0 \tag{19}$$

with the convention $\lambda_i = -\infty$ *for* $i \le 0$ *and* $\lambda_i = +\infty$ *for* $i \ge n$.

(The proof is given in §7.1.)

Remark 2.4. Let $x \in (\lambda_m, \lambda_{m+1})$ be the starting point of the Laguerre iteration for evaluating λ_{m+1}. On some occasions, there may exist a group of zeros to the right of λ_{m+1}, say, $\lambda_{m+1} < \lambda_{m+2} < \cdots < \lambda_{m+r}$, which are relatively close compared to the distance from x to λ_{m+1}, as shown in Fig. 2.

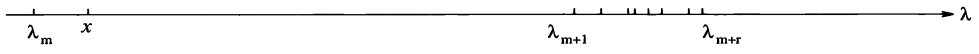


FIG. 2.

To reach λ_{m+1} from x, it may take many steps in the actual computation before reaching the cubic convergence neighborhood of λ_{m+1}, and the algorithm may be slowed down significantly.

In this case, the modified Laguerre iteration can be used to speed up the process, since, by the above theorem, $L_{r+}(x)$ is always bigger than $L_+(x)$. Specifically, if the number r of those relatively close zeros can be estimated, we may use the modified Laguerre iteration $L_{r+}(x)$. By Theorem 2.3,

$$x < L_+(x) < L_{r+}(x) < \lambda_{m+r},$$

and $L_{r+}(x)$ is relatively close to λ_{m+1}. It might happen that $\lambda_{m+1} < L_{r+}(x)$. In this case, we can reduce r according to $\kappa(L_{r+}(x))$, the number of eigenvalues of T smaller than $L_{r+}(x)$, which is a by-product of the algorithm DETEVL in Fig. 1. An algorithm ESTMLT for estimating r will be given in Fig. 8.

2.3. Stopping criteria. Let x_i, $i = 1, 2, \ldots$ be a sequence generated by the Laguerre iteration and $x_i \longrightarrow x^*$. Then, ultimately,

$$|x_{i+1} - x^*| = o(|x_{i+1} - x_i|).$$

From (16) an obvious criterion for stopping the iteration at x_{i+1} is

$$|x_{i+1} - x_i| \le 2.5\varepsilon \left[\max_j (|\beta_j| + |\beta_{j+1}|)\right] + |x_{i+1}|\varepsilon. \tag{20}$$

From our experience in numerical testing, this criterion seems too strict. In fact, the criterion

$$\frac{|x_{i+1} - x_i|^2}{|x_i - x_{i-1}|} \le 2.5\varepsilon \left[\max_j (|\beta_j| + |\beta_{j+1}|)\right] + |x_{i+1}|\varepsilon \tag{21}$$

has never failed. It comes from the following consideration: the Laguerre iteration converges cubically. Assuming $|x_{i+1} - x_i| \le |x_i - x_{i-1}|$, we have

$$|x_{i+1} - x_i| \approx |x_i - x^*| \approx c|x_{i-1} - x^*|^3 \approx c|x_i - x_{i-1}|^3,$$
$$c \approx \frac{|x_{i+1} - x_i|}{|x_i - x_{i-1}|^3}.$$

Thus

$$|x_{i+1} - x^*| \approx c|x_i - x^*|^3 \approx c|x_{i+1} - x_i|^3 \approx |x_{i+1} - x_i| \left(\frac{|x_{i+1} - x_i|}{|x_i - x_{i-1}|}\right)^3 < \frac{|x_{i+1} - x_i|^2}{|x_i - x_{i-1}|}.$$

In our algorithm, we stop the iteration if either (20) or (21) is satisfied.

The algorithm LAGIT in Fig. 3 summarizes the fundamental elements of the Laguerre iteration we described in this section.

3. Splitting and merging processes.

3.1. Splitting process. In order to find *some* or *all* the zeros of $f(\lambda) \equiv \det[T - \lambda I]$, where

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & 0 & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \beta_{n-1} & \alpha_n \end{pmatrix}, \qquad \begin{array}{l} \beta_i \neq 0, \\ i = 1, \ldots, n-1, \end{array} \tag{22}$$

Algorithm LAGIT

input: subscript i, initial point $x \in (\lambda_{i-1}, \lambda_{i+1})$,
interval $[a_i, b_i] \ni \lambda_i$, mlt evaluated by ESTMLT in Fig. 8.
output: the i-th eigenvalue λ_i of T.
begin LAGIT
$x_1 = x$
for $l = 1, 2, \ldots$

$$x_{l+1} = \begin{cases} L_{r+}(x_l), & \text{if } \kappa(x_l) < i \\ L_{r-}(x_l), & \text{if } \kappa(x_l) \geq i \end{cases} \quad \text{with } r = mlt$$

$\delta_l = x_{l+1} - x_l$
$tol = 2.5\varepsilon \left[\max_j(|\beta_j| + |\beta_{j+1}|)\right] + |x_{i+1}|\varepsilon$
if $(|\delta_l| < tol)$ or $\left(l > 1 \text{ and } \left|\frac{\delta_l^2}{\delta_{l-1}}\right| < tol\right)$ go to (#)
(##) evaluate $-\frac{f'}{f}(x_{l+1})$, $\frac{f''}{f}(x_{l+1})$ and $\kappa(x_{l+1})$ by DETEVL
update $[a_i, b_i]$ according to $\kappa(x_{l+1})$
if $mlt > 1$ and $|\kappa(x_{l+1}) - \kappa(x_l)| > 1$ then
$mlt = |\kappa(x_{l+1}) - \kappa(x_l)|$
$x_{l+1} = (x_l + x_{l+1})/2$
go to (##)
end if
end for
(#) $\lambda_i = x_{l+1}$
end LAGIT

FIG. 3. *Algorithm* LAGIT.

by the Laguerre iteration, a set of proper starting values is desirable. For this purpose, we introduce the *split matrix* $\hat{T}$ by replacing some β_k in T with zero, i.e.,

$$\hat{T} = \begin{pmatrix} T_0 & 0 \\ 0 & T_1 \end{pmatrix}, \tag{23}$$

where

$$T_0 = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{pmatrix}, \qquad T_1 = \begin{pmatrix} \alpha_{k+1} & \beta_{k+1} & & \\ \beta_{k+1} & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix}. \tag{24}$$

The eigenvalues of $\hat{T}$ consist of eigenvalues of T_0 and T_1. In this section, we shall study the relations between the eigenvalues of $\hat{T}$ and those of T. It turns out that eigenvalues of $\hat{T}$ form an excellent set of starting values of the Laguerre iteration.

Now, let us consider the homotopy $h : \boldsymbol{R} \times [0, 1] \longrightarrow \boldsymbol{R}$ defined by

$$\begin{aligned} h(\lambda, t) &= \det\left(\left[(1-t)\hat{T} + tT\right] - \lambda I\right) \\ &= \det[T(t) - \lambda I], \end{aligned}$$

where $T(t) \equiv (1-t)\hat{T} + tT$. It is clear that for each $t \in (0, 1]$, $T(t)$ is an unreduced symmetric tridiagonal matrix, and $h(\lambda_0, t_0) = 0$ if and only if λ_0 is an eigenvalue of the matrix $T(t_0)$. Let

$\lambda_i(t)$ denote the ith smallest eigenvalue of $T(t)$. Then every $\lambda_i(t)$ is a continuous function of t and for each $t \neq 0$, $\lambda_1(t) < \cdots < \lambda_n(t)$. Hence, the graphs of $\lambda_i(t)$'s are disjoint except possibly at $t = 0$. It was proved in [16] that any horizontal line $\lambda = \lambda^*$ can intersect at most one nonconstant $\lambda_i(t)$ in [0,1] and, consequently, for each $1 \leq i \leq n$, $\lambda_i(t)$ is either constant for all $t \in [0, 1]$ or strictly monotonic, as shown in Fig. 4.

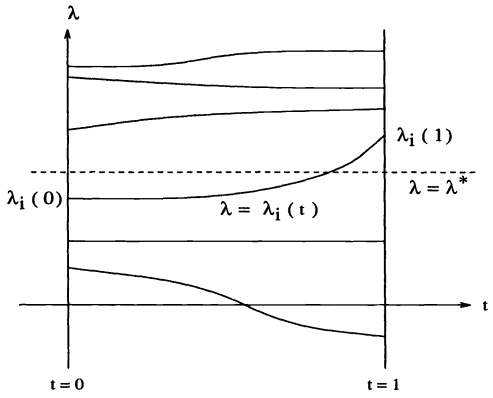


FIG. 4. *Eigenvalues of $T(t)$ when t increases from 0 to 1.*

These important properties, along with certain interlacing and perturbation theorems, lead to the following theorem, which is essential to our algorithm.

THEOREM 3.1. *Let*

$$\lambda_1 < \lambda_2 < \cdots \lambda_n \quad \textit{and} \quad \hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_n$$

be eigenvalues of T and $\hat{T}$, respectively. Then

(i) $\hat{\lambda}_1 \in (\lambda_1, \lambda_2)$, $\hat{\lambda}_n \in (\lambda_{n-1}, \lambda_n)$, $\hat{\lambda}_i \in (\lambda_{i-1}, \lambda_{i+1})$, $2 \leq i \leq n-1$;

(ii) $\lambda_1 \in [\hat{\lambda}_1 - |\beta_k|, \hat{\lambda}_1)$, $\lambda_n \in (\hat{\lambda}_n, \hat{\lambda}_n + |\beta_k|]$, $\lambda_i \in (\hat{\lambda}_{i-1}, \hat{\lambda}_{i+1})$, $2 \leq i \leq n-1$.

(The proof of this theorem is given in §7.2.)

Notice that the multiplicity of any eigenvalue $\hat{\lambda}_i$ of $\hat{T}$ can be at most two, because both T_0 and T_1 are unreduced. Therefore, for any $2 \leq i \leq n-1$, $(\hat{\lambda}_{i-1}, \hat{\lambda}_{i+1})$ is always a nonempty interval.

Corollary 3.2 follows immediately from the above theorem.

COROLLARY 3.2. *For each $i = 1, \ldots, n$, in the open interval $(\hat{\lambda}_i, \lambda_i)$ (or $(\lambda_i, \hat{\lambda}_i)$ if $\hat{\lambda}_i > \lambda_i$), there is no λ_j or $\hat{\lambda}_j$ for $j = 1, \ldots n$. In other words, for $i = 1, \cdots, n$, let $\mathcal{I}_i$ be the open interval with end points $\hat{\lambda}_i$ and λ_i, then $\{\mathcal{I}_i\}_{i=1}^n$ is a collection of disjoint intervals.*

According to this corollary, for each $i = 1, \ldots, n$, the Laguerre iteration starting from $\hat{\lambda}_i$ can reach λ_i without any obstacles (see Fig. 5). Thus, to evaluate the ith smallest eigenvalue λ_i of T, the corresponding eigenvalue $\hat{\lambda}_i$ of $\hat{T}$ is always used as a starting point in our algorithm.

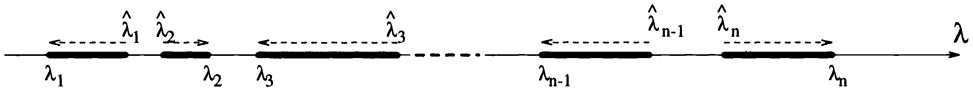


FIG. 5. *Corollary* 3.2.

3.2. Merging process. The eigenvalues of $\hat{T}$ in (22) consist of eigenvalues of T_0 and T_1 in (24). To find eigenvalues of T_0 and T_1, the *splitting process* may be applied recursively (see Fig. 6) until 2×2 or 1×1 matrices are reached.

After T is well split into a tree structure as in Fig. 6, a *merging process* in the reverse direction from 2×2 and/or 1×1 matrices can be started. More specifically, let T_σ be split into $T_{\sigma 0}$ and $T_{\sigma 1}$ in the splitting process. 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 Laguerre iteration is applied to the polynomial equation

$$f_\sigma(\lambda) \equiv \det[T_\sigma - \lambda I] = 0$$

from every $\hat{\lambda}_i^{(\sigma)}$ to obtain the corresponding eigenvalue $\lambda_i^{(\sigma)}$ of T_σ, $i = 1, \ldots, m$. This process is continued 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 Laguerre iteration to $f(\lambda) = \det(T - \lambda I)$ from eigenvalues of T_0 and T_1.

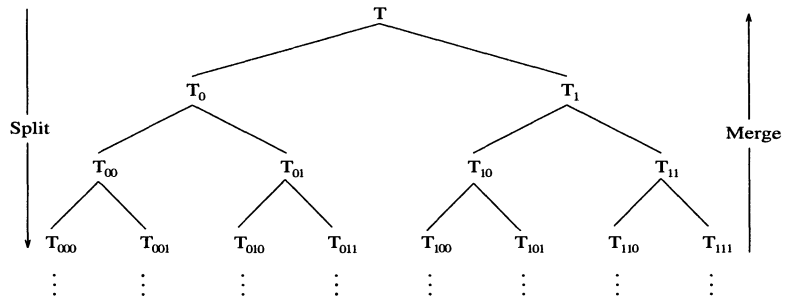


FIG. 6. *Splitting and merging.*

In the rest of this section, we discuss, without loss of generality, only the case of merging submatrices T_0 and T_1 in (24) into T in (22), or finding eigenvalues

$$\lambda_1 < \lambda_2 < \cdots < \lambda_n$$

of T from eigenvalues of

$$\hat{\lambda}_1 \leq \hat{\lambda}_2 \leq \cdots \leq \hat{\lambda}_n$$

of $\hat{T}$ by the Laguerre iteration.

From property (i) of Theorem 3.1, for each $i = 1, \ldots, n$, $\hat{\lambda}_i \in (\lambda_{i-1}, \lambda_{i+1})$ with the convention $\lambda_0 = -\infty$ and $\lambda_{n+1} = +\infty$. Suppose $\hat{\lambda}_i$ is not a zero of

$$f(\lambda) = \det(T - \lambda I),$$

namely, $\hat{\lambda}_i \neq \lambda_i$. Then, either $\hat{\lambda}_i > \lambda_i$ or $\hat{\lambda}_i < \lambda_i$; this can be determined by the Sturm sequence at $\hat{\lambda}_i$. In the following discussion, we suppose $\hat{\lambda}_i > \lambda_i$. (If $\hat{\lambda}_i < \lambda_i$, similar arguments along the same line given below follow.) It follows from Proposition 2.1, when the Laguerre iteration

$$L_\pm(x) = x + \frac{n}{\left(-\frac{f'(x)}{f(x)}\right) \pm \sqrt{(n-1)\left[(n-1)\left(-\frac{f'(x)}{f(x)}\right)^2 - n\left(\frac{f''(x)}{f(x)}\right)\right]}}$$

is applied at $\hat{\lambda}_i$, we have

$$\lambda_i < L_-(\hat{\lambda}_i) < \hat{\lambda}_i < L_+(\hat{\lambda}_i) < \lambda_{i+1}$$

and two sequences

$$x_+^{(k)} = L_+^k(\hat{\lambda}_i) \equiv L_+(L_+^{k-1}(\hat{\lambda}_i)), \qquad x_-^{(k)} = L_-^k(\hat{\lambda}_i) \equiv L_-(L_-^{k-1}(\hat{\lambda}_i)),$$

$k = 1, 2, \ldots$, with $L_+^0(\hat{\lambda}_i) = L_-^0(\hat{\lambda}_i) = \hat{\lambda}_i$ can be generated with the property

$$\lambda_i \longleftarrow \cdots < x_-^{(2)} < x_-^{(1)} < \hat{\lambda}_i < x_+^{(1)} < x_+^{(2)} < \cdots \longrightarrow \lambda_{i+1}.$$

In order to reach λ_i from $\hat{\lambda}_i$, we want to stay with the sequence $\{x_-^{(k)}\}$. Recall that this sequence converges cubically to λ_i only when $x_-^{(k)}$ reaches a certain neighborhood N of λ_i. Before reaching N, the progress of the Laguerre iteration toward convergence may not be faster than that of the method of bisection. A simple observation (see Fig. 7) shows that for x near λ_i in $(\lambda_i, \lambda_{i+1})$, $-f'(x)/f(x)$ always assumes a negative value and thus the denominator in the definition of $L_\pm(x)$ has larger magnitude if "$-$" is chosen and the necessary condition for cubic convergence is satisfied ([28, p. 445]). To illustrate the role of the sign of $-f'/f$ in the convergence behavior, we use W_{21}^+ [28, p. 308] as an example. Starting from 0.27 and targeting eigenvalue $\lambda_3 = 0.9475343675292932$, the iteration L_+ is given below.

k	$x_+^{(k)} - \lambda_3 \approx$	sign of $-f'/f$	
1	0.8	+	linear convergence
2	0.7	+	
3	0.5	+	
4	0.2	−	$-f'/f$ changes sign
5	0.01	−	and cubic convergence begins
6	0.000004	−	
7	0.00000000000002	−	

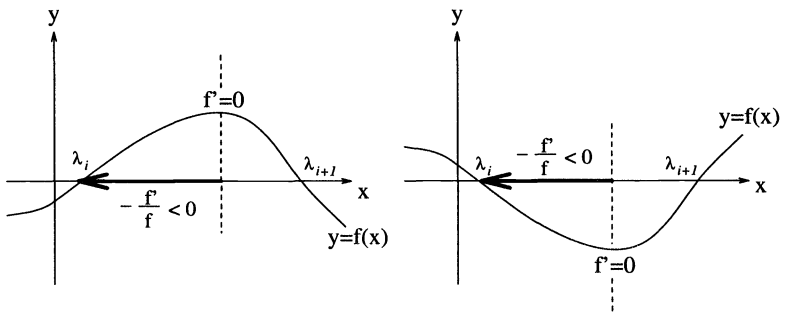


FIG. 7. *Natural direction of convergence.*

Therefore, the sign of $-f'/f$ at $x_-^{(0)} \equiv \hat{\lambda}_i$ plays a critical role in deciding whether the Laguerre iteration at $x_-^{(0)}$ should be replaced by the method of bisection to accelerate the process:

(i) If $-f'/f < 0$ at $x_-^{(0)}$, then the Laguerre iteration should be used, i.e., $x_-^{(k)} = L_-^k(x_-^{(0)})$, $k = 1, 2, \ldots$.

(ii) If $-f'/f > 0$ at $x_-^{(0)}$, then $x_-^{(0)}$ is not in N. The method of bisection should be used at $x_-^{(0)}$. Namely, $x_-^{(0)}$ should be replaced by $(\hat{\lambda}_{i-1} + x_-^{(0)})/2$.

As we mentioned in Remark 2.4, if there are certain eigenvalues of T, $\lambda_{i-r+1} < \cdots < \lambda_{i-1} < \lambda_i$, which are relatively close to λ_i compared to the starting point $x_-^{(0)}$, then it may take many steps of the standard Laguerre iteration L_- before showing cubic convergence. In this situation, the modified Laguerre iteration L_{r-} should be used at $x_-^{(0)}$. Specifically, after $x_-^{(0)}$ is altered in a bisection adjustment in (ii) above, we first estimate the number r of those eigenvalues of T from the information of the closeness of the eigenvalues $\hat{\lambda}_j$'s of $\hat{T}$: In our algorithm, we put λ_j in this group if

$$|\hat{\lambda}_j - \hat{\lambda}_{i-1}| < 0.01|x_-^{(0)} - \hat{\lambda}_{i-1}|.$$

An overestimate of the number r causes essentially no harm because LAGIT dynamically reduces r during the iteration, while an underestimate may result in slow convergence. An algorithm ESTMLT to estimate r is designed in Fig. 8.

Algorithm ESTMLT

```
    input:  initial point x, sign(λ_i − x), subscript i,
            eigenvalues λ̂_1 < ··· < λ̂_n of T̂
    output: mlt, the estimated numerical multiplicity of
            the eigenvalue λ_i of T
    begin ESTMLT
        mlt = 1
        for k = 1, 2, ...
            j = i + k · sign(λ_i − x)
            if |λ̂_{i−1} − λ̂_j| < 0.01|x − λ̂_{i−1}| then
                mlt = mlt + 1
            else
                go to (#)
            end if
        end for
(#) end ESTMLT
```

FIG. 8. *Algorithm* ESTMLT.

4. Cluster and partial spectrum.

4.1. Cluster evaluation. By a straightforward verification, from Theorem 3.1 one can obtain the following proposition.

PROPOSITION 4.1. *If $[a, b]$ contains k eigenvalues of $\hat{T}$, then $[a, b]$ contains at least $k-2$ and at most $k+2$ eigenvalues of T. More precisely, let $\kappa(x)$ be the number of eigenvalues of T that are less than $x \in \mathbf{R}$ and let $\hat{\lambda}_{s+1}, \ldots, \hat{\lambda}_{s+k}$ be all the eigenvalues of $\hat{T}$ in $[a, b]$. Then $s - 1 \le \kappa(a) \le s + 1$ and $s + k - 1 \le \kappa(b) \le s + k + 1$.*

This proposition can be illustrated by following eigenvalue curves $\lambda_i(t)$ of $T(t) \equiv (1 - t)\hat{T} + tT$ as two extreme cases that are shown in Fig. 9.

By this proposition, if $\hat{T}$ has a cluster of $m(\ge 3)$ eigenvalues, say $\hat{\lambda}_{i+1}, \ldots, \hat{\lambda}_{i+m}$, then $\hat{\lambda}_{i+2}, \ldots, \hat{\lambda}_{i+m-1}$ can be accepted as eigenvalues of T without further computations. In our implementation, we set $\lambda_i = \hat{\lambda}_i$ when

$$|\hat{\lambda}_{i+1} - \hat{\lambda}_{i-1}| < error\_tolerance.$$

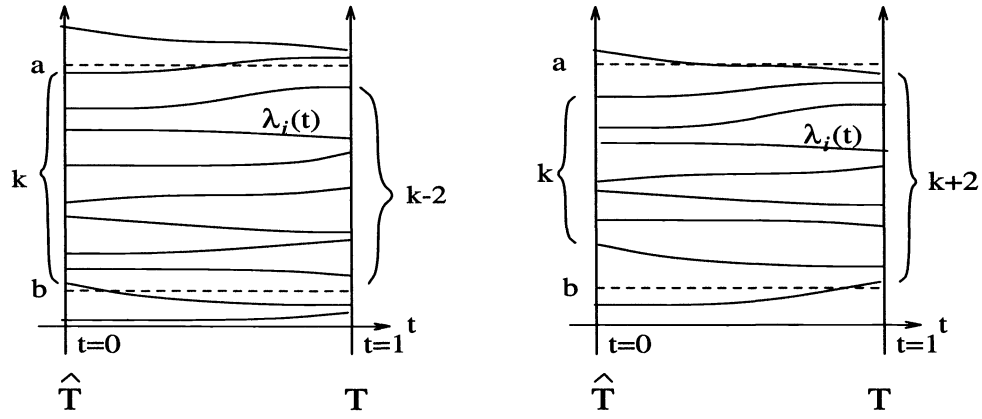


FIG. 9.

4.2. Evaluating partial spectrum. In our algorithm, the ith smallest eigenvalue $\hat{\lambda}_i$ of $\hat{T}$ is always used as the starting point of the Laguerre iteration to obtain the ith smallest eigenvalue λ_i of T. Therefore, when eigenvalues of T in demand are identified in magnitude, say largest 20%, they can be evaluated by using the largest 20% eigenvalues of $\hat{T}$ as starting points without computing other eigenvalues of either $\hat{T}$ or T.

To find eigenvalues of T in a given interval $[a, b]$, the eigenvalues of $\hat{T}$ in $[a, b]$ are known, say $\hat{\lambda}_{s+1}, \ldots, \hat{\lambda}_{s+k}$. By evaluating $\kappa(a)$ and $\kappa(b)$, the actual member of eigenvalues of T in $[a, b]$ is $\sigma = \kappa(b) - \kappa(a)$, so that $\lambda_{\kappa(a)+1}, \ldots, \lambda_{\kappa(a)+\sigma}$ are the eigenvalues of T in $[a, b]$. By Proposition 4.1, $s - 1 \le \kappa(a) \le s + 1$ and $s + k - 1 \le \kappa(b) \le s + k - 1$, so σ can either be $k - 2, k - 1, k, k + 1$, or $k + 2$. Thus, at most $k + 2$ starting points are necessary to evaluate these σ eigenvalues of T. Let

$$\hat{\hat{\lambda}}_s = a, \quad \hat{\hat{\lambda}}_{s+1} = \hat{\hat{\lambda}}_{s+1}, \ldots, \quad \hat{\hat{\lambda}}_{s+k} = \hat{\hat{\lambda}}_{s+k}, \quad \hat{\hat{\lambda}}_{s+k+1} = b.$$

Among them, $\hat{\hat{\lambda}}_{\kappa(a)+1}, \ldots, \hat{\hat{\lambda}}_{\kappa(a)+\sigma}$ will lead to all σ eigenvalues of T in $[a, b]$.

The algorithm MERGE in Fig. 10 puts together all the elements discussed in §§3 and 4.

5. Numerical testing. Our algorithm is implemented and tested on SPARC station 1 with IEEE floating point standard. The machine precision is $\varepsilon \approx 2.2 \times 10^{-16}$.

5.1. Testing matrices. In the following description of matrix types, α_i, $i = 1, \ldots, n$ denote the diagonal entries and β_i, $i = 1, \ldots, n - 1$ are the offdiagonal entries. The testing matrices are classified into 12 types.

Matrices with known eigenvalues.

Type 1. Toeplitz matrices $[b, a, b]$. Exact eigenvalues: $\{a + 2b\cos\frac{k\pi}{n+1}\}_{k=1}^{n}$ ([10, 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: $\{a + 2b\cos\frac{(2k-1)\pi}{2n}\}_{1\le k\le n}$ ([10, Ex. 7.6, p. 138]).

Type 3. $\alpha_i = \begin{cases} a & \text{for odd } i, \\ b & \text{for even } i, \end{cases} \quad \beta_i = 1.$

Exact eigenvalues:

$$\left\{\frac{a + b \pm \left[(a-b)^2 + 16\cos^2\frac{k\pi}{n}\right]}{2}\right\}_{1\le k\le n/2} \quad \text{and } a \text{ if } n \text{ is odd}$$

Algorithm MERGE

input: $T = [\beta_{i-1}, \alpha_i, \beta_i]$, interval $[a, b]$
eigenvalues $\hat{\lambda}_{s+1} < \cdots < \hat{\lambda}_{s+k}$ of $\hat{T}$ in $[a, b]$.
output: eigenvalues of T in $[a, b]$.
begin MERGE
set $\hat{\lambda}_s = a$, $\hat{\lambda}_{s+k+1} = b$
$i_1 = \kappa(a) + 1$, $i_2 = \kappa(b)$
for $i = i_1 : i_2$
determine the interval $(a_i, b_i) \ni \lambda_i$ by Theorem 3.1
if $|b_i - a_i| > tol$ then
$x = \hat{\lambda}_i$
(#) call DETEVL at x to obtain $-\frac{f'(x)}{f(x)}$, $\frac{f''(x)}{f(x)}$ and $\kappa(x)$

$$(a_i, b_i) = \begin{cases} (x, b_i) & \text{if } x < \lambda_i \text{ (i.e. } \kappa(x) < i) \\ (a_i, x) & \text{if } x > \lambda_i \text{ (i.e. } \kappa(x) \geq i) \end{cases}$$

if $x \notin (\lambda_{i-1}, \lambda_{i+1})$ or $sign\left(-\frac{f'(x)}{f(x)}\right) \neq sign(\lambda_i - x)$ then
$x = (a_i + b_i)/2$
go to (#)
end if
call ESTMLT to estimate the multiplicity of λ_i
call LATIT to obtain λ_i
else
$\lambda_i = (a_i + b_i)/2$
end if
end for
end MERGE

FIG. 10. *Algorithm* MERGE.

([10, Ex. 7.8 and 7.9, p. 139]).

Type 4. $\alpha_i = 0$, $\beta_i = \sqrt{i(n-i)}$. Exact eigenvalues: $\{-n + 2k + 1\}_{1 \leq k \leq n}$ ([10, Ex. 7.10, p. 140]).

Type 5. $\alpha_i = -[(2i-1)(n-1) - 2(i-1)^2]$, $\beta_i = i(n-i-2)$. Exact eigenvalues: $\{-k(k-1)\}_{1 \leq k \leq n}$ ([10, Ex. 7.11, p. 141]).

Wilkinson and random matrices.

Type 6. Wilkinson matrices W_n^+. $\beta_i = 1$,

$$\alpha_i = \begin{cases} \frac{n}{2} - i + 1 & \text{for even } n \text{ and } 1 \leq i \leq \frac{n}{2}, \\ i - \frac{n}{2} & \text{for even } n \text{ and } \frac{n}{2} < i \leq n, \\ \frac{n-1}{2} - i + 1 & \text{for odd } n \text{ and } 1 \leq i \leq \frac{n+1}{2}, \\ i - \frac{n+1}{2} & \text{for odd } n \text{ and } \frac{n+1}{2} < i \leq n \end{cases}$$

([28, pp. 308–309]). Most of the eigenvalues are in pairs consisting of two numerically indistinguishable eigenvalues.

Type 7. Random matrices. α_i's and β_i's are random numbers on [0, 1].

LAPACK testing matrices. (Matrix Types 8–12 were generated using the LAPACK test matrix generator [1].)

Type 8. Matrices with eigenvalues evenly distributed between its 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 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 $[10^{-12} - \varepsilon, 10^{-12} + \varepsilon]$.

5.2. Testing codes. We compare the performance of the following five algorithms:

(1) S-M: our split-merge algorithm;

(2) B/M: bisection/multisection subroutine DSTEBZ in LAPACK;

(3) D&C: divide-and-conquer subroutine TREEQL developed by J. Dongarra and D. Sorensen, dated May 17, 1985 [7];

(4) RFQR: root-free QR routine DSTERF in LAPACK, as recommended in LAPACK for evaluating eigenvalues only;

(5) QR: QR routine DSTEQR in LAPACK, as recommended in LAPACK for evaluating eigenvalues or eigenpairs.

5.3. Storage comparison. The storage requirements of these five codes for evaluating eigenvalues of an $n \times n$ matrix are:

RFQR: $2n$.
QR: $4n$. ($n^2 + O(n)$ if eigenvectors are also evaluated.)
S-M: $7n$.
B/M: $12n$.
D&C: $2n^2 + O(n)$.

The storage requirement of our algorithm, as well as those of QR and B/M, is low. Because eigenvectors are stored, the code D&C requires n^2 storage, which makes it somewhat difficult to manipulate matrices with order higher than 500 on most of the workstations. The low storage of our algorithm also eases the memory contention in the parallel implementation.

5.4. Accuracy test. The accuracy is tested in two ways. First, we test all those algorithms on matrices chosen from Type 1 to Type 5 using their default error tolerances. These matrices have known eigenvalues. Thus the error can be directly evaluated. Let $\tilde{\lambda}_i$ be the approximation of the exact eigenvalue λ_i. Table 1 gives the error

$$\max_i \left\{ |\tilde{\lambda}_i - \lambda_i| / \|T\| \right\}.$$

It appears that our algorithm achieves the best accuracy on these matrices. The accuracy of our algorithm, as well as B/M, is independent of the matrix size, whereas the QR, RFQR, and D&C seem less accurate when the matrix size becomes larger. On matrices of Type 4, with $n = 1999$, the error of RFQR is more than 1916 times larger than ours.

On the remaining types of matrices whose exact eigenvalues are unknown, the following test is made. From §2.1, the accuracy of DETEVL is

$$e(\lambda) = 2.5\varepsilon \left[\max_j (|\beta_j| + |\beta_{j+1}|) \right] + |\lambda|\varepsilon.$$

If an evaluated eigenvalue λ_i has this accuracy, then $\kappa(\lambda_i - 2e(\lambda_i)) < i$ and $\kappa(\lambda_i + 2e(\lambda_i)) \ge i$. We examine the percentage of failures of this test on all the methods. The results are in Table 2. As a reference, we also list the percentage of failures on matrices of Type 1 to Type 5. Notice that this test is apparently in favor of B/M and S-M because both algorithms

TABLE 1

Accuracy on matrices with known eigenvalues. Because of the memory requirement, the D&C *algorithm cannot be executed on our machine for* $n = 999$ *or* 1999.

		$n = 99$	$n = 199$	$n = 499$	$n = 999$	$n = 1999$
Type 1	S-M	0.67 ε	0.67 ε	0.67 ε	0.67 ε	0.67 ε
$a = 4$	B/M	1.00 ε	1.00 ε	1.00 ε	1.00 ε	1.00 ε
$b = 1$	D&C	1.00 ε	2.67 ε	4.67 ε	—	—
	QR	2.00 ε	4.00 ε	4.00 ε	4.67 ε	9.00 ε
Type 2	S-M	0.67 ε	0.67 ε	0.67 ε	0.67 ε	0.67 ε
$a = 4$	B/M	0.67 ε	0.67 ε	0.67 ε	0.67 ε	1.00 ε
$b = 1$	D&C	1.33 ε	2.67 ε	4.67 ε	—	—
	QR	2.67 ε	2.67 ε	4.00 ε	4.67 ε	6.67 ε
Type 3	S-M	0.80 ε	0.80 ε	0.80 ε	0.80 ε	0.80 ε
$a = 4$	B/M	0.80 ε	0.80 ε	0.80 ε	0.80 ε	0.80 ε
$b = 1$	D&C	3.68 ε	1.60 ε	2.40 ε	—	—
	QR	4.00 ε	4.00 ε	8.00 ε	8.00 ε	13.6ε
Type 4	S-M	0.16 ε	0.04 ε	0.13 ε	0.036 ε	0.032ε
	B/M	0.65 ε	1.29 ε	1.03 ε	1.23 ε	1.23 ε
	D&C	1.95 ε	7.76 ε	7.71 ε	—	—
	QR	7.18 ε	16.2 ε	15.4 ε	22.6 ε	24.1 ε
Type 5	S-M	0.53 ε	0.65 ε	0.65 ε	0.65 ε	0.65 ε
	B/M	0.85 ε	0.85 ε	1.06 ε	1.05 ε	1.05 ε
	D&C	1.27 ε	1.25 ε	1.85 ε	—	—
	QR	1.69 ε	4.99 ε	6.86 ε	3.26 ε	7.35 ε

involve the evaluation of $\kappa(\cdot)$, while the QR algorithm, which is considered quite accurate in general, may have a failure rate close to 70%.

5.5. Speed test. To compare the efficiency of these four algorithms, we perform the following experiments:

• *Evaluating all eigenvalues without computing eigenvectors.* The results are in Table 3. Our S-M algorithm shares this feature with B/M, QR, and RFQR, while the D&C algorithm cannot separate the evaluation of eigenvalues and eigenvectors.

• *Evaluating all eigenvalues* and *eigenvectors.* Results are in Table 4. For this test, we use the standard QR routine DSTEQR instead of root-free QR DSTERF. Both S-M and B/M are basically designed to evaluate eigenvalues only. So, after running S-M and B/M, the inverse iteration code DSTEIN in LAPACK is used to evaluate the corresponding eigenvectors. In Table 4, the time for S-M is actually the time consumption of (S-M)+DSTEIN, and the time for B/M is the time consumption of (B/M)+DSTEIN.

• *Evaluating* $\frac{1}{3}$ *of the largest eigenvalues without computing eigenvectors.* Both S-M and B/M have a great advantage in this case. The time consumption is reduced to about $\frac{1}{3}$ of the time for evaluating the entire spectrum. On the other hand, both QR and D&C must evaluate all the eigenvalues to obtain the largest $\frac{1}{3}$. The numerical results are shown in Table 3.

• *Evaluation of* $\frac{1}{3}$ *of the largest eigenvalues* and *their corresponding eigenvectors.* Again, only S-M and B/M have this feature. The results are in Table 4. Both QR and D&C must evaluate all the eigenpairs even if $\frac{2}{3}$ of them are unnecessary.

From those results, RFQR is apparently the fastest algorithm in computing all the eigenvalues. In this case, the speed of our algorithm S-M is competitive with RFQR with better accuracy. In evaluating all the eigenvalues and their corresponding eigenvectors, our algorithm leads DSTEQR by a considerable margin.

TABLE 2

Accuracy on matrices with unknown eigenvalues (percentages are the rates of failure in the accuracy test). Because of the memory requirement, LAPACK *matrix generator can not produce testing matrices for higher order.*

		$n = 99$	$n = 199$	$n = 499$			$n = 99$	$n = 199$	$n = 499$
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 1	B/M	0%	0%	0%	Type 7	B/M	0%	0%	0%
	D&C	0%	10.6%	56.9%		D&C	error	error	error
	RFQR	5.1%	4.52%	6.81%		RFQR	11.1%	11.6%	26.9%
	QR	0%	7.53%	7.01%		QR	5.05%	12.1%	24.6%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 2	B/M	0%	0%	0%	Type 8	B/M	0%	0%	0%
	D&C	0.%	9.04%	58.5%		D&C	0%	0%	4.01%
	RFQR	4.04%	4.02%	8.42%		RFQR	2.02%	1.50%	15.2%
	QR	3.03%	6.53%	8.42%		QR	0%	0.50%	16.8%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 3	B/M	0%	0%	0%	Type 9	B/M	0%	0%	0%
	D&C	21.2%	0%	0%		D&C	error	error	error
	RFQR	3.03%	1.51%	3.61%		RFQR	1.01%	0%	0%
	QR	9.09%	7.54%	10.0%		QR	0%	0%	0%
	S-M	0%	0%	0%		S-M	0%	0%	0.20%
Type 4	B/M	0%	0%	0%	Type 10	B/M	0%	0%	0.40%
	D&C	0%	4.52%	0.40%		D&C	12.1%	2.51%	2.40%
	RFQR	6.06%	11.1%	14.6%		RFQR	0%	0%	0%
	QR	3.03%	8.54%	9.42%		QR	0%	0%	0.40%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 5	B/M	0%	0%	0%	Type 11	B/M	0%	0%	0%
	D&C	0%	0%	0%		D&C	1.01%	1.01%	19.0%
	RFQR	1.01%	2.01%	2.00%		RFQR	0.81%	10.6%	18.2%
	QR	0%	1.00%	3.21%		QR	0.84%	6.53%	21.0%
	S-M	0%	0%	0%		S-M	0%	0%	0%
Type 6	B/M	4.04%	8.04%	8.22%	Type 12	B/M	10.1%	0%	3.21%
	D&C	8.08%	14.1%	36.9%		D&C	5.05%	1.51%	0.80%
	RFQR	51.5%	60.8%	69.3%		RFQR	0%	0%	0%
	QR	51.5%	56.3%	56.1%		QR	3.03%	0%	0.20%

The results also show that our algorithm can be applied efficiently to evaluate partial spectrum and partial eigenpairs. The speed is several times faster than B/M in most of the cases. In comparison with D&C, our algorithm leads substantially except in the cases where a large number of clusters exist (Types 6, 10, and 12). However, the timing of D&C in those cases is somewhat misleading, because we obtained error messages. Sorensen–Tang [25] and Gu–Eisenstat [11] reported different approaches to achieve orthogonality of eigenvectors for D&C algorithm. The version of D&C code TREEQL we used does not have this feature.

6. Discussions.

6.1. Rank-two versus rank-one tearing. For the matrix

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{pmatrix}, \qquad \beta_i \neq 0, \quad i = 1, \ldots, n-1. \tag{25}$$

TABLE 3

Execution time (in seconds) for evaluating all eigenvalues without computing eigenvectors: figures outside parentheses are the number of seconds for evaluating all eigenvalues. The figures inside parentheses represent the times for the corresponding codes to evaluate $\frac{1}{3}$ of the largest eigenvalues. Neither D&C *nor* QR *can take advantage of this case and thus has no such time listed. The* D&C *code* TREEQL *must evaluate all eigenvectors in order to evaluate the eigenvalues.*

		$n = 99$	$n = 199$	$n = 499$
	S-M	1.27 (0.52)	4.13 (1.55)	23.4 (8.18)
Type 1	B/M	3.40 (1.26)	12.4 (4.24)	73.5 (24.5)
$a = 4$	RFQR	0.76	2.21	12.5
$b = 1$	D&C	8.56	59.1	910.
	S-M	1.51 (0.66)	5.33 (1.94)	31.2 (10.5)
Type 2	B/M	3.30 (1.31)	12.3 (4.27)	73.5 (24.5)
$a = 4$	RFQR	0.69	2.28	12.6
$b = 1$	D&C	8.15	55.7	851.
	S-M	1.26 (0.61)	4.25 (1.61)	23.8 (8.49)
Type 3	B/M	3.26 (1.26)	12.2 (4.26)	72.4 (24.4)
$a = 4$	RFQR	0.69	2.05	11.5
$b = 1$	D&C	9.05	62.3	905.
	S-M	1.29 (0.56)	4.35 (1.64)	24.9 (8.55)
Type 4	B/M	3.43 (1.32)	12.8 (4.56)	77.4 (26.1)
	RFQR	0.71	2.30	13.0
	D&C	8.99	62.3	906.
	S-M	1.32 (0.56)	4.39 (1.67)	25.0 (8.85)
Type 5	B/M	3.38 (1.28)	12.5 (4.23)	74.8 (23.4)
	RFQR	0.70	2.31	12.7
	D&C	8.92	62.2	893.
	S-M	0.82 (0.34)	2.13 (0.79)	9.11 (3.14)
Type 6	B/M	2.21 (0.72)	7.13 (2.45)	40.7 (13.4)
	RFQR	0.68	2.05	10.0
	D&C	2.39	6.89	46.9
	S-M	1.07 (0.50)	3.71 (1.25)	15.9 (5.17)
Type 7	B/M	3.39 (1.26)	12.4 (4.39)	74.5 (25.4)
	RFQR	0.86	2.58	15.0
	D&C	7.27	23.7	141.
	S-M	1.47 (0.59)	5.12 (1.81)	29.1 (9.96)
Type 8	B/M	3.43 (1.32)	12.8 (4.43)	76.8 (26.0)
	RFQR	0.71	12.4	12.0
	D&C	8.99	62.5	848.
	S-M	1.02 (0.46)	2.95 (1.21)	16.1 (5.82)
Type 9	B/M	1.98 (0.98)	7.00 (3.39)	40.6 (20.2)
	RFQR	0.45	1.30	6.61
	D&C	1.48	6.96	73.6
	S-M	0.51 (0.25)	0.57 (0.34)	2.53 (1.47)
Type 10	B/M	0.25 (0.27)	0.37 (0.39)	0.60 (0.79)
	RFQR	0.59	1.83	9.94
	D&C	0.94	4.06	41.9
	S-M	1.43 (0.63)	5.00 (1.81)	28.4 (9.39)
Type 11	B/M	3.38 (1.31)	12.7 (4.44)	75.4 (25.4)
	RFQR	0.72	2.17	12.2
	D&C	9.01	62.2	845.
	S-M	0.30 (0.28)	0.35 (0.33)	0.65 (0.69)
Type 12	B/M	0.26 (0.29)	0.32 (0.46)	0.60 (0.77)
	RFQR	0.53	1.47	7.59
	D&C	0.85	3.90	39.1

TABLE 4

Execution time (in seconds) for evaluating both eigenvalues and eigenvectors: the figures outside parentheses are the number of seconds for evaluating all eigenpairs. The figures in parentheses represent the times for the corresponding codes to evaluate eigenpairs of the largest $\frac{1}{3}$ eigenvalues. Neither D&C *nor* QR *can take advantage of this case and thus has no such time listed. *: error messages output when executing the* D&C *code* TREEQL

		$n = 99$	$n = 199$	$n = 499$
	S-M	3.45 (1.32)	13.6 (4.77)	116. (45.0)
Type 1	B/M	5.55 (2.02)	21.6 (7.38)	164. (60.9)
$a = 4$	D&C	8.56	59.1	910.
$b = 1$	QR	15.2	112.	1744
	S-M	3.73 (1.40)	14.8 (5.10)	124. (47.2)
Type 2	B/M	5.48 (2.00)	21.6 (7.45)	165. (60.8)
$a = 4$	D&C	8.15	55.7	851.
$b = 1$	QR	15.2	115.	1763
	S-M	3.50 (1.36)	14.0 (4.96)	202. (68.8)
Type 3	B/M	5.65 (2.05)	21.8 (7.48)	249. (84.1)
$a = 4$	D&C	9.05	62.3	905.
$b = 1$	QR	14.0	104.	1582
	S-M	3.55 (1.37)	13.4 (4.69)	81.7 (27.5)
Type 4	B/M	11.4 (2.75)	67.6 (12.6)	846. (124.)
	D&C	8.99	62.3	906.
	QR	14.9	115.	1815
	S-M	3.72 (1.39)	14.1 (5.34)	127. (72.7)
Type 5	B/M	11.7 (12.6)	67.7 (12.6)	843. (123.)
	D&C	8.92	62.2	892.
	QR	15.0	115.	1781
	S-M	3.05 (1.13)	11.3 (3.82)	65.6 (21.9)
Type 6	B/M	4.34 (1.50)	16.1 (5.45)	95.6 (31.9)
	D&C	2.39*	6.89*	46.9*
	QR	14.1	105.	1478
	S-M	3.38 (1.29)	13.1 (4.40)	75.1 (24.7)
Type 7	B/M	5.64 (2.10)	21.3 (7.36)	131. (44.0)
	D&C	error	error	error
	QR	17.4	135.	2108
	S-M	3.78 (1.40)	14.5 (4.97)	87.3 (28.9)
Type 8	B/M	5.61 (2.07)	21.8 (7.41)	132. (44.1)
	D&C	8.99	62.5	849.
	QR	15.3	121.	1818
	S-M	7.35 (1.75)	45.2 (7.30)	610. (76.4)
Type 9	B/M	7.95 (2.19)	45.2 (9.36)	601. (83.7)
	D&C	error	error	error
	QR	12.0	93.4	1265
	S-M	8.31 (1.79)	55.6 (8.30)	784. (99.4)
Type 10	B/M	8.16 (1.67)	55.1 (8.28)	771. (98.1)
	D&C	0.94*	4.06*	41.9*
	QR	13.1	100.	1906
	S-M	3.74 (1.41)	14.4 (5.01)	86.8 (28.9)
Type 11	B/M	5.61 (2.07)	21.4 (7.33)	130. (43.6)
	D&C	9.01	62.2	845.
	QR	15.2	121.	1906
	S-M	8.13 (1.69)	55.0 (8.30)	776. (99.1)
Type 12	B/M	8.56 (1.70)	56.2 (8.61)	769. (99.5)
	D&C	0.85*	3.90*	39.1*
	QR	11.3	80.7	1186

Our split-merge process is similar to Cuppen's divide-and-conquer method [4], where T is split by a rank-one tearing:

$$(26)\quad T^{(1)} \equiv \begin{pmatrix} T_0^{(1)} & 0 \\ 0 & T_1^{(1)} \end{pmatrix} = T - P^{(1)} \quad \text{for } P^{(1)} = \begin{pmatrix} & \vdots & \\ \cdots & \theta \quad \beta_k & \cdots \\ & \beta_k \quad \frac{\beta_k^2}{\theta} & \\ & \vdots & \end{pmatrix}.$$

In our algorithm, we split T by a rank-two tearing:

$$(27)\quad T^{(2)} \equiv \begin{pmatrix} T_0^{(2)} & 0 \\ 0 & T_1^{(2)} \end{pmatrix} = T - P^{(2)} \quad \text{for } P^{(2)} = \begin{pmatrix} & \vdots & \\ \cdots & 0 \quad \beta_k & \cdots \\ & \beta_k \quad 0 & \\ & \vdots & \end{pmatrix}.$$

For $\theta \neq 0$, the matrix $P^{(1)}$ is of rank one, and the rank of $P^{(2)}$ is of rank two. (Throughout this section, the superscripts $^{(1)}$ and $^{(2)}$ are used to indicate the rank-one and rank-two tearings, respectively.) There are some interesting properties of Cuppen's rank-one tearing such as a separation result stronger than Theorem 3.1 ([9, Thm. 8.6.2, p. 462]). However, if we consider the distance from the eigenvalues of perturbed matrices $T^{(1)}$ or $T^{(2)}$ to the corresponding eigenvalues of T, our rank-two tearing is better than the rank-one tearing, because we can always apply the Laguerre iteration from closer starting points.

LEMMA 6.1 [3, p. 34]. *Let A and B be real symmetric matrices with eigenvalues $\lambda_1(A) \leq \cdots \leq \lambda_n(A)$ and $\lambda_1(B) \leq \cdots \leq \lambda_n(B)$, respectively. Then*

$$\max_j |\lambda_j(A) - \lambda_j(B)| \leq \|A - B\|.$$

LEMMA 6.2. *For real symmetric matrices A and B, let $\{\lambda_i(A)\}_{i=1}^n$, $\{\lambda_i(B)\}_{i=1}^n$ and $\{\lambda_i(A-B)\}_{i=1}^n$ be eigenvalues of A, B, and $A - B$ (in ascending order), respectively. Then*

$$\sum_{i=1}^n |\lambda_i(A) - \lambda_i(B)| \leq \sum_{i=1}^n |\lambda_i(A - B)|.$$

Proof. Use [3, Thm. 9.7, p. 45] with the Ky Fan n-norm ([3, p. 28]). □

The following theorem compares the rank-one and rank-two perturbations as in (26) and (27). It shows that the upper bound for rank-two perturbation on each eigenvalue is no more than one half of the upper bound for rank-one perturbation; and the total perturbation of rank-two tearing is bounded above by the *lower* bound of total perturbation of rank-one tearing.

THEOREM 6.3. *Let T, $T^{(1)}$, and $T^{(2)}$ be as in (25)–(27) with spectrums $\{\lambda_i\}_{i=1}^n$, $\{\lambda_i^{(1)}\}_{i=1}^n$, and $\{\lambda_i^{(2)}\}_{i=1}^n$, respectively, in ascending order. Then*

$$(28)\quad |\lambda_i - \lambda_i^{(1)}| \leq \frac{\theta^2 + \beta_k^2}{|\theta|}, \qquad i = 1, \ldots, n,$$

$$(29)\quad |\lambda_i - \lambda_i^{(2)}| \leq |\beta_k| \leq \frac{1}{2}\frac{\theta^2 + \beta_k^2}{|\theta|}, \qquad i = 1, \ldots, n,$$

$$(30)\quad \sum_{i=1}^n |\lambda_i - \lambda_i^{(2)}| \leq 2\left|\beta_k\right| \leq \sum_{i=1}^n \left|\lambda_i - \lambda_i^{(1)}\right|.$$

Proof. The spectrums of $P^{(1)}$ and $P^{(2)}$ are $\{0, (\theta^2+\beta_k^2)/\theta\}$ and $\{0, \pm\beta_k\}$, respectively. By Lemma 6.1, for each $1 \le i \le n$,

$$|\lambda_i - \lambda_i^{(1)}| \le \|P^{(1)}\| = \frac{\theta^2+\beta_k^2}{|\theta|}, \qquad |\lambda_i - \lambda_i^{(2)}| \le \|P^{(2)}\| = |\beta_k|.$$

It is easy to see that

$$\min_{\theta>0} \frac{\theta^2+\beta_k^2}{\theta} = 2|\beta_k|.$$

Thus,

$$|\lambda_i - \lambda_i^{(2)}| \le |\beta_k| \le \frac{1}{2}\frac{\theta^2+\beta_k^2}{|\theta|}.$$

It follows from Lemma 6.2 that

$$\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(2)}| \le 2|\beta_k|.$$

On the other hand,

$$\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(1)}| \ge \left|\sum_{i=1}^{n} (\lambda_i - \lambda_i^{(1)})\right|$$

$$= \left|\sum_{i=1}^{n} \lambda_i - \sum_{i=1}^{n} \lambda_i^{(1)}\right| = |\mathrm{tr}(T) - \mathrm{tr}(T^{(1)})| = |\mathrm{tr}(P^{(1)})| = |\theta| + \frac{\beta_k^2}{|\theta|} \ge 2|\beta_k|,$$

where $\mathrm{tr}(A)$ denotes the trace of a matrix A. □

From this theorem one can easily see that to reach the eigenvalue λ_i of T by using $\lambda_i^{(1)}$ of $T^{(1)}$ as a starting point, more work is expected than starting from $\lambda_i^{(2)}$ of $T^{(2)}$ instead. Indeed, numerical testing shows that for rank-one perturbation, $\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(1)}|$ can be four times as big as $\sum_{i=1}^{n} |\lambda_i - \lambda_i^{(2)}|$ in rank-two perturbation. Table 5 lists the comparison of these two perturbations on 12 types of matrices given in §5 with $n = 99$.

TABLE 5
Numerical comparison between rank-one and rank-two perturbations.

Matrix types	Rank two $\sum_i \lvert\lambda_i - \lambda_i^{(2)}\rvert$	Rank one $\sum_i \lvert\lambda_i - \lambda_i^{(1)}\rvert$	Matrix types	Rank two $\sum_i \lvert\lambda_i - \lambda_i^{(2)}\rvert$	Rank one $\sum_i \lvert\lambda_i - \lambda_i^{(1)}\rvert$
Type 1	0.726965858	2.000000000	Type 7	0.151873922	0.455857515
Type 2	1.472573929	2.000000000	Type 8	0.451587126	0.805204125
Type 3	0.468625974	2.000000000	Type 9	2.3505×10^{-8}	3.9610×10^{-8}
Type 4	35.97841949	98.99494937	Type 10	$\approx 10^{-17}$	$\approx 10^{-17}$
Type 5	1780.795116	4900.000000	Type 11	0.229865425	0.397815459
Type 6	0.758494678	2.000000000	Type 12	$\approx 10^{-17}$	$\approx 10^{-17}$

In the case of the singular value evaluation, which is equivalent to the symmetric tridiagonal eigenvalue problem on matrices with zero diagonal entries, rank-two tearing has another advantage of keeping the structure of zero diagonal. Thus the eigenvalues of split matrices can be evaluated within a tiny relative error [18].

6.2. On parallelization and vectorization. Our algorithm can be parallelized in a similar way as in D&C [7]. There are two levels of parallelism. First, when the matrix T is split in a tree as in Fig. 6, then solving the eigenproblem of each submatrix is independent of the others and thus can be done by one processor or a group of processors. Second, when solving the eigenproblem of a submatrix of order m, each one of the m eigenvalues can be evaluated separately by performing the independent Laguerre iteration. Thus if a group of processors are assigned to this submatrix, these independent Laguerre iterations can be distributed to those processors. The combination of both levels makes our method fully parallel and an excellent candidate for advanced architectures.

Our method is also suitable for vector computing. For instance, when m eigenvalues of a submatrix are being computed, the algorithm evaluates the determinants $\det(T - xI)$ and their derivatives at several starting points $x_1, \ldots, x_m$. This process can be performed with some do loops described in Fig. 11, where the i-loops are vector operations.

```
for i = 1 : m
    initialize ξ_{i1}, η_{i0}, η_{i1}, ζ_{i0}, and ζ_{i1}
end for
for j = 2 : m
    for i = 1 : m
        ξ_{ij} = α_j − x_i − β²_{j−1} / ξ_{i,j−1}
        η_{ij} = (1/ξ_{ij}) [ (α_j − x_i) η_{i,j−1} + 1 − (β²_{j−1} / ξ_{i,j−1}) η_{i,j−2} ]
        ζ_{ij} = (1/ξ_{ij}) [ (α_j − x_i) ζ_{i,j−1} + η_{i,j−1} − (β²_{i,j−1} / ξ_{i,j−1}) ζ_{i,j−2} ]
    end for
end for
```

FIG. 11. *Vectorizable loops.*

The implementation strategies and experimental results of the parallelization and vectorization of our method will be reported in a separate paper [26].

6.3. Other possible variations. Our algorithm presented in this paper mainly consists of a split-merge process and a carefully implemented Laguerre iteration. The purpose of the split-merge process is to separate the eigenvalues of the target matrix T and obtain initial approximations to these eigenvalues. The Laguerre iteration is used because of its superior global properties and rapid convergence. There are other possible substitutes for the elements of our method.

- *The use of rank-one tearing in splitting.* It was shown in §6.1 that the rank-one tearing makes a larger perturbation to the spectrum and hence requires more iteration steps in merging. Nevertheless, its distinguished separation property and deflation potential may serve as a trade-off.
- *The use of the Newton iteration in merging.* The Newton iteration can certainly be used as a substitute for the Laguerre iteration in our method. Then the process will be slower in convergence but the cost per step become less. The major disadvantage of using the Newton iteration is the loss of the global monotonic convergence property. An efficient algorithm based on the Newton iteration may be developed if such difficulty can be resolved.

• *Acceleration of bisection.* Instead of splitting the matrix, one can split the spectrum by bisection/multisection using the Sturm sequence. After the eigenvalues are well separated, the Laguerre iteration can be used to achieve rapid convergence. This would be similar to the idea of [2]. The difficulties of this approach seem to be:

(1) if the spectrum can not be well separated (e.g., when clusters exist), the algorithm would be reduced to bisection/multisection;
(2) usually good initial points for the iteration are not available unless enough bisection steps have been taken.

7. Proofs of the theorems.

7.1. Proof of Theorem 2.3.

LEMMA 7.1. *Let* $f(\lambda) \equiv \prod_{i=1}^{n}(\lambda - \lambda_i)$ *be a polynomial with real zeros* $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$. *Then, for any real number* x *for which* $f(x) \neq 0$, *there is a parameter* $\theta \neq x$ *such that* $L_{r+}(x)$ *and* $L_{r-}(x)$ *defined in* (17) *are the two zeros of the quadratic function in* y

$$\phi(\theta, y) \equiv (x-y)^2 \sum_{i=1}^{n} \left(\frac{\theta - \lambda_i}{x - \lambda_i}\right)^2 - r(\theta - y)^2 \quad \text{with } r < n. \tag{31}$$

Proof. Let $\eta = x - y$ and $\mu = \theta - x$. Then

$$\sum_{i=1}^{n} \left(\frac{\theta - \lambda_i}{x - \lambda_i}\right)^2 = \mu^2 s_2 + 2\mu s_1 + n$$

where

$$s_1 = \frac{f'(x)}{f(x)}, \qquad s_2 = \frac{f'(x)^2 - f(x)f''(x)}{f(x)^2}.$$

With these notations, $\phi(\theta, y) = 0$ can be rewritten as

$$\mu^2(\eta^2 s_2 - r) + 2\mu\eta(\eta s_1 - r) + \eta^2(n - r) = 0, \tag{32}$$

which can be considered a quadratic equation in $\mu = \theta - x$. By making the discriminant Δ of (32) in μ zero, we have

$$[\eta(\eta s_1 - r)]^2 - (\eta^2 s_2 - r)\eta^2(n - r) = 0.$$

And, consequently, for

$$\eta = \frac{n}{s_1 \pm \sqrt{\frac{n-r}{r}(ns_2 - s_1)}}$$

or

$$y = x - \eta = x - \frac{n}{s_1 \pm \sqrt{\frac{n-r}{r}(ns_2 - s_1)}} = L_{r\pm}(x),$$

equation (32) in μ has a double zero μ_*. It is clear that $\mu_* \neq 0$. Otherwise, $\eta = 0$ and thereby $L_{r\pm}(x) = x$, which leads to $f(x) = 0$. Thus $\phi(\theta, L_{r\pm}(x)) = 0$ for $\theta \neq x$. □

Proof of Theorem 2.3. We shall only prove (18). Inequalities in (19) follow by a similar argument.

First, we show that there is always a unique zero $y(\theta)$ of (31) in $(x, \lambda_{m+r}]$ for every $\theta \neq x$. In fact, if $\theta < x$, then $\phi(\theta, x) = -r(\theta - y)^2 < 0$ and

$$\begin{aligned}\phi(\theta, \lambda_{m+r}) &\geq \sum_{i=1}^{r}\left[\left(\frac{x-\lambda_{m+r}}{x-\lambda_{m+i}}\right)^2(\theta-\lambda_{m+i})^2-(\theta-\lambda_{m+r})^2\right]\\ &> \sum_{i=1}^{r}\left[\left(\frac{\theta-\lambda_{m+r}}{\theta-\lambda_{m+i}}\right)^2(\theta-\lambda_{m+i})^2-(\theta-\lambda_{m+r})^2\right]=0.\end{aligned}$$

Hence there is a zero of $\phi(\theta, \cdot)$ in (x, λ_{m+r}). This zero is unique because $\phi(\theta, \cdot)$ is quadratic. For $\theta > x$, equation (31) can be rewritten as

$$\left(\frac{\theta-y}{x-y}\right)^2 = \frac{\sum_{i=1}^{n}\left(\frac{\theta-\lambda_i}{x-\lambda_i}\right)^2}{r} \quad (>0).$$

As a continuous function of y in (x, θ), $\left(\frac{\theta-y}{x-y}\right)^2$ is monotonically decreasing from $+\infty$ to 0. Thus there is a unique $y(\theta) \in (x, \theta)$ satisfying (31) for every $\theta > x$. If $\theta \leq \lambda_{m+r}$, $y(\theta) < \theta \leq \lambda_{m+r}$. If $\theta > \lambda_{m+r}$,

$$\begin{aligned}\phi(\theta, \lambda_{m+r}) &\geq \sum_{i=1}^{r}\left[\left(\frac{x-\lambda_{m+r}}{x-\lambda_{m+i}}\right)^2(\theta-\lambda_{m+i})^2-(\theta-\lambda_{m+r})^2\right]\\ &\geq \sum_{i=1}^{r}\left[(\theta-\lambda_{m+i})^2-(\theta-\lambda_{m+r})^2\right] \geq 0.\end{aligned}$$

Thus $y(\theta) \in (x, \lambda_{m+r}]$ since $\phi(\theta, x) < 0$.

It is easy to verify that $L_{r+}(x)$ is the closest zero of $\phi(\theta, y)$ to the right of x for some θ, that is, $y(\theta)$.

The inequality $L_+(x) < L_{r+}(x)$ for $r > 1$ is trivial in case of $-\frac{f'(x)}{f(x)} > 0$. □

7.2. Proof of Theorem 3.1.

LEMMA 7.2 (Strict interlacing property). *Let A be an unreduced symmetric tridiagonal matrix. For $m = 1, 2, \ldots, n$, let $A^{(m)}$ be the leading $m \times m$ principal minor (i.e., $A^{(m)}$ consists of entries of A in the first m rows and m columns) with eigenvalues $\{\lambda_i^{(m)}\}_{i=1}^{m}$ in ascending order. Then*

$$\lambda_1^{(m+1)} < \lambda_1^{(m)} < \lambda_2^{(m+1)} < \lambda_2^{(m)} < \cdots < \lambda_m^{(m+1)} < \lambda_m^{(m)} < \lambda_{m+1}^{(m+1)}.$$

Proof. See [24, p. 131]. □

LEMMA 7.3 [24, p. 193]. *Let A and B be $n \times n$ symmetric matrices and $\lambda_i(A)$ and $\lambda_i(A + B)$ be the ith smallest eigenvalues of A and $A + B$, respectively. Suppose B has π positive and ν negative eigenvalues. Then*

$$\lambda_{i-\nu}(A) \leq \lambda_i(A + B) \leq \lambda_{i+\pi}(A)$$

with the convention $\lambda_0(\cdot) = -\infty$ and $\lambda_{n+1}(\cdot) = +\infty$.

Proof of Theorem 3.1. The matrix T_0 is the $k \times k$ leading principal minor of T. By the interlacing property given in Lemma 7.2, the spectrum of T_0 is contained in (λ_1, λ_n). We may permute the rows and columns of T in reverse order, so that the accordingly permuted T_1 is

an $(n-k)\times(n-k)$ leading principal minor of permuted T. Hence the spectrum of T_1 is also contained in (λ_1,λ_n). It follows that $\lambda_1<\hat{\lambda}_1$ and $\hat{\lambda}_n<\lambda_n$. Since $T=\hat{T}+P^{(2)}$ where $P^{(2)}$ is defined in (27) with $\|P^{(2)}\|=|\beta_k|$, by the Weyl lemma, $|\lambda_j-\hat{\lambda}_j|\le|\beta_k|$. Hence $\lambda_1-|\beta_k|\le\lambda_1<\hat{\lambda}_1$ and $\hat{\lambda}_n+|\beta_k|\ge\lambda_n>\hat{\lambda}_n$. The eigenvalues of $P^{(2)}$ are 0 and $\pm\beta_k$, thus, by Lemma 7.3, $\hat{\lambda}_{i-1}\le\lambda_i\le\hat{\lambda}_{i+1}$ and $\lambda_{i-1}\le\hat{\lambda}_i\le\lambda_{i+1}$. It remains to show that $\hat{\lambda}_i\ne\lambda_{i-1}$, $\hat{\lambda}_i\ne\lambda_{i+1}$, and $\lambda_i\ne\hat{\lambda}_{i-1}$, $\lambda_i\ne\hat{\lambda}_{i+1}$. We shall prove $\hat{\lambda}_i\ne\lambda_{i-1}$. The remaining cases follow by similar arguments.

Consider the homotopy

$$\begin{aligned}h(\lambda,t)&=\det[(1-t)\hat{T}+tT-\lambda I]\\&=\det[T(t)-\lambda I],\end{aligned}$$

where $T(\lambda)\equiv(1-t)\hat{T}+tT$. Let $\lambda_i(t)$ denote the ith smallest eigenvalue of $T(t)$. Then every $\lambda_i(t)$ is a continuous function of t. For $t\ne0$, $T(t)$ is unreduced, so $\lambda_1(t)<\cdots<\lambda_n(t)$, and the graphs of $\lambda_i(t)$'s are disjoint except possibly at $t=0$.

By the three-term recurrence (6), it is easy to see that $h(\lambda,t)=t^2\beta_k^2f_1(\lambda)+f_2(\lambda)$ where f_1 and f_2 are polynomials in λ. Suppose $\hat{\lambda}_i=\lambda_{i-1}=\lambda_*$, then $h(\lambda_i(0),0)=h(\lambda_{i-1}(1),1)=0$, which implies $f_1(\lambda_*)=f_2(\lambda_*)=0$. Thus the constant function $\lambda(t)\equiv\lambda_*$ satisfies $h(\lambda(t),t)=0$ for all t. This constant function $\lambda(t)$ must be one of $\{\lambda_j(t)\}_{j=1}^n$. Since $\lambda(1)=\lambda_*=\lambda_{i-1}(1)$, so $\lambda(t)\equiv\lambda_{i-1}(t)$ and hence $\lambda_{i-1}(t)\equiv\lambda_*=\hat{\lambda}_i$. Accordingly, $\lambda_{i-1}(0)=\hat{\lambda}_{i-1}=\hat{\lambda}_i$, or $\hat{\lambda}_i$ is a double eigenvalue of $\hat{T}$. Since both T_0 and T_1 are unreduced, they must have the same eigenvalue λ_* at which $f_1(\lambda_*)=0$.

Now, by differentiating

$$h(\lambda,t)=t^2\beta_k^2f_1(\lambda)+f_2(\lambda)$$

$$=\det\begin{pmatrix}&&&|&&&\\&T_0-\lambda I&&|&&&\\&&&|&t\beta_k&&\\-&------&--&+&--&------&-\\&&t\beta_k&|&&&\\&&&|&&T_1-\lambda I&\\&&&|&&&\end{pmatrix}$$

twice with respect to t, one can verify that $f_1(\lambda)$ is the characteristic polynomial of the $(n-2)\times(n-2)$ matrix

$$\begin{pmatrix}\tilde{T}_0&0\\0&\tilde{T}_1\end{pmatrix}$$

where $\tilde{T}_0$ is the leading $(k-1)\times(k-1)$ minor of T_0, and $\tilde{T}_1$ is equal to the submatrix T_1 with the first row and the first column deleted. By the interlacing property, $f_1(\lambda_*)\ne0$ since λ_* is an eigenvalue of both T_0 and T_1 and thus cannot be an eigenvalue of either $\tilde{T}_0$ or $\tilde{T}_1$. This contradiction proves $\hat{\lambda}_i\ne\lambda_{i-1}$. □

Acknowledgment. We would like to acknowledge the valuable suggestions from Zhaojun Bai, James Demmel, Stanley Eisenstat, Kuiyuan Li, Beresford Parlett, and anonymous referees.

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] H. BERNSTEIN, *An Accelerated Bisection Method for the Calculation of Eigenvalues of Symmetric Tridiagonal Matrix*, Computer Science Dept., Tech. Report 79, Courant Institute, New York, NY, 1983.

[3] R. BHATIA, *Perturbation Bounds for Matrix Eigenvalues*, John Wiley & and Sons, Inc., New York, 1987.

[4] J. J. J. M. CUPPEN, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[5] J. DEMMEL, J. DU CROZ, S. HAMMARLING, AND D. SORENSEN, *Guidelines for the design of symmetric eigenroutines, svd, and iterative refinement and condition estimation for linear systems*, MCS-TM-111, (LAPACK Working Note 4), Argonne National Laboratory, 1988.

[6] J. DEMMEL AND W. KAHAN, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1988), pp. 873–912.

[7] 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.

[8] J. G. F. FRANCIS, *The QR transformation, parts* I *and* II, Computer J., 4 (1961), pp. 265–271, pp. 332–345.

[9] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations, 2nd Ed.*, The Johns Hopkins University Press, Baltimore, MD, 1989.

[10] R. T. GREGORY AND D. L. KARNEY, *A Collection of Matrices for Testing Computational Algorithms*, Robert E. Krieger Publishing Company, Huntington, NY, 1978.

[11] M. GU AND S. C. EISENSTAT, *A Divide-and-Conquer Algorithm for the Symmetric Tridiagonal Eigenproblem*, Research Report YALEU/DCS/RR-916, Dept. of Computer Science, Yale University, New Haven, CT, 1992.

[12] L. J. HUANG, *Parallel Homotopy Algorithm for Large Sparse Symmetric Eigenproblems*. Ph.D. thesis, Michigan State University, East Lansing, MI, 1992.

[13] E. R. JESSUP, *Parallel Solution of the Symmetric Tridiagonal Eigenproblem*, Ph.D. thesis, Yale University, New Haven, CT, 1989.

[14] E. R. JESSUP AND I. C. F. IPSEN, *Improving the accuracy of inverse iteration*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 550–572.

[15] W. KAHAN, *Accurate eigenvalues of a symmetric tridiagonal matrix*, Tech. Report CS41, Computer Science Dept., Stanford University, Stanford, CA (1966) (revised June 1968).

[16] K. LI AND T. Y. LI, *An algorithm for symmetric tridiagonal eigenproblems—divide and conquer with homotopy continuation*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 735–751.

[17] T. Y. LI AND N. H. RHEE, *Homotopy algorithm for symmetric eigenvalue problems*, Numer. Math., 55 (1989), pp. 265–280.

[18] T. Y. LI, N. H. RHEE, AND Z. ZENG, *An efficient and accurate parallel algorithm for the singular value problem of bidiagonal matrices*, Michigan State University, East Lansing, MI, preprint; Numer. Math., submitted.

[19] T. Y. LI AND Z. ZENG, *Homotopy-determinant algorithm solving nonsymmetric eigenvalue problems*, Math. Comp., 59 (1992), pp. 483–502.

[20] T. Y. LI, Z. ZENG, AND L. CONG, *Solving eigenvalue problems of real nonsymmetric matrices with real homotopies*, SIAM J. Numer. Anal., 29 (1992), pp. 229–248.

[21] T. Y. LI, H. ZHANG, AND X. H. SUN, *Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problems*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 464–485.

[22] S.-S. LO, B. PHILLIPE, AND A. SAMEH, *A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problems*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 155–165.

[23] B. N. PARLETT, *Laguerre's method applied to the matrix eigenvalue problem*, Math. Comp., 18 (1964), pp. 464–485.

[24] ———, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[25] D. C. SORENSEN AND P. P. T. TANG, *On the orthogonality of eigenvectors computed by divide-and-conquer techniques*, SIAM. J. Numer. Anal., 28 (1991), pp. 1752–1775.

[26] C. TREFFTZ, C. C. HUANG, P. MCKINLEY, T. Y. LI, AND Z. ZENG, *A scalable eigenvalue solver for symmetric tridiagonal matrices*, Michigan State University, East Lansing, MI, preprint.

[27] R. C. WARD, *The QR algorithm and Hyman's method on vector computers*, Math. Comp., 30 (1976), pp. 132–142.

[28] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

[29] Z. ZENG, *Homotopy-Determinant Method for Solving Matrix Eigenvalue Problems and its Parallelizations*, Ph.D. thesis, Michigan State University, East Lansing, MI, 1991.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1174–1189, September 1994

012

THREE-DIMENSIONAL INVERSE OBSTACLE SCATTERING FOR TIME HARMONIC ACOUSTIC WAVES: A NUMERICAL METHOD*

LUCIANO MISICI† AND FRANCESCO ZIRILLI‡

Abstract. A numerical method for a three-dimensional inverse acoustic scattering problem is considered. From the knowledge of several far fields patterns of the Helmholtz equation a closed surface ∂D representing the boundary of an unknown obstacle D is reconstructed. The obstacle D is supposed to be acoustically soft or acoustically hard or characterized by a given acoustic impedance.

Key words. acoustic scattering, inverse obstacle problem

AMS subject classifications. 35R30, 35J05

1. Introduction.

Let $\mathbf{R}^3$ be the three-dimensional euclidean space, $\underline{x}=(x, y, z)^T \in \mathbf{R}^3$ be a generic vector, and the superscript T denote the transpose operation. The euclidean scalar product will be denoted with $(\cdot, \cdot)$ and $\| \cdot \|$ will denote the euclidean norm.

Let $D \subset \mathbf{R}^3$ be a bounded simply connected domain with smooth boundary ∂D; in the following we assume that D contains the origin. Let $u^i(\underline{x})$ be an incoming acoustic plane wave, that is,

$$u^i(\underline{x}) = e^{ik(\underline{x},\underline{\alpha})}, \tag{1.1}$$

where $k > 0$ is the wave number and $\underline{\alpha} \in \mathbf{R}^3$ is a fixed unit vector (i.e., $\|\underline{\alpha}\| = 1$). Let us denote with $u^s(\underline{x})$ the acoustic field scattered by the obstacle D and with $u(\underline{x})$ the total acoustic field, that is,

$$u(\underline{x}) = u^i(\underline{x}) + u^s(\underline{x}). \tag{1.2}$$

The total acoustic field $u(\underline{x})$ satisfies the Helmholtz equation

$$\Delta u(\underline{x}) + k^2 u(\underline{x}) = 0 \qquad \underline{x} \in \mathbf{R}^3 \setminus D, \tag{1.3}$$

where $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2 + \partial^2/\partial z^2$ is the Laplace operator. The scattered field $u^s(\underline{x})$ satisfies the Sommerfeld radiation condition at infinity

$$\lim_{r\to\infty} r \left\{ \frac{\partial u^s}{\partial r} - iku^s \right\} = 0, \tag{1.4}$$

where $r = \|\underline{x}\|$. Moreover the total acoustic field $u(\underline{x})$ satisfies a boundary condition on ∂D. This boundary condition is the mathematical counterpart of the physical character of the obstacle, that is, for acoustically soft obstacles we require the Dirichlet boundary condition

$$u(\underline{x}) = 0, \qquad \underline{x} \in \partial D; \tag{1.5}$$

for acoustically hard obstacles we require the Neumann boundary condition

$$\frac{\partial u(\underline{x})}{\partial n} = 0, \qquad \underline{x} \in \partial D, \tag{1.6}$$

*Received by the editors September 20, 1991; accepted for publication (in revised form) August 7, 1993. The research reported in this paper has been made possible through the support and sponsorship of the United States Government through the Air Force Office of Scientific Research under contract AFOSR 90-0228 with the Università di Camerino.

†Dipartimento di Matematica e Fisica, Università di Camerino, 62032 Camerino (MC), Italy (misici@camvax.cineca.it).

‡Dipartimento di Matematica "G. Castelnuovo", Università di Roma "La Sapienza," 00185 Roma, Italy (apzrm@itcaspur.bitnet).

where $n(\underline{x})$, $\underline{x} \in \partial D$ is the unit outward normal to ∂D at the point $\underline{x}$; and finally for obstacles characterized by an acoustic impedance we require the mixed boundary condition

$$u(\underline{x}) + \chi \frac{\partial u(\underline{x})}{\partial n} = 0, \qquad \underline{x} \in \partial D. \tag{1.7}$$

We assume χ to be a real constant to derive the relations of §3; however this assumption can be avoided.

We call the direct problem the problem of determining the scattered field $u^s(\underline{x})$, $\underline{x} \in \mathbf{R}^3 \backslash D$ given the incoming field $u^i(\underline{x})$, the obstacle D, and its physical character, that is, given one of the three boundary conditions (1.5), (1.6), (1.7). The direct problem is a boundary value problem for the Helmholtz equation and has been widely studied. For an exposition of several mathematical results on the direct problem, see, for example, [1]. In particular it can be shown [1] that the scattered field $u^s(\underline{x})$ corresponding to the boundary condition (1.5), (1.6), or (1.7) has the following expansion:

$$u^s(\underline{x}) = \frac{e^{ikr}}{r} F_0(\underline{\hat{x}}, k, \underline{\alpha}) + O\left(\frac{1}{r^2}\right), \qquad r \to \infty \tag{1.8}$$

so that when $r \to \infty$ the leading term of the expansion in inverse powers of r is given by a spherical wave e^{ikr}/r coming out from the origin modulated by the "far field" F_0. We note that F_0 depends on k, $\underline{\alpha}$, which are the parameters characterizing the incoming wave (1.1) and on $\underline{\hat{x}} = \underline{x}/\|\underline{x}\|$ for $\underline{x} \neq 0$. We note that the Helmholtz equation (1.3) is obtained from the wave equation assuming that the incoming field and the corresponding scattered field are time harmonic; that is, their time dependence is given by a factor $e^{i\omega t}$ where ω is a constant.

The inverse problem that we consider here is the following: given the character of the obstacle (i.e., acoustically soft, hard, or characterized by an acoustic impedance) and the far field $F_0(\underline{\hat{x}}, k, \underline{\alpha})$ for one or several incoming waves $u^i(\underline{x})$ with different incident directions $\underline{\alpha}$ and/or wave number k, determine the boundary of the obstacle ∂D. This inverse problem is known to be ill posed and, due to its great interest both in mathematics and in several application fields, has been widely studied; for a review see [2].

The numerical methods used to solve the inverse problem considered here can be divided in two types: the first type consists of an iterative procedure that at each step requires the numerical solution of a direct problem; the second type consists of genuine methods for the inverse problem that do not require the solution of the direct problem. In the first type we mention the work of Roger [3], Murch, Tan, and Wall [4], Wang and Chen [5], Angell, Colton, and Kirsch [6], and Kristensson and Vogel [7]. In the second type we mention the work of Kirsch and Kress [8]–[10] and the work of Colton and Monk [11].

In this paper we introduce a numerical method to solve this inverse problem based on the Herglotz wave function method introduced by Colton and Monk in [11] and further developed by Misici, Zirilli, and their coauthors in [12]–[15]. In particular, based on previous work by Misici and Zirilli [14], we extend the Herglotz function method introduced in [11] for acoustically soft obstacles to hard obstacles or obstacles characterized by an acoustic impedance. The analytical relations obtained are exploited to build up numerical algorithms. Finally these algorithms are efficient in the so-called resonance region, that is, when

$$kL = O(1), \tag{1.9}$$

where L is a characteristic length of the obstacle D. In §2 we derive the analytical relations that are the basis of the numerical methods. In §3 the basic numerical method developed for the solution of the inverse problem is presented. In §4 some special features of the reconstruction

procedure in the case of acoustically soft obstacles are shown. Finally in §5 we introduce the test problems used to test the methods of §§3 and 4 and we show the numerical results obtained.

2. The mathematical formulation of the inverse problem. For $\underline{x}, \underline{y} \in \mathbf{R}^3$ let

$$\Phi(\underline{x}, \underline{y}) = \frac{e^{ik\|\underline{x}-\underline{y}\|}}{4\pi\|\underline{x}-\underline{y}\|} \tag{2.1}$$

be the Green's function of the Helmholtz operator that satisfies the Sommerfeld radiation condition at infinity. It is easy to see that

$$\Phi(\underline{x}, \underline{y}) = \frac{e^{ik\|\underline{x}\|}}{4\pi k\|\underline{x}\|} e^{-ik(\hat{\underline{x}}, \underline{y})} + O\left(\frac{1}{\|\underline{x}\|^2}\right) \quad \text{when } \underline{x} \to \infty; \tag{2.2}$$

moreover from the Helmholtz formula [14] we have

$$\int_{\partial D} \left[u(\underline{y}) \frac{\partial \Phi(\underline{x}, \underline{y})}{\partial n(\underline{y})} - \Phi(\underline{x}, \underline{y}) \frac{\partial u(\underline{y})}{\partial n(\underline{y})} \right] d\sigma(\underline{y}) = \begin{cases} -u^i(\underline{x}) & \text{if } \underline{x} \in D, \\ u^s(\underline{x}) & \text{if } \underline{x} \in \mathbf{R}^3 \setminus D, \end{cases} \tag{2.3}$$

where $d\sigma(\underline{y})$ is the surface measure on ∂D. Substituting (2.2) in (2.3) and using (1.8) we have

$$F_0(\hat{\underline{x}}, k, \underline{\alpha}) = \frac{1}{4\pi} \int_{\partial D} \left[u(\underline{y}) \frac{\partial (e^{-ik(\hat{\underline{x}}, \underline{y})})}{\partial n(\underline{y})} - \frac{\partial u(\underline{y})}{\partial n(\underline{y})} e^{-ik(\hat{\underline{x}}, \underline{y})} \right] d\sigma(\underline{y}). \tag{2.4}$$

Let $B = \{\underline{x} \in \mathbf{R}^3 \mid \|\underline{x}\| \leq 1\}$, ∂B be the boundary of B and $d\lambda$ be the surface measure on ∂B. We denote with $\mathbf{L}^2(\partial B, d\lambda)$ the space of square integrable complex functions with respect to the measure $d\lambda$. For $g(\hat{\underline{x}}) \in \mathbf{L}^2(\partial B, d\lambda)$ and (2.4) interchanging the integrals we have

$$\begin{aligned} &\int_{\partial B} F_0(\hat{\underline{x}}, k, \underline{\alpha}) \overline{g(\hat{\underline{x}})} d\lambda(\hat{\underline{x}}) \\ &= \frac{1}{4\pi} \int_{\partial B} \overline{g(\hat{\underline{x}})} d\lambda(\hat{\underline{x}}) \int_{\partial D} \left[u(\underline{y}) \frac{\partial (e^{-ik(\hat{\underline{x}}, \underline{y})})}{\partial n(\underline{y})} - \frac{\partial u(\underline{y})}{\partial n(\underline{y})} e^{-ik(\hat{\underline{x}}, \underline{y})} \right] d\sigma(\underline{y}) \\ &= \frac{1}{4\pi} \int_{\partial D} \left[u(\underline{y}) \frac{\overline{\partial v(\underline{y})}}{\partial n(\underline{y})} - \frac{\partial u(\underline{y})}{\partial n(\underline{y})} \overline{v(\underline{y})} \right] d\sigma(\underline{y}), \end{aligned} \tag{2.5}$$

where

$$v(\underline{y}) = \int_{\partial B} g(\hat{\underline{x}}) e^{ik(\hat{\underline{x}}, \underline{y})} d\lambda(\hat{\underline{x}}). \tag{2.6}$$

It is easy to see by differentiating under the integral sign that $v(\underline{y})$ is a solution of the Helmholtz equation for every $\underline{y} \in \mathbf{R}^3$.

Let Σ_1, Σ_2, Σ_3 be the sets of the eigenvalues of the Laplace operator inside the domain D with the Dirichlet boundary condition (1.5), the Neumann boundary condition (1.6), or the mixed boundary condition (1.7), respectively. We give the following definitions.

DEFINITION 2.1. *Given $-k^2 \notin \Sigma_1$, let $w_1(\underline{y})$ be the unique solution of the following boundary value problem:*

$$(\Delta + k^2) w_1(\underline{y}) = 0, \qquad \underline{y} \in D, \tag{2.7}$$

$$w_1(\underline{y}) = -\frac{e^{-ik\|\underline{y}\|}}{k\|\underline{y}\|}, \qquad \underline{y} \in \partial D. \tag{2.8}$$

We say that D is a Herglotz domain with respect to the Dirichlet boundary condition if there exists $g_1(\hat{\underline{x}}) \in \mathbf{L}^2(\partial B, d\lambda)$ such that

$$w_1(\underline{y}) = \int_{\partial B} g_1(\hat{\underline{x}}) e^{ik(\hat{\underline{x}},\underline{y})} d\lambda(\hat{\underline{x}}). \tag{2.9}$$

In a similar way we have the following definition.

DEFINITION 2.2. *Given $-k^2 \notin \Sigma_2$, let $w_2(\underline{y})$ be the unique solution of the following boundary value problem:*

$$(\Delta + k^2) w_2(\underline{y}) = 0, \qquad \underline{y} \in D, \tag{2.10}$$

$$\frac{\partial w_2(\underline{y})}{\partial n(\underline{y})} = -\frac{\partial}{\partial n(\underline{y})}\left(\frac{e^{-ik\|\underline{y}\|}}{k\|\underline{y}\|}\right), \qquad \underline{y} \in \partial D. \tag{2.11}$$

We say that D is a Herglotz domain with respect to the Neumann boundary condition if there exists $g_2(\hat{\underline{x}}) \in \mathbf{L}^2(\partial B, d\lambda)$ such that

$$w_2(\underline{y}) = \int_{\partial B} g_2(\hat{\underline{x}}) e^{ik(\hat{\underline{x}},\underline{y})} d\lambda(\hat{\underline{x}}). \tag{2.12}$$

DEFINITION 2.3. *Given $\chi \in \mathbf{R}$ and $-k^2 \notin \Sigma_3$, let $w_3(\underline{y})$ be the unique solution of the following boundary value problem:*

$$(\Delta + k^2) w_3(\underline{y}) = 0, \qquad \underline{y} \in D, \tag{2.13}$$

$$w_3(\underline{y}) + \chi \frac{\partial w_3(\underline{y})}{\partial n(\underline{y})} = -\left(1 + \chi \frac{\partial}{\partial n(\underline{y})}\right)\left(\frac{e^{-ik\|\underline{y}\|}}{k\|\underline{y}\|}\right), \qquad \underline{y} \in \partial D. \tag{2.14}$$

We say that D is a Herglotz domain with respect to the mixed boundary condition if there exists $g_3(\hat{\underline{x}}) \in \mathbf{L}^2(\partial B, d\lambda)$ such that

$$w_3(\underline{y}) = \int_{\partial B} g_3(\hat{\underline{x}}) e^{ik(\hat{\underline{x}},\underline{y})} d\lambda(\hat{\underline{x}}). \tag{2.15}$$

To our knowledge a characterization of Herglotz domains is not known; however it is easy to see that the class of Herglotz domains is not empty. In fact a straightforward computation shows that the sphere of center the origin is an Herglotz domain in the sense of Definitions 2.1–2.3; moreover the numerical experience of §5 can be regarded as experimental evidence that the domains considered satisfy the previous definitions.

In the following we consider only domains D that satisfy the appropriate Herglotz condition, that is, Definitions 2.1, 2.2, or 2.3. In the case of the inverse problem for the acoustically soft obstacle from (2.5), using (1.5) and (2.8) we have

$$\int_{\partial B} F_0(\hat{\underline{x}}, k, \underline{\alpha}) \overline{g_1(\hat{\underline{x}})} d\lambda(\hat{\underline{x}}) = \frac{1}{k} \quad \forall\, \underline{\alpha} \in \partial B. \tag{2.16}$$

Reasoning in the same way we have

$$\int_{\partial B} F_0(\hat{\underline{x}}, k, \underline{\alpha})\overline{g_2(\hat{\underline{x}})}d\lambda(\hat{\underline{x}}) = \frac{1}{k} \quad \forall\, \underline{\alpha} \in \partial B \tag{2.17}$$

for the acoustically hard obstacle and

$$\int_{\partial B} F_0(\hat{\underline{x}}, k, \underline{\alpha})\overline{g_3(\hat{\underline{x}})}d\lambda(\hat{\underline{x}}) = \frac{1}{k} \quad \forall\, \underline{\alpha} \in \partial B \tag{2.18}$$

for the obstacle characterized by an acoustic impedance.

The numerical method for the inverse problem for the acoustically soft obstacle is based on the relations (2.16), (2.9), (2.8), which connect the data, i.e., the far fields to the unknown ∂D. In a similar way we will exploit (2.17), (2.12), (2.11) to solve the inverse problem for the acoustically hard obstacle and (2.18), (2.15), (2.14) to solve the inverse problem for the obstacle characterized by an acoustic impedance.

3. The numerical method. Given $D \subset \mathbf{R}^3$ and the boundary condition (1.5), (1.6), or (1.7) satisfied by u on ∂D, we will exploit numerically the analytic relations derived in §2 as follows. Since most of the content of this section is independent of the boundary conditions chosen, to fix the ideas we consider the inverse problem for acoustically soft obstacles, i.e., the relations (2.16), (2.9), (2.8). When necessary we will comment on the peculiar features of the corresponding problems for acoustically hard obstacles or obstacles characterized by an acoustic impedance. Our general strategy can be summarized in three points.

(i) Use (2.16) to go from the knowledge of the far fields F_0 to the Herglotz kernel $g_1(\hat{\underline{x}})$ of the domain D.

(ii) Use (2.9) to go from the knowledge of the Herglotz kernel $g_1(\hat{\underline{x}})$ to the Herglotz wave function $w_1(\underline{y})$.

(iii) Use (2.8) to go from the knowledge of the Herglotz wave function $w_1(\underline{y})$ to the boundary of the obstacle ∂D.

More precisely, given D, let $\Omega_1 = \{\underline{\alpha}_i \in \partial B \mid i = 1, 2, \ldots, N\}$ be the set of directions of the incoming waves, $\Omega_2 = \{k_i \in \mathbf{R} \mid -k_i^2 \notin \Sigma_1\,,\ i = 1, 2, \ldots, N_1\}$ be the set of the nonresonant wave numbers of the incoming waves and $\Omega_3 = \{\hat{\underline{x}}_i \in \partial B \mid i = 1, 2, \ldots, M\}$ be the set of directions where the far fields F_0 are measured. For $i = 1, 2, 3$ we assume that the elements of Ω_i are distinct. We observe that is not possible to know a priori that the wave numbers chosen are nonresonant. The data of our inverse problems will be the numbers $F_{i,i_1,j}$, $i = 1, 2, \ldots, N$, $i_1 = 1, 2, \ldots, N_1$, $j = 1, 2, \ldots, M$ that represent the measurements of $F_0(\hat{\underline{x}}_j, k_{i_1}, \underline{\alpha}_i)$. In the numerical experience shown in §5 these data are obtained by solving numerically the direct problem (1.3)–(1.5).

Let (θ, ϕ) be the polar angles so that

$$\hat{\underline{x}} = \hat{\underline{x}}(\theta, \phi) = (\sin\theta\cos\phi,\ \sin\theta\sin\phi,\ \cos\theta)^T \tag{3.1}$$

and $U_{lm}(\hat{\underline{x}}) = \gamma_{lm} P_l^m(\cos\theta)\cos m\phi$, $V_{lm}(\hat{\underline{x}}) = \gamma_{lm} P_l^m(\cos\theta)\sin m\phi$, $l = 0, 1, 2, \ldots$ $m = 0, 1, \ldots, l$ be the spherical harmonics, where γ_{lm} are the normalization factors in $\mathbf{L}^2(\partial B, d\lambda)$ and P_l^m are the Legendre functions. It is well known that

$$\{U_{lm}, V_{lm}\}_{\substack{l=0,1,\ldots \\ m=0,1,\ldots,l}} \quad \text{is an orthonormal complete set of } \mathbf{L}^2(\partial B, d\lambda).$$

From the data our computation proceeds as follows.

Step 1. From the data to the coefficients of the expansion in spherical harmonics of the far field $F_0(\hat{\underline{x}}, k_{i_1}, \underline{\alpha}_i)$, $i_1 = 1, 2, \ldots, N_1$, $i = 1, 2, \ldots, N$.

Let us first consider the case $\Omega_3 = \{\underline{\hat{x}}_i \mid i = 1, 2, \ldots, M\}$. Given an integer $L_{\max} \geq 0$ we assume that the far field $F_0(\underline{\hat{x}}, k_{i_1}, \underline{\alpha}_i)$ can be approximated by a truncated expansion in spherical harmonics, that is,

$$F_0(\underline{\hat{x}}, k_{i_1}, \underline{\alpha}_i) = \sum_{l=0}^{L_{\max}} \sum_{m=0}^{l} F_{0,l,m,1}^{i_1,i} U_{lm}(\underline{\hat{x}}) + \sum_{l=1}^{L_{\max}} \sum_{m=1}^{l} F_{0,l,m,2}^{i_1,i} V_{lm}(\underline{\hat{x}}). \tag{3.2}$$

We note that the unknown coefficients appearing in the truncated expansion (3.2) are $(L_{\max} + 1)^2$ complex constants. To determine these coefficients we impose that

$$\sum_{l=0}^{L_{\max}} \sum_{m=0}^{l} F_{0,l,m,1}^{i_1,i} U_{lm}(\underline{\hat{x}}_j) + \sum_{l=1}^{L_{\max}} \sum_{m=1}^{l} F_{0,l,m,2}^{i_1,i} V_{lm}(\underline{\hat{x}}_j) = F_{i,i_1,j}, \quad j = 1, 2, \ldots, M. \tag{3.3}$$

Equations (3.3) are a linear system of M equations in $(L_{\max} + 1)^2$ unknowns, so that to determine the unknowns we need $M \geq (L_{\max} + 1)^2$. The condition number of the linear system (3.3) grows dramatically with the number of equations M, so that its numerical solution become practically unfeasible even for small values of $L_{\max}$ such as $L_{\max} = 4, 6, 8$.

The use of regularization procedures such as Tichonov regularization to solve (3.3) are helpful but insufficient to cure the ill conditioning. We note that the right-hand side of (3.3) is supposed to represent actual physical measurements that are affected by significant experimental errors so that ill conditioning is a true challenge. This ill conditioning problem is solved by reducing the number of unknowns to be determined and reducing in a similar way the number of equations to be solved. This is obtained exploiting the special features of the linear system (3.3).

Step 2. From the far fields $F_0(\underline{\hat{x}}, k_{i_1}, \underline{\alpha}_i) i_1 = 1, 2, \ldots, N_1$ $i = 1, 2, \ldots, N$ to the Herglotz kernel $g_{i_1}(\underline{\hat{x}})$.

Let $g_{i_1}(\underline{\hat{x}}) \in \mathbf{L}^2(\partial B, d\lambda)$ be the Herglotz kernel associated to the boundary condition (1.5), the domain D, and the wave number $k_{i_1} \in \Omega_2$. We assume that $g_{i_1}(\underline{\hat{x}})$ can be approximated by a truncated expansion in spherical harmonics, that is,

$$g_{i_1}(\underline{\hat{x}}) = \sum_{l=0}^{L_g} \sum_{m=0}^{l} g_{lm1}^{i_1} U_{lm}(\underline{\hat{x}}) + \sum_{l=1}^{L_g} \sum_{m=1}^{l} g_{lm2}^{i_1} V_{lm}(\underline{\hat{x}}), \tag{3.4}$$

where $0 \leq L_g \leq L_{\max}$ is an integer. Using the orthogonality properties of the spherical harmonics the relation (2.16) will be approximated with

$$\sum_{l=0}^{L_g} \sum_{m=0}^{l} F_{0,l,m,1}^{i_1,i} \overline{g_{lm1}^{i_1}} + \sum_{l=1}^{L_g} \sum_{m=1}^{l} F_{0,l,m,2}^{i_1,i} \overline{g_{lm2}^{i_1}} = \frac{1}{k_{i_1}}, \qquad i = 1, 2, \ldots, N. \tag{3.5}$$

Equations (3.5) are a linear system of N equations in the $(L_g + 1)^2$ unknown coefficients $\{g_{lm1}^{i_1}, g_{lm2}^{i_1}\}$, so that to determine the unknowns we need $N \geq (L_g + 1)^2$. We remark that the unknown coefficients $\{g_{lm1}^{i_1}, g_{lm2}^{i_1}\}$ are complex numbers. The number of incoming waves N can be drastically reduced if the unknown coefficients to be determined are reduced in a corresponding manner. This reduction can be achieved if we make some a priori assumptions about the symmetries of the obstacle D, and we assume that the same symmetry is conserved in the Herglotz kernel $g_{i_1}(\underline{\hat{x}})$. In this paper we consider the following three choices.

(i) ∂D is cylindrically symmetric with respect the z-axis, that is, $\partial D = \{\underline{x} = f(\theta)\underline{\hat{x}}(\theta, \phi) \mid 0 \leq \theta \leq \pi,\ 0 \leq \phi < 2\pi\}$ for some function f. In this case if we assume the same symmetry for $g_{i_1}(\underline{\hat{x}})$ we have $g_{lm1}^{i_1} = 0$ if $m > 0$ and $g_{lm2}^{i_1} = 0$ for every l, m so that the linear system (3.5) has only $L_g + 1$ unknowns and as a consequence we need only $N \geq L_g + 1$ incoming waves.

(ii) ∂D is cylindrically symmetric with respect to the z-axis and symmetric with respect to the equator. Arguing as in (i) we can conclude that the linear system (3.5) has only $[L_g/2]+1$ unknowns. With $[L_g/2]$ we have denoted the integer part of $L_g/2$.

(iii) ∂D is a general surface, that is, no special symmetries are assumed.

When it is necessary the system (3.5) is solved using a regularization procedure. We note that when some symmetry about ∂D is assumed and these symmetries are exploited as previously shown a large number of the coefficients $\{F^{i_1,i}_{0,l,m,1}, F^{i_1,i}_{0,l,m,2}\}$ of the system (3.5) determined in Step 1 are multiplied by unknowns $\{g^{i_1}_{lm1}, g^{i_1}_{lm2}\}$ that are assumed zero.

Step 3. From the Herglotz kernel $g_{i_1}(\hat{\underline{x}})$ to the Herglotz wave function $w_{1,i_1}(\underline{y})$, $i_1 = 1, 2, \ldots, N_1$.

From (2.9) and (3.4) an explicit computation gives us an approximated expression of $w_{1,i_1}(\underline{y})$, that is,

(3.6)
$$w_{1,i_1}(\underline{y}) = 4\pi \left\{ \sum_{l=0}^{L_g} \sum_{m=0}^{l} g^{i_1}_{lm1} i^l j_l(k_{i_1}\|\underline{y}\|) U_{lm}(\hat{\underline{y}}) + \sum_{l=1}^{L_g} \sum_{m=1}^{l} g^{i_1}_{lm2} i^l j_l(k_{i_1}\|\underline{y}\|) V_{lm}(\hat{\underline{y}}) \right\},$$

where $\hat{\underline{y}} = \underline{y}/\|\underline{y}\|$ and j_l is the spherical Bessel function of order l.

Step 4. From the Herglotz functions $w_{1,i_1}(\underline{y})$, $i_1 = 1, 2, \ldots, N_1$ to the boundary of the obstacle ∂D.

For simplicity we assume that there exists $0 < a < b < +\infty$ and a smooth function $f(\theta, \phi)$ such that $a < f(\theta, \phi) < b$, for all θ, ϕ and we have $\partial D = \{(r, \theta, \phi) \mid r = f(\theta, \phi), 0 \le \theta \le \pi, \ 0 \le \phi < 2\pi\}$.

The reconstruction of the boundary ∂D from the Herglotz functions is based on the relation (2.8) in the case of the acoustically soft obstacle, or the relation (2.11) in the case of the acoustically hard obstacle, or the relation (2.14) in the case of the obstacle characterized by an acoustic impedance. The relation (2.8) must be interpreted as a nonlinear equation that defines implicitly f as a function of θ and ϕ, the relations (2.11), (2.14), due to the presence of the $\partial/\partial n$ term, are nonlinear expressions involving f, $\frac{\partial f}{\partial \theta}$, $\frac{\partial f}{\partial \phi}$, that is, first-order partial differential equations. So we expect that the reconstruction of ∂D in the acoustically soft case should be easier than in the remaining cases. In §4 we will show an ad hoc procedure to exploit (2.8). Here we restrict our attention to a general procedure to obtain ∂D that can be applied always. We approximate ∂D with a truncated series of spherical harmonics:

$$f(\theta, \phi) = \sum_{l=0}^{L_\rho} \sum_{m=0}^{l} c_{lm1} U_{lm}(\hat{\underline{x}}) + \sum_{l=1}^{L_\rho} \sum_{m=1}^{l} c_{lm2} V_{lm}(\hat{\underline{x}}), \tag{3.7}$$

where $L_\rho \ge 0$ is chosen depending on ∂D, i.e., simple obstacles can be reconstructed with small L_ρ, that is, $L_\rho = 4, 6$. Moreover if it is assumed that ∂D has some of the symmetries previously considered, these symmetries can be easily translated into properties of the coefficients $\{c_{lm1}, c_{lm2}\}$, that is, the appropriate coefficients can be chosen zero. Let $\underline{c} = \{c_{lm1}, c_{lm2}\}$, $0 \le l \le L_\rho, 0 \le m \le l$ be the $(L_\rho + 1)^2$-dimensional real vector of the unknown coefficients; the unknown boundary is obtained by minimizing with respect to $\underline{c}$ the following functions:

$$I_1(\underline{c}) = \sum_{i_1=1}^{N_1} \frac{1}{k_{i_1}} \int_0^{2\pi} d\phi \int_0^{\pi} \sin\theta \left| w_{1,i_1} + \frac{e^{-ik_{i_1} f}}{k_{i_1} f} \right|^2 d\theta, \tag{3.8}$$

when the acoustically soft obstacle is considered;

$$I_2(\underline{c}) = \sum_{i_1=1}^{N_1} \frac{1}{k_{i_1}} \int_0^{2\pi} d\phi \int_0^{\pi} \sin\theta \left| \frac{\partial}{\partial n} \left(w_{2,i_1} + \frac{e^{-ik_{i_1} f}}{k_{i_1} f} \right) \right|^2 d\theta, \tag{3.9}$$

when the acoustically hard obstacle is considered;

$$I_3(\underline{c}) = \sum_{i_1=1}^{N_1} \frac{1}{k_{i_1}} \int_0^{2\pi} d\phi \int_0^{\pi} \sin\theta \left| \left(1 + \chi \frac{\partial}{\partial n}\right) \left(w_{3,i_1} + \frac{e^{-ik_{i_1} f}}{k_{i_1} f}\right) \right|^2 d\theta, \tag{3.10}$$

when the obstacle with acoustic impedance is considered.

In (3.8)–(3.10) the Herglotz wave functions obtained in Step 3 are computed in ($r = f(\theta, \phi), \theta, \phi$), $\frac{\partial}{\partial n}$ is the normal derivative with respect to the surface $r = f(\theta, \phi)$, and f is approximated with (3.7). The factor $1/k_{i_1}$ in (3.8)–(3.10) is chosen to make the addenda corresponding to different values of k of the same order of magnitude. Finally the integrals appearing in (3.8)–(3.10) are approximated by some simple quadrature rule. We observe that the functions $I_i(\underline{c})$, $i = 1, 2, 3$ are nonnegative functions and that, if we neglect the effects of the approximations introduced, the surface $r = f(\theta, \phi)$ corresponds to a point where $I_i(\underline{c})$ $i = 1, 2, 3$ is zero, that is, $r = f(\theta, \phi)$ is a global minimizer of $I_i(\underline{c})$. We note that in general the surface ∂D is only a proper subset of the set of points satisfying the relations (2.8) or (2.11) or (2.14).

When the minimization of $I_i(\underline{c})$ does not give a satisfactory reconstruction of ∂D we minimize $P_i(\underline{c})$ instead of $I_i(\underline{c})$, where

$$P_i(\underline{c}) = I_i(\underline{c}) + q(\underline{c}), \qquad i = 1, 2, 3 \tag{3.11}$$

and $q(\underline{c})$ is a penalization term. The penalization term $q(\underline{c})$ is chosen as follows:

$$q(\underline{c}) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} p_{ij} (f(\theta_i, \phi_j) - f_{ij}^{\star})^2, \tag{3.12}$$

where $p_{ij} > 0$ are weight factors, and $f_{ij}^{\star}$ are approximated values of f in the direction (θ_i, ϕ_j) that we assume known. For the acoustically soft obstacles the values $f_{ij}^{\star}$ can be obtained by solving (2.8) with $\theta = \theta_i$, $\phi = \phi_j$ as a nonlinear equation for f. In the remaining cases the nonlinear partial differential equations (2.11), (2.14) become nonlinear equations when

$$\frac{\partial f}{\partial \theta} = \frac{\partial f}{\partial \phi} = 0. \tag{3.13}$$

For a large class of surfaces (3.13) is satisfied in some special points such as the North Pole (i.e., $\theta = 0$, ϕ arbitrary), the South Pole (i.e., $\theta = \pi$, ϕ arbitrary) or the equator (i.e., $\theta = \frac{\pi}{2}, \phi$ arbitrary). At these points (2.11), (2.14) can be solved as nonlinear equations to obtain approximate values $f_{ij}^{\star}$ of f to be used in the penalization term (3.12).

The minimization of $I_i(\underline{c})$ or $P_i(\underline{c})$ is performed with one of the following three algorithms: DBCONF [17], that is, a quasi-Newton local minimization algorithm, SIGMA [18],[19], that is, a global minimization stochastic algorithm, or DUNLSJ [20], that is, a nonlinear least squares algorithm.

The quasi-Newton algorithm performs only a local minimization but is computationally cheaper than the global minimization algorithm. The nonlinear least squares algorithm is specifically suited for the minimization of $I_i(\underline{c})$ $i = 1, 2, 3$.

4. The reconstruction of ∂D in the acoustically soft case. In §3 we have observed that (2.8) can be interpreted as a nonlinear equation defining f implicitly as a function of θ and ϕ. Differentiating (2.8) with respect to θ and ϕ we have

$$\left[k_{i_1} \frac{\partial w_{1,i_1}}{\partial r} + \left(if + \frac{1}{k_{i_1}}\right) \frac{e^{-ik_{i_1} f}}{f^2}\right] \frac{\partial f}{\partial \theta} + \frac{\partial w_{1,i_1}}{\partial \theta} = 0, \quad 0 \le \theta \le \pi, \quad 0 \le \phi < 2\pi, \tag{4.1}$$

$$\left[k_{i_1}\frac{\partial w_{1,i_1}}{\partial r}+\left(if+\frac{1}{k_{i_1}}\right)\frac{e^{-ik_{i_1}f}}{f^2}\right]\frac{\partial f}{\partial \phi}+\frac{\partial w_{1,i_1}}{\partial \phi}=0,\quad 0\le\theta\le\pi,\quad 0\le\phi<2\pi. \tag{4.2}$$

Equations (4.1), (4.2) can be interpreted as ordinary differential equations for the unknown f. We note that, in the reconstruction procedure considered here, f is not approximated with a truncated series in spherical harmonics as in (3.7) and that the choice of several values of k (i.e., k_{i_1}, $i_1=1,2,\dots,N_1$) does not play any role. To obtain $f(\theta,\phi)$ we proceed as follows.

(i) For fixed θ and ϕ, let $\theta=\phi=0$; we solve (2.8) as a nonlinear equation. Let f_{00} be the solution found.

(ii) Given f_{00} we solve, for $0\le\theta\le\pi$, the differential equation obtained by taking the real part of (4.1) with the initial condition

$$f(0,0)=f_{00}. \tag{4.3}$$

Let $f_0(\theta,0)$ be the solution found. We verify that $f_0(\theta,0)$ satisfies (2.8).

(iii) Given $f_0(\theta,0)$ we solve, for $0\le\phi<2\pi$, the differential equation obtained by taking the real part of (4.2) with the initial condition

$$f(\theta,0)=f_0(\theta,0). \tag{4.4}$$

Finally we verify that the $f(\theta,\phi)$ obtained in this way satisfies (2.8).

The differential problems considered in (ii), (iii) are initial value problems for scalar differential equations and are solved numerically using a Runge–Kutta–Fehlberg method (i.e., the subroutine RKF45 [21]).

The problem considered in (iii) is performed only a finite number of times corresponding to a discretization of the variable θ.

We note that if the obstacle is cylindrically symmetric with respect to the z-axis (iii) is not necessary and the problem is solved after performing (i), (ii). Moreover the differential equations of (iii) for different values of θ are independent from one another and can be solved in parallel. The set of solutions of (2.8) in general contains the surface $f(\theta,\phi)$ as a subset so that when performing (i)–(iii) only trajectories that appear to define a closed surface should be considered. Finally the solution of the inverse problem based on the differential equations (4.1), (4.2) is limited to the acoustically soft obstacles and appears to be more sensitive to error in the data than the minimization procedure described in §3. However its computational cost is very small compared to the minimization procedures previously described.

5. The numerical experience. In this section we describe the numerical results obtained using the numerical methods described in §§3 and 4 on several test problems. When we consider the boundary condition (1.7) we choose $\chi=1$. To the numerically generated data $F_{i,i_1,j}=F_0(\underline{\hat{x}}_j,k_{i_1},\underline{\alpha}_i)$ is added a random error term, that is, $F_{i,i_1,j}$ is substituted with

$$F^{\star}_{i,i_1,j}=F_{i,i_1,j}+\epsilon\zeta\left|F_{i,i_1,j}\right|,$$

where $\epsilon\ge 0$ is a parameter and ζ is a random number uniformly distributed in $[-1,1]$. The surfaces ∂D considered are the following:

$$\left(\frac{2}{3}x\right)^2+\left(\frac{2}{3}y\right)^2+z^2=1 \qquad \text{oblate ellipsoid,} \tag{5.1}$$

$$x^2+y^2+\left(\frac{2}{3}z\right)^2=1 \qquad \text{prolate ellipsoid,} \tag{5.2}$$

$$\left(\left(\frac{2}{3}x\right)^2+\left(\frac{2}{3}y\right)^2\right)^5+z^{10}=1 \qquad \text{short cylinder,} \tag{5.3}$$

$$(x^2+y^2)^5+\left(\frac{2}{3}z\right)^{10}=1 \qquad \text{long cylinder,} \tag{5.4}$$

$$r=\frac{2}{3}\left(\cos^2\theta+\frac{1}{4}\sin^2\theta\right)^{1/2} \qquad \text{Vogel's peanut,} \tag{5.5}$$

$$r=1-\frac{1}{2}\cos 2\theta \qquad \text{horizontal platelet,} \tag{5.6}$$

$$r=1+\frac{1}{2}\cos 2\theta \qquad \text{vertical peanut,} \tag{5.7}$$

$$r=\frac{3}{5}\left(\frac{17}{4}+2\cos 3\theta\right)^{1/2} \qquad \text{pseudo-Apollo,} \tag{5.8}$$

$$r=\left(\left(\frac{\sin\theta}{H(\phi)}\right)^2+\left(\frac{2}{3}\cos\theta\right)^2\right)^{-1/2} \qquad \text{corrugated ellipsoid,} \tag{5.9}$$

$$r=\left(\left(\frac{\sin\theta}{h(\phi)}\right)^2+\left(\frac{2}{3}\cos\theta\right)^2\right)^{-1/2} \qquad \text{ellipsoid,} \tag{5.10}$$

$$r=\left(\left(\frac{2\sin\theta}{3H(\phi)}\right)^{10}+\cos^{10}\theta\right)^{-1/10} \qquad \text{corrugated cylinder,} \tag{5.11}$$

$$r=\frac{1}{2}+H(\phi)\sin^2\theta \qquad \text{corrugated platelet,} \tag{5.12}$$

where

$$h(\phi)=\left(\left(\frac{3}{4}\cos\phi\right)^2+\sin^2\phi\right)^{-1/2}, \tag{5.13}$$

$$H(\phi)=(R_h+A_h\cos 4\phi+B_h\cos 8\phi+C_h\cos 16\phi)^2, \tag{5.14}$$

and

$$A_h=\frac{0.3}{1.34}f_h;\quad B_h=\frac{0.05}{1.34}f_h;\quad C_h=-\frac{0.01}{1.34}f_h;\quad R_h=1-(A_h+B_h+C_h), \tag{5.15}$$

f_h is the corrugation parameter. In our numerical experience $f_h=0.2$. The obstacles D corresponding to (5.1)–(5.4) are convex bodies symmetric with respect to the z-axis and to the equator, the obstacles corresponding to (5.5)–(5.8) are nonconvex but they maintain the symmetry with respect to the z-axis. Finally the obstacles D corresponding to (5.9)–(5.12) in general are nonconvex and nonsymmetric with respect to the z-axis. We observe that a characteristic length L of the obstacles can be chosen to be one. Tables 5.1–5.4 show some numerical results obtained with the methods described in the previous sections. In the E_{L^2} and $E_{\max}$ columns of those tables we use the notation $\alpha(\beta)$ to mean $\alpha\cdot 10^{\beta}$. Table 5.1 summarizes the results obtained with the obstacles (5.1)–(5.4); Table 5.2 summarizes the results obtained with the obstacles (5.5)–(5.8); finally Table 5.3 summarizes the results obtained with the obstacles (5.9)–(5.12).

We remark that in the reconstructions presented in Table 5.3 the obstacles are considered as general surfaces, that is there is no use of symmetries in the reconstruction procedure.

TABLE 5.1
Axially symmetric convex obstacles: $L_{\max} = L_g = 8$; $\Omega_1 = A_1$; $\Omega_3 = A_2$.

Object	Recon-struction	Boundary condition	L_ρ	k_1	k_2	Reconstruction method	Penalization term North Pole	Penalization term Equator	*epsilon*	E_{L^2}
oblate ellipsoid	1	Dirichlet	–	3	–	RKF45	–	–	0.0	0.868(−4)
"	2	"	–	3	–	RKF45	–	–	0.05	0.703(−3)
"	3(*)	Neumann	4	2	no	DBCONF	no	no	0.0	0.546(−2)
"	4(*)	"	4	2	no	DBCONF	no	no	0.05	0.880(−2)
"	5	Mixed	4	2	no	DBCONF	no	no	0.0	0.365(−2)
"	6	"	4	2	no	DBCONF	no	no	0.05	0.589(−2)
prolate ellipsoid	7	Dirichlet	–	3	–	RKF45	–	–	0.0	0.256(−4)
"	8	"	–	3	–	RKF45	–	–	0.05	0.966(−2)
"	9	Neumann	4	2	no	DBCONF	no	no	0.0	0.172(−1)
"	10	"	4	2	no	DBCONF	no	no	0.05	0.157(−1)
"	11	Mixed	4	2	no	DBCONF	no	no	0.0	0.432(−2)
"	12	"	4	2	no	DBCONF	no	no	0.05	0.869(−2)
short cylinder	13	Dirichlet	–	3	–	RKF45	–	–	0.0	0.629(−2)
"	14	"	–	3	–	RKF45	–	–	0.05	0.147(−1)
"	15	Neumann	6	2	no	SIGMA	no	no	0.0	0.304(−1)
"	16	"	6	2	no	SIGMA	yes	yes	0.05	0.626(−1)
"	17(*)	Mixed	6	2	no	SIGMA	yes	yes	0.0	0.306(−1)
"	18(*)	"	6	2	no	SIGMA	yes	yes	0.05	0.468(−1)
long cylinder	19(*)	Dirichlet	–	3	–	RKF45	–	–	0.0	0.302(−2)
"	20(*)	"	–	3	–	RKF45	–	–	0.05	0.369(−2)
"	21	Neumann	6	2	3	SIGMA	yes	yes	0.0	0.543(−1)
"	22	"	6	2	3	SIGMA	yes	yes	0.05	0.482(−1)
"	23	Mixed	6	2	no	SIGMA	yes	yes	0.0	0.415(−1)
"	24	"	6	2	no	SIGMA	yes	yes	0.05	0.425(−1)

TABLE 5.2
Axially symmetric nonconvex obstacles: $L_{\max} = L_g = 8$; $\Omega_1 = A_1$; $\Omega_3 = A_2$.

Object	Recon-struction	Boundary condition	L_ρ	k_1	k_2	Reconstruction method	Penalization term North Pole	Penalization term Equator	*epsilon*	E_{L^2}
Vogel's peanut	1(*)	Dirichlet	4	3	no	SIGMA	no	no	0.0	0.128(−1)
"	2(*)	"	4	3	no	SIGMA	no	no	0.05	0.175(−1)
"	3	Neumann	4	2	no	SIGMA	no	no	0.0	0.596(−1)
"	4	"	4	2	no	SIGMA	no	no	0.05	0.176(−1)
"	5	Mixed	4	2	no	SIGMA	no	no	0.0	0.273(−1)
"	6	"	4	2	no	SIGMA	no	no	0.05	0.128(−1)
horizontal platelet	7	Dirichlet	4	3	no	SIGMA	no	no	0.0	0.451(−1)
"	8	"	4	3	no	SIGMA	no	no	0.05	failure
"	9	Neumann	4	2	no	SIGMA	no	no	0.0	0.103(+0)
"	10	"	4	2	no	SIGMA	no	no	0.05	0.202(+0)
"	11(*)	Mixed	4	2	no	SIGMA	no	no	0.0	0.861(−1)
"	12(*)	"	4	2	no	SIGMA	no	no	0.05	0.156(0)
vertical peanut	13	Dirichlet	4	3	no	SIGMA	yes	yes	0.0	0.172(−1)
"	14	"	4	3	no	SIGMA	yes	yes	0.05	failure
"	15	Neumann	4	2	3	SIGMA	no	no	0.0	0.725(−1)
"	16	"	4	2	3	SIGMA	no	no	0.05	failure
"	17	Mixed	4	2	3	SIGMA	no	no	0.0	0.649(−1)
"	18	"	4	2	3	SIGMA	no	no	0.05	failure
pseudo-Apollo	19	Dirichlet	4	3	no	SIGMA	no	no	0.0	0.602(−1)
"	20	"	4	3	no	SIGMA	no	no	0.05	0.114(+0)
"	21(*)	Neumann	4	2	3	SIGMA	yes	no	0.0	0.359(−1)
"	22(*)	"	4	2	3	SIGMA	yes	no	0.05	0.325(−1)
"	23	Mixed	4	2	3	SIGMA	yes	no	0.0	0.551(−1)
"	24	"	4	2	3	SIGMA	yes	no	0.05	0.321(−1)

TABLE 5.3

Generic obstacles: Reconstruction method = "DUNLSJ," *without penalization term*: $L_{\max} = L_g = 6$; $L_\rho = 4$; $k_1 = 2$; $\Omega_1 = A_3$; $\Omega_3 = A_3$.

Object	Reconstruction	Boundary condition	ϵ	$E_{\mathbf{L}^2}$
Corrugated Ellipsoid	1	Dirichlet	0.0	0.276(−1)
"	2	"	0.02	0.428(−1)
"	3(*)	Neumann	0.0	0.287(−1)
"	4(*)	"	0.02	0.417(−1)
"	5	Mixed	0.0	0.325(−1)
"	6	"	0.02	0.133(+0)
Ellipsoid	7	Dirichlet	0.0	0.291(−1)
"	8	"	0.02	0.296(−1)
"	9	Neumann	0.0	0.303(−1)
"	10	"	0.02	0.308(−1)
"	11	Mixed	0.0	0.152(+0)
"	12	"	0.02	0.245(−1)
Corrugated Cylinder	13	Dirichlet	0.0	0.534(−1)
"	14	"	0.02	0.652(−1)
"	15	Neumann	0.0	0.502(−1)
"	16	"	0.02	0.118(+0)
"	17(*)	Mixed	0.0	0.569(−1)
"	18(*)	"	0.02	0.675(−1)
Corrugated Platelet	19(*)	Dirichlet	0.0	0.663(−1)
"	20(*)	"	0.02	0.129(+0)
"	21	Neumann	0.0	0.105(+0)
"	22	"	0.02	failure
"	23	Mixed	0.0	0.654(−1)
"	24	"	0.02	0.175(+0)

TABLE 5.4

Performance as a function of ϵ. *Object: Long Cylinder*; *Boundary condition: Dirichlet.* $L_{\max} = L_g = 8$; $L_\rho = 6$; $k_1 = 3$; $\Omega_1 = A_1$; $\Omega_3 = A_3$. *Reconstruction method* = "SIGMA" *without penalization term.*

Reconstruction	ϵ	E_{max}	$E_{\mathbf{L}^2}$
1	0.0	0.611(−1)	0.286(−1)
2	0.05	0.614(−1)	0.294(−1)
3	0.10	0.587(−1)	0.298(−1)
4	0.20	0.525(−1)	0.398(−1)
5	0.30	0.865(−1)	0.548(−1)
6	0.40	0.109(+0)	0.655(−1)
7	0.50	0.123(+0)	0.714(−1)
8	0.60	0.131(+0)	0.752(−1)

In Figs. 5.1–5.9 the reconstructions denoted with (*) in Tables 5.1–5.3 are shown. Let $(\theta_i, \phi_j) = (i\frac{\pi}{36}, j\frac{\pi}{18})$, $i = 1, 2, \ldots, 35$, $j = 0, 1, \ldots, 35$ and let $f(\theta_i, \phi_j)$ be the exact values of the surfaces given by (5.1)–(5.12) and $f_c(\theta_i, \phi_j)$ be the corresponding values obtained by the reconstruction procedure of §§3 and 4. Let $e(\theta, \phi) = f(\theta, \phi) - f_c(\theta, \phi)$; in the Tables 5.1–5.4 we use as a performance index the relative $\mathbf{L}^2$ error, that is,

$$E_{\mathbf{L}^2} = \frac{\left[e^2(0,0) + e^2(\pi,0) + \sum_{i=1}^{35}\sum_{j=0}^{35} e^2(\theta_i,\phi_j)\right]^{1/2}}{\left[f^2(0,0) + f^2(\pi,0) + \sum_{i=1}^{35}\sum_{j=0}^{35} f^2(\theta_i,\phi_j)\right]^{1/2}} \tag{5.16}$$

or the relative $\mathbf{L}^\infty$ error, that is,

$$E_{\max} = \frac{\max\{|e(0,0)|, |e(\pi,0)|, |e(\theta_i,\phi_j)|\,,\ {}^{i=1,2,\ldots,35}_{j=0,1,\ldots,35}\}}{\max\{|f(0,0)|, |f(\pi,0)|, |f(\theta_i,\phi_j)|\,,\ {}^{i=1,2,\ldots,35}_{j=0,1,\ldots,35}\}}. \tag{5.17}$$

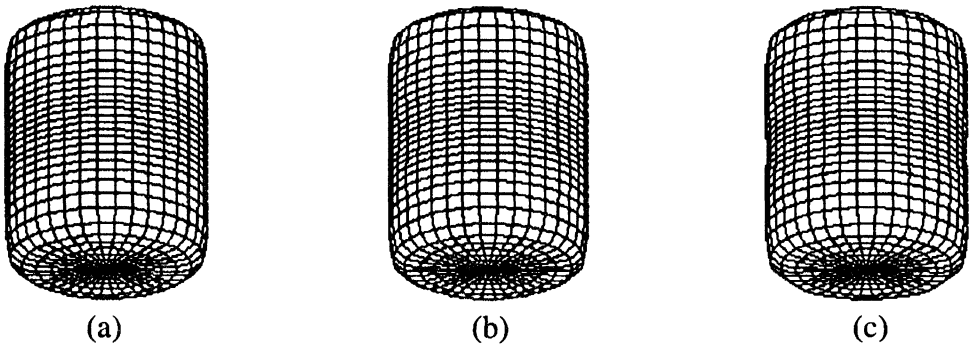


FIG. 5.1. (*long cylinder*). (a) *Original*; (b) *reconstruction* 19 *of Table* 5.1; (c) *reconstruction* 20 *of Table* 5.1.

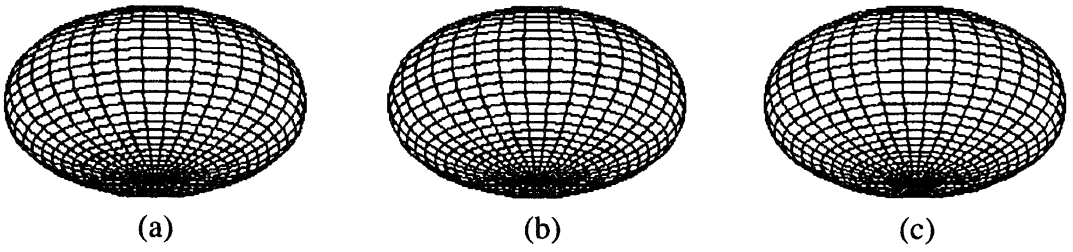


FIG. 5.2. (*oblate ellipsoid*). (a) *Original*; (b) *reconstruction* 3 *of Table* 5.1, (c) *reconstruction* 4 *of Table* 5.1.

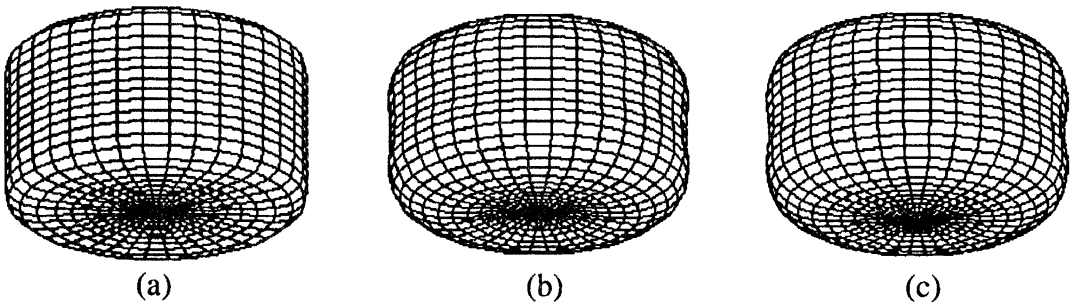


FIG. 5.3. (*short cylinder*). (a) *Original*; (b) *reconstruction* 17 *of Table* 5.1; (c) *reconstruction* 18 *of Table* 5.1.

Let us define the following sets:

$$A_1 = \left\{(\theta_i, 0) \mid \theta_i = i\frac{\pi}{10},\ i = 0, 1, \ldots, 10\right\}, \tag{5.18}$$

$$A_2 = \left\{(\theta_i, \phi_j) \mid \theta_i = i\frac{\pi}{10}\,, i = 1, \ldots, 9; \phi_j = j\frac{\pi}{4}\,,\ j = 0, 1, \ldots, 8\right\} \cup \{0, 0\} \cup \{\pi, 0\}, \tag{5.19}$$

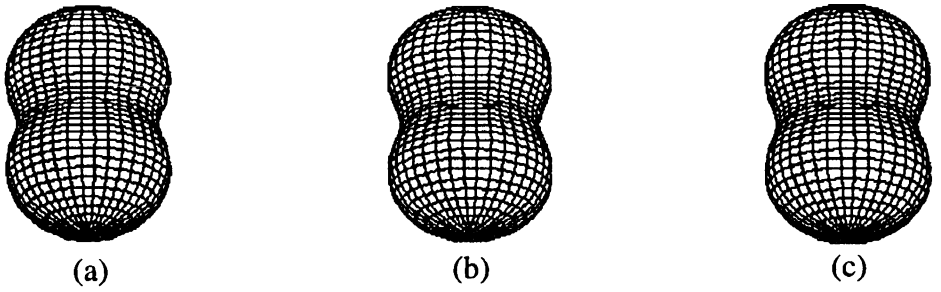


FIG. 5.4. (*Vogel's peanut*). (a) *Original*; (b) *reconstruction* 1 *of Table* 5.2; (c) *reconstruction* 2 *of Table* 5.2.

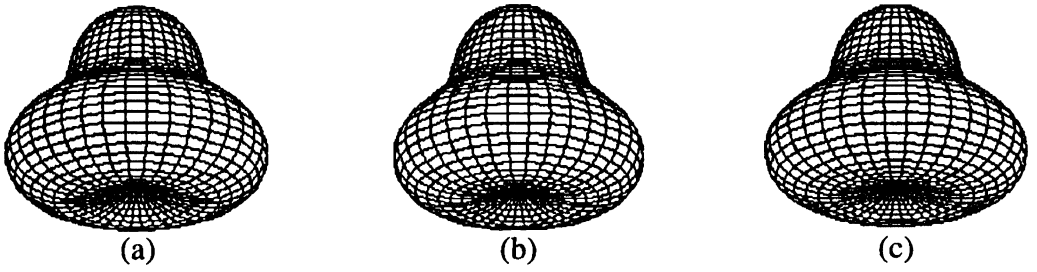


FIG. 5.5. (*pseudo-Apollo*). (a) *Original*; (b) *reconstruction* 21 *of Table* 5.2; (c) *reconstruction* 22 *of Table* 5.2.

$$A_3 = \left\{(\theta_i, \phi_j) \mid \theta_i = i\frac{\pi}{8}, i = 1, \ldots, 7; \phi_j = j\frac{\pi}{4}, j = 0, 1, \ldots, 6\right\} \cup \{0, 0\} \cup \{\pi, 0\}. \tag{5.20}$$

In our numerical experience we choose

$$\Omega_1 = A_1 \tag{5.21}$$

when the obstacles of Tables 5.1, 5.2 are considered or

$$\Omega_1 = A_3 \tag{5.22}$$

when the obstacles of Table 5.3 are considered. The set Ω_2 is either $\{2\}$, $\{3\}$, or $\{2, 3\}$ (see Tables 5.1–5.3). Finally when the obstacles of Tables 5.1, 5.2 are considered Ω_3 is given by A_2 and when the obstacles of Table 5.3 are considered Ω_3 is given by A_3. We observe that with these choices the resonance condition (1.9) is satisfied. The far field data corresponding to these choices are obtained by solving numerically the corresponding direct problems, that is, the boundary value problems (1.3), (1.4), (1.5), or (1.3), (1.4), (1.6), or (1.3), (1.4), (1.7) using a T-matrix approach [16].

In our numerical experience we have $L_{\max} = L_g = 6$ or 8, $L_\rho = 4$ or 6 (see Tables 5.1–5.3). Finally in Table 5.4 we show the performance of our algorithms for increasing values of ϵ in the case of the acoustically soft long cylinder. The method based on the global minimization algorithm SIGMA appears to be the most powerful one at the price of higher computational cost. The computations previously described have been performed on a VAX 6310 with VMS operating system.

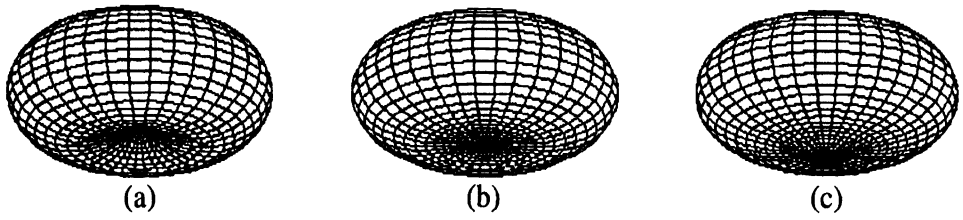


FIG. 5.6. (*horizontal platelet*). (a) *Original*; (b) *reconstruction* 11 *of Table* 5.2; (c) *reconstruction* 12 *of Table* 5.2.

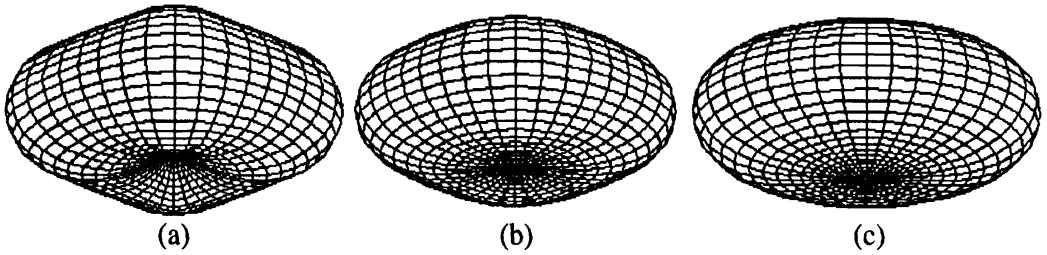


FIG. 5.7. (*corrugated platelet*). (a) *Original*; (b) *reconstruction* 19 *of Table* 5.3; (c) *reconstruction* 20 *of Table* 5.3.

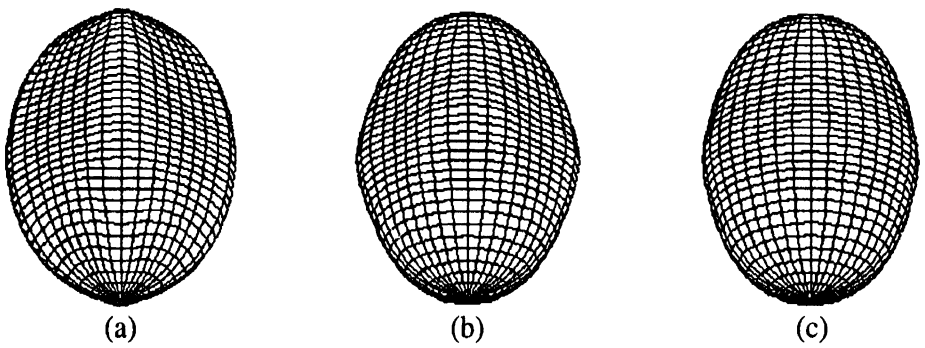


FIG. 5.8. (*corrugated ellipsoid*). (a) *Original*; (b) *reconstruction* 3 *of Table* 5.3; (c) *reconstruction* 4 *of Table* 5.3.

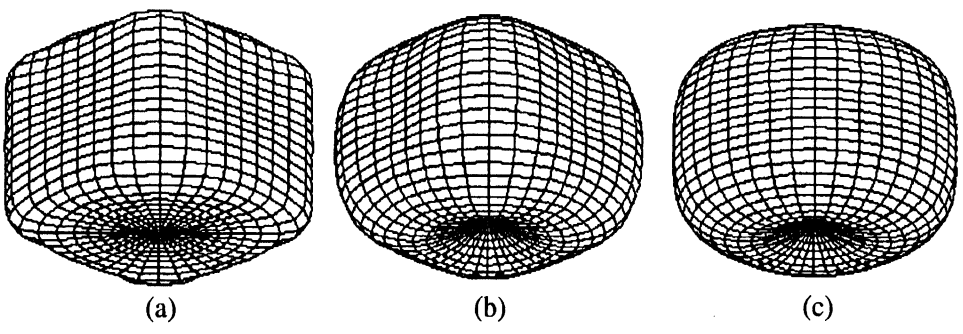


FIG. 5.9. (*corrugated cylinder*). (a) *Original*; (b) *reconstruction* 17 *of Table* 5.3; (c) *reconstruction* 18 *of Table* 5.3.

REFERENCES

[1] D. COLTON AND R. KRESS, *Integral Equation Methods in Scattering Theory*, John Wiley, New York, 1983.

[2] R. KRESS, *Inverse acoustic obstacle scattering*, in Mathematical and Numerical Aspects of Wave Propagation Phenomena, G. Cohen, L. Halpern, and P. Joly, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 487–497.

[3] A. ROGER, *Newton–Kantorovich algorithm applied to an electromagnetic inverse problem*, IEEE Trans. Ant. Prop., AP-29 (1981), pp. 232–238.

[4] R. D. MURCH, D. G. H. TAN, AND D. J. N. WALL, *Newton–Kantorovich method applied to two dimensional inverse scattering for an exterior Helmholtz problem*, Inverse Problems, 4 (1988), pp. 1117–1128.

[5] S. L. WANG AND Y. M. CHEN, *An efficient numerical method for exterior and interior inverse problems for the Helmholtz equation*, Wave Motion, 13 (1991), pp. 387–399.

[6] T. S. ANGELL, D. COLTON, AND A. KIRSCH, *The three dimensional inverse scattering problem for acoustic waves*, J. Differential Equations, 46 (1982), pp. 46–58.

[7] G. KRISTENSSON AND C. R. VOGEL, *Inverse problems for acoustic waves using the penalised likelihood method*, Inverse Problems, 2 (1986), pp. 461–479.

[8] A. KIRSCH AND R. KRESS, *On an integral equation of the first kind in inverse acoustic scattering*, in Inverse Problem, U. Hornung and J. R. Cannon, eds., Internat. Ser. Numer. Math., Vol. 77, Birkhauser, Boston, MA, 1986, pp. 93–102.

[9] ———, *A numerical method for an inverse scattering problem*, in Inverse Problems, Engl, Groetsch Editors, Academic Press, New York, 1987, pp. 279–290.

[10] ———, *An optimization method in inverse acoustic scattering*, in Boundary Elements IX, Vol.. 3, Fluid Flow and Potential Applications, C. A. Brebbia, W. L. Wendland, and G. Kuhn, eds., Springer-Verlag, Heidelberg, 1987, pp. 3–18.

[11] D. COLTON AND P. MONK, *The numerical solution of the three dimensional inverse scattering problem for time harmonic acoustic waves*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 278–291.

[12] F. ALUFFI-PENTINI, E. CAGLIOTI, L. MISICI, AND F. ZIRILLI, *A parallel algorithm for a three dimensional inverse acoustic scattering problem*, in Parallel Computing: Methods, Algorithms and Applications, D. J. Evans and C. Sutti, eds., I.O.P. Publishing, Bristol, 1989, pp. 193–200.

[13] ———, *A numerical method for the three dimensional inverse acoustic scattering problem with incomplete data*, in Advances in Numerical Partial Differential Equations and Optimization, S. Gomez, J. P. Hennart, and R. A. Tapia, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 273–283.

[14] L. MISICI AND F. ZIRILLI, *An inverse problem for the three dimensional Helmholtz equation with Neumann or mixed boundary condition: A numerical method*, in Mathematical and numerical aspects of wave propagation phenomena, G. Cohen, L. Halpern, and P. Joly, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1991, pp. 497–508.

[15] P. MAPONI, L. MISICI, AND F. ZIRILLI, *An inverse problem for the three dimensional vector Helmholtz equation for a perfectly conducting obstacle*, in Comput. Math. Appl., 22 (1991), pp. 137–146.

[16] P. C. WATERMAN, *New formulation of acoustic scattering*, J. Acoust. Soc. Amer., 45 (1988), pp. 1417–1429.

[17] Subroutine DBCONF of the IMSL Mathematical Software Library, 1987.

[18] F. ALUFFI-PENTINI, V. PARISI, AND F. ZIRILLI, *A global optimization algorithm using stochastic differential equations*, ACM Trans. Math. Software, 14 (1988), pp. 345–365.

[19] ———, *Algorithm 667 SIGMA: A stochastic integration global minimization algorithm*, ACM Trans. Math. Software, 14 (1988), pp. 366–380.

[20] Subroutine DUNLSJ of the IMSL Mathematical Software Library, 1987.

[21] G. FORSYTE, M. MALCOM, AND C. MOLER, *Computer methods for mathematical computations*, Prentice Hall Inc., Englewood Cliffs, NJ, 1977, pp. 129–147.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1190–1200, September 1994

013

SPARSE PRECONDITIONED ITERATIVE METHODS FOR DENSE LINEAR SYSTEMS*

YI YAN†

Abstract. Two sparse preconditioned iterative methods are presented to solve dense linear systems arising in the solution of two-dimensional boundary integral equations. In the first method, the sparse preconditioner is constructed simply by choosing a small block of elements in the coefficient matrix of a dense linear system. The two-grid method falls into this category when the dense linear system arises from the Nyström method for a second kind boundary integral equation. In the second method, the sparse preconditioner is obtained through condensation of the coefficient matrix by discrete Fourier transforms, which can be implemented efficiently using fast Fourier transforms. Both iterative methods involve only $O(N^2)$ arithmetic operations per iteration and converge rapidly when the dense linear systems arise from quadrature methods for boundary integral equations arising in two-dimensional problems. The author's numerical experiments demonstrate the computational efficiency of each method.

Key words. dense linear systems, sparse preconditioners

AMS subject classifications. 65F10, 65R20

1. Introduction.

We present two algebraic iterative methods for solving the dense linear system $\boldsymbol{L}\boldsymbol{w} = \boldsymbol{b}$, where $\boldsymbol{L}$ is an $N \times N$ nonsingular, nonsymmetric, and dense matrix, and $\boldsymbol{b}$ is an $N \times 1$ vector. Such a linear system arises in boundary integral equation methods (BIEMs) for solving problems from two-dimensional potential flows, acoustics, electromagnetic waves, and elasticity. For these two-dimensional problems, the size of the linear system may be moderate, but many such systems may need to be solved. Among the examples are multifrequency analysis using BIEMs in acoustics and electromagnetic applications and time-dependent problems solved in the frequency domain by BIEMs. Hence, iterative methods are usually required for its efficient solution. The development of iterative methods has focused on the solution of the typical dense linear system

$$(\boldsymbol{I} + \boldsymbol{A})\boldsymbol{w} = \boldsymbol{b} \tag{1.1}$$

arising in second kind boundary integral equations, where both matrices $\boldsymbol{I} + \boldsymbol{A}$ and $(\boldsymbol{I} + \boldsymbol{A})^{-1}$ are bounded by constants independent of N under the induced L^2-norm (see, for example, [3] and [13]). Among the iterative methods are two-grid methods (see Brakhage [1], Atkinson [2], [3], Hackbusch [9], [10], Atkinson and Graham [5], Atkinson [6]) and multigrid methods (see Hackbusch [9], [10], Hemker and Schippers [12], Schippers [21], and Mandel [14]). The two-grid and multigrid methods take advantage of the properties that an integral equation is approximated by a family of linear systems (1.1), and that information from different grid levels can be communicated. These properties enable the linear system (1.1) to be solved at a lower level iteratively. Other iterative methods include the Krylov-space-based methods such as conjugate gradient to the normal equations (CGN), generalized minimum residual (GMRES) and conjugate gradient square (CGS) methods, which have been combined with the fast multipole method for the rapid calculation of the matrix vector products; see, for example, Rokhlin [20] and Greenbaum, Greengard, and Mayo [8]. A comparison of CGN, GMRES, and CGS methods can be found in [16]. Recently some preconditioners have been proposed (see Vavasis [22] and Nabors, Korsmeyer, and White [15]) for the linear system (1.1) arising from three-dimensional problems.

*Received by the editors May 18, 1992; accepted for publication (in revised form) September 1, 1993. This work was supported in part by National Science Foundation grant RII-8610671 and the Commonwealth of Kentucky through the University of Kentucky Center for Computational Sciences.

†Department of Mathematics, University of Kentucky, Lexington, Kentucky 40506 (yan@ukcc.uky.edu).

Another typical dense linear system is from singular boundary integral equations with logarithmic, Cauchy, or hypersingular kernels and has the form (see, for example, [19], [4], [7])

$$(\boldsymbol{C}+\boldsymbol{B})\boldsymbol{w}=\boldsymbol{b}, \tag{1.2}$$

where $\boldsymbol{C}$ is an invertible circulant matrix, and $\boldsymbol{B}$ is a $N \times N$ dense matrix with both matrices $\boldsymbol{I}+\boldsymbol{C}^{-1}\boldsymbol{B}$ and $(\boldsymbol{I}+\boldsymbol{C}^{-1}\boldsymbol{B})^{-1}$ bounded by constants independent of N under the induced L^2-norm (see, for example, [19]). Iterative methods for this linear system (1.2) have also been studied (see Reichel [17], Hebeker [11], and Graham and Yan [7]), but have received much less attention.

In the iterative methods of this paper, the most difficult task is to find appropriate preconditioners. These preconditioners are quite different from those proposed in [15] and [22] and are constructed by using information only from the linear systems (1.1) or (1.2) themselves. This idea is different from those in the two-grid and multigrid methods, where preconditioners are built by employing information of a linear system at a low level. When the preconditioners are found, we introduce preconditioned stationary Richardson iterative methods. Our major focus in this paper is on the description of these iterative methods. Their convergence analysis and error estimates will not be provided, except for a necessary citation of some references.

A block-preconditioned Richardson iteration (BPRI) method is presented for the linear system (1.1). It is equivalent to a two-grid method when (1.1) arises from the Nyström method for second kind boundary itegral equations. Thus the BPRI method provides a new way to look at the two-grid method. Its convergence in this special case follows from an analysis for a two-grid method given in [2].

The block-preconditioned iterative method cannot be applied directly to the linear system (1.2) when $\boldsymbol{C}$ is not the identity matrix. Instead, a Fourier-condensation-preconditioned Richardson iteration (FCPRI) method is introduced. Since (1.1) can be viewed as a special case of (1.2), the FCPRI method also applies to (1.1). Its convergence was recently analyzed by Yan [23] and Reichel and Yan [19] for systems (1.1) and (1.2) arising from quadrature methods for boundary integral equations. A similar preconditioned iterative method was introduced in [18] for solving some Cauchy singular integral equations of the second kind.

Both BPRI and FCPRI methods have sparse preconditioner matrices and involve only $O(N^2)$ arithmetic operations per iteration. They are applied to boundary integral equations arising in some exterior boundary value problems and various numerical results are presented that demonstrate their computational efficiency.

We remark that the $O(N^2)$ arithmetic operation can be further reduced to $O(N)$ by applying the fast multipole method used in [8] and [20]. A justification of this reduction in arithmetic operation is beyond the scope of this paper.

2. BPRI method. Assume that N is an even integer, and let

$$\boldsymbol{A}=[a_{j,k}], \quad -\frac{N}{2}<j, \quad k\le\frac{N}{2}, \tag{2.1}$$

which can be partitioned by columns as

$$\boldsymbol{A}=\left(\boldsymbol{a}_{-N/2+1},\ldots,\boldsymbol{a}_{N/2}\right). \tag{2.2}$$

In addition, we assume that $N=\eta M$ with $\eta\in\mathbf{Z}^+$ and M is an even integer. Let

$$\boldsymbol{a}'_k=\begin{cases}\boldsymbol{a}_k, & k=\eta l, -\frac{M}{2}<l\le\frac{M}{2},\\ \mathbf{o}, & \text{otherwise},\end{cases}$$

and define

$$A' = (a_{-N/2+1}, \ldots, a'_{N/2}). \tag{2.3}$$

The matrix A' is a matrix whose ηl column, $l = -\frac{M}{2} + 1, \ldots, \frac{M}{2}$, is the ηl column of A, and whose other columns are zeros. Thus A' is constructed by a block of columns of A and is sparse when η is relatively large.

The dense linear system (1.1) can be written as

$$(I + \eta A')w = (\eta A' - A)w + b. \tag{2.4}$$

Generally, we do not expect $\|\eta A' - A\|$ to be small, where $\|\cdot\|$ denotes the L^{-2}-norm, because $\eta A'$ is sparse. Equation (2.4) is then not appropriate for a Richardson iteration. It is known that the integral operator form of the matrix A usually has the property of compactness. This property allows us to expect $\|(\eta A' - A)A\|$ to be small. Therefore, we use the relation $w = b - Aw$ to rewrite (2.4) as

$$(I + \eta A')w = (\eta A' - A)(b - Aw) + b. \tag{2.5}$$

Based on this equation, we introduce the following stationary Richardson iteration method

$$(I + \eta A')w^{(m+1)} = (\eta A' - A)(b - Aw^{(m)}) + b, \qquad m = 0, \ldots \tag{2.6}$$

in which $(I + \eta A')$ is used as its preconditioner. This preconditioner is sparse because A' is sparse. Combining (2.5) and (2.6) we obtain

$$(I + \eta A')(w^{(m+1)} - w) = -(\eta A' - A)A(w^{(m)} - w), \qquad m = 0, \ldots,$$

which obviously leads to the convergence of this iterative method when the condition

$$\|(I + \eta A')^{-1}(\eta A' - A)A\| \leq \epsilon < 1$$

is satisfied.

At each step of the iterative method, a sparse linear system of the form

$$(I + \eta A')v = g \tag{2.7}$$

must be solved. We now show that this requires only the solution of an $M \times M$ linear system.

Let

$$a^*_{p,q} = a_{\eta p, \eta q}, \qquad -\frac{M}{2} < p, \quad q \leq \frac{M}{2},$$

and define

$$A_M = \left[a^*_{p,q}\right].$$

The matrix A_M is a submatrix of the matrix A.

We introduce a restriction operator R_M from $\mathbf{C}^N$ to $\mathbf{C}^M$ by

$$R_M u = [u_\eta, u_{\eta 2}, \ldots, u_{\eta M}]^T, \quad \text{for } u = [u_1, \ldots, u_N]^T$$

and a prolongation operator P_M from $\mathbf{C}^M$ to $\mathbf{C}^N$ by

$$P_M v = [v'_1, \ldots, v'_N]^T, \quad \text{for } v = [v_1, \ldots, v_M]^T$$

with

$$v'_k = \begin{cases} v_l, & k = \eta l, \\ 0, & \text{otherwise.} \end{cases}$$

With this notation, it can be proved that (2.7) is equivalent to

$$\boldsymbol{v}^* + \eta \boldsymbol{A}_M \boldsymbol{v}^* = \boldsymbol{R}_M \boldsymbol{g}, \tag{2.8}$$

$$\boldsymbol{v} = -\eta \boldsymbol{A}' \boldsymbol{P}_M \boldsymbol{v}^* + \boldsymbol{g}, \tag{2.9}$$

where $\boldsymbol{v}^* \in \boldsymbol{C}^M$. This shows that the solution of the sparse linear system (2.7) can be determined in only $(N-M)M$ operations by solving the $M \times M$ system (2.8) followed by a matrix-vector multiplication.

Let the residual be $\boldsymbol{r}^{(m)} = \boldsymbol{b} - (\boldsymbol{I} + \boldsymbol{A})\boldsymbol{w}^{(m)}$. The iterative method (2.6) can be written as

$$\boldsymbol{w}^{(m+1)} = \boldsymbol{w}^{(m)} + \boldsymbol{r}^{(m)} - (\boldsymbol{I} + \eta \boldsymbol{A}')^{-1} \boldsymbol{A} \boldsymbol{r}^{(m)}. \tag{2.10}$$

This form of the iterative method is far superior to the form (2.6) because it requires fewer arithmetic operations involving matrices and vectors. An algorithm for this method is given as follows.

ALGORITHM 1

`Input:`	$\boldsymbol{w}^{(m)}$.
`Compute the residual:`	$\boldsymbol{r}^{(m)} = \boldsymbol{b} - (\boldsymbol{I} + \boldsymbol{A})\boldsymbol{w}^{(m)}$.
`Smoothing step:`	$\boldsymbol{A}\boldsymbol{r}^{(m)}$.
`Preconditioner correction:`	$\boldsymbol{\delta}^{(m)} = (\boldsymbol{I} + \eta \boldsymbol{A}')^{-1} \boldsymbol{A} \boldsymbol{r}^{(m)}$.
`Output:`	$\boldsymbol{w}^{(m+1)} = \boldsymbol{w}^{(m)} + \boldsymbol{r}^{(m)} - \boldsymbol{\delta}^{(m)}$.

The total number of arithmetic operations for each step of the iteration is approximately

$$2N^2 + \frac{2}{3}M^3 + (N - M)M.$$

In the case where the linear system (1.1) arises from the Nyström method, this algorithm is an equivalent form of the algorithm given by Atkinson and Graham (see (1.17) of [5]). The algorithm of this algebraic form is more transparent for computing since neither approximate integral operators nor a matrix from a low level are involved. We emphasize that Algorithm 1 is valid based on the assumption that (1.1) arises in second kind boundary integral equations on smooth boundaries. When a boundary with corners is considered, it is shown in [5] that some special treatment near corners is necessary for satisfactory performance of the two-grid method. It may be possible to modify the preconditioner in Algorithm 1 and preserve its sparsity structure to cope with the special modification around corners. Moreover, it may be possible to apply Algorithm 1 to (1.1) arising in three-dimensional problems.

3. FCPRI method. Now consider the solution of (1.2), and let

$$\boldsymbol{B} = [b_{j,k}], \quad -\frac{N}{2} < j, \quad k \le \frac{N}{2}.$$

We introduce a unitary matrix $\boldsymbol{F}$ defined by

$$\boldsymbol{F} = N^{-1/2}[w^{jk}], \quad -\frac{N}{2} < j, \quad k \le \frac{N}{2},$$

where $\omega = e^{-ih}$ with $h = \frac{2\pi}{N}$. This matrix is the matrix of the discrete Fourier transform.

If we apply the discrete Fourier transform to the linear system (1.2), and let $\boldsymbol{\phi} = \boldsymbol{N}^{-1/2}\boldsymbol{F}^*\boldsymbol{w}, \boldsymbol{g} = \boldsymbol{N}^{-1/2}\boldsymbol{F}^*\boldsymbol{b}$, and $\boldsymbol{B}_f = \boldsymbol{F}^*\boldsymbol{B}\boldsymbol{F}$, then we obtain an equivalent linear system

$$(\boldsymbol{D} + \boldsymbol{B}_f)\boldsymbol{\phi} = \boldsymbol{g}, \tag{3.1}$$

where $\boldsymbol{D} = \boldsymbol{F}^*\boldsymbol{C}\boldsymbol{F} = \operatorname{diag}(\delta_{-N/2+1}, \ldots, \delta_{N/2})$, and δ_j are the eigenvalues of the circulant matrix $\boldsymbol{C}$.

The matrix $\boldsymbol{B}_f$ has a nice property, namely, if the elements of the matrix $\boldsymbol{B}$ are

$$b_{j,k} = hb(t_j, t_k), \quad t_j = hj, \quad t_k = hk,$$

where $b(s, \sigma)$ is a given function, then the elements $b_f(j, k)$ of $\boldsymbol{B}_f$ are the approximate Fourier coefficients of the function $b(s, \sigma)$. In the boundary integral equations arising in two-dimensional problems, the function $b(s, \sigma)$ is usually periodic in each variable. Thus these Fourier coefficients are usually small when either j or k becomes large. Therefore $\boldsymbol{B}_f$ can be approximated by a sparse matrix $\widetilde{\boldsymbol{B}}_f$ in the block matrix form

$$\widetilde{\boldsymbol{B}}_f = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B}_{f'} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix},$$

where $\boldsymbol{B}_{f'}$ is an $M \times M$ matrix described as follows.

As in the BPRI method, we employ the elements $b_{\eta p, \eta q}$ of the matrix $\boldsymbol{B}$. These elements form a submatrix $\boldsymbol{B}_M$ given by

$$\boldsymbol{B}_M = [b^*_{p,q}]$$

with

$$b^*_{p,q} = b_{\eta p, \eta q}, \quad -\frac{M}{2} < p, \quad q \le \frac{M}{2}.$$

Let $\omega_M = e^{-ih'}$ with $h' = \frac{2\pi}{M}$, and define the $M \times M$ unitary matrix $\boldsymbol{F}_M$ by

$$\boldsymbol{F}_M = M^{-1/2}[\omega_M^{p,q}], \quad -\frac{M}{2} < p, \quad q \le \frac{M}{2}.$$

This is a lower-dimensional matrix of the discrete Fourier transform. The matrix $\boldsymbol{B}_{f'}$ is then defined as

$$\boldsymbol{B}_{f'} = \boldsymbol{F}_M^*\boldsymbol{b}_M\boldsymbol{F}_M = [b_{f'}(p, q)].$$

Consequently, the matrix $\widetilde{\boldsymbol{B}}_f = [\tilde{b}_f(j, k)]$ is defined by

$$\tilde{b}_f(j, k) = \begin{cases} b_{f'}(j, k), & \text{if } -\frac{M}{2} < j, \quad k \le \frac{M}{2}, \\ 0, & \text{otherwise.} \end{cases}$$

We view $\widetilde{\boldsymbol{B}}_f$ as a condensed matrix of $\boldsymbol{B}$ by the discrete Fourier transform.

Now choosing the preconditioner as $\boldsymbol{D} + \widetilde{\boldsymbol{B}}_f$, we introduce the stationary Richardson iteration method

$$(\boldsymbol{D} + \widetilde{\boldsymbol{B}}_f)\boldsymbol{\phi}^{(m+1)} = (\widetilde{\boldsymbol{B}}_f - \boldsymbol{B}_f)\boldsymbol{\phi}^{(m)} + \boldsymbol{g}, \qquad m = 0, 1, \ldots. \tag{3.2}$$

Combining (3.1) and (3.2) we obtain

$$(\boldsymbol{D}+\widetilde{\boldsymbol{B}}_f)(\boldsymbol{\phi}^{(m+1)}-\boldsymbol{\phi})=(\widetilde{\boldsymbol{B}}_f-\boldsymbol{B}_f)(\boldsymbol{\phi}^{(m)}-\boldsymbol{\phi}),\qquad m=0,1,\ldots.$$

Hence, this iterative method is convergent when the condition

$$\|(\boldsymbol{D}+\widetilde{\boldsymbol{B}}_f)^{-1}(\widetilde{\boldsymbol{B}}_f-\boldsymbol{B}_f)\|\le\epsilon<1$$

is satisfied.

At each step of the iteration, a sparse linear system of the form

$$(\boldsymbol{D}+\widetilde{\boldsymbol{B}}_f)\boldsymbol{v}=\rho \tag{3.3}$$

must be solved. This sparse linear system only requires the solution of an $M\times M$ linear system, which is explained as follows.

Let $\boldsymbol{v}_M=[v_{-M/2+1},\ldots,v_{M/2}]^T$, $\rho_M=[\rho_{-M/2+1},\ldots,\rho_{M/2}]^T$, and $\boldsymbol{D}_M=\operatorname{diag}(\delta_{-M/2+1},\ldots,\delta_{M/2})$. Equation (3.3) is then equivalent to

$$(\boldsymbol{D}_M+\boldsymbol{B}_{f'})\boldsymbol{v}_M=\boldsymbol{\rho}_M, \tag{3.4}$$

$$\delta_j v_j=\rho_j\quad -\frac{N}{2}<j\le-\frac{M}{2},\quad\text{and}\quad\frac{M}{2}<j\le\frac{N}{2}. \tag{3.5}$$

Equation (3.4) is clearly an $M\times M$ linear system, while the solution of (3.5) is explicit.

Define the residual by $\boldsymbol{r}^{(m)}=\boldsymbol{g}-(\boldsymbol{D}+\boldsymbol{B}_f)\boldsymbol{\phi}^{(m)}$. Equation (3.2) can be rewritten as

$$\boldsymbol{\phi}^{(m+1)}=\boldsymbol{\phi}^{(m)}+(\boldsymbol{D}+\widetilde{\boldsymbol{B}}_f)^{-1}\boldsymbol{r}^{(m)}. \tag{3.6}$$

An algorithm for this method is given as follows.

ALGORITHM 2

(I) Matrix generation:

1. Compute $\boldsymbol{D}=\boldsymbol{F}^*\boldsymbol{C}\boldsymbol{F}$ using FFT.
2. Compute $\boldsymbol{g}=N^{-1/2}\boldsymbol{F}^*\boldsymbol{b}$ using FFT.
3. Compute $\boldsymbol{B}_{f'}=\boldsymbol{F}_M^*\boldsymbol{B}_M\boldsymbol{F}_M$ using 2-d FFT.

(II) Iteration:

Input: $\boldsymbol{\phi}^{(m)}$.

Compute the residual: $\boldsymbol{r}^{(m)}=\boldsymbol{g}-\boldsymbol{D}\boldsymbol{\phi}^{(m)}-\boldsymbol{F}^*[\boldsymbol{B}(\boldsymbol{F}\boldsymbol{\phi}^{(m)})]$.

Preconditioner correction: $\boldsymbol{\delta}^{(m)}=(\boldsymbol{D}+\widetilde{\boldsymbol{B}}_f)^{-1}\boldsymbol{r}^{(m)}$.

Output: $\boldsymbol{\phi}^{(m+1)}=\boldsymbol{\phi}^{(m)}+\boldsymbol{\delta}^{(m)}$.

(III) Final output:

Compute $\boldsymbol{w}^{(m)}=N^{1/2}\boldsymbol{F}\boldsymbol{\phi}^{(m)}$ using FFT.

The final step is conducted only when a stopping criterion for the iteration is satisfied. The operation count for each step of the iteration is approximately

$$N^2+\frac{2}{3}M^3+2N\log N+N+(N-M).$$

For both the BPRI and FCPRI methods, M should be chosen within $O(N^{2/3})$ to minimize operation counts in the iterations. Since (1.1) is a special case of (1.2), both the BPRI and

FCPRI methods can be applied. With the choice of $M \approx cN^{2/3}$, the FCPRI method obviously requires less arithmetic operations than the BPRI method does in solving (1.1).

Algorithm 2 is designed for (1.2) arising in singular boundary integral equations on a smooth boundary. When a boundary with corners is considered, the algorithm may lead to poor performance. Also, unlike the BPRI method, Algorithm 2 cannot be applied to three-dimensional problems.

4. Applications and numerical results. Consider the exterior Neumann boundary value problem

$$\begin{aligned} \Delta u &= 0 \quad \text{for } x \in \boldsymbol{R}^2 \backslash \Omega, \\ \frac{\partial u}{\partial n} &= f \quad \text{for } x \in \partial\Omega, \\ u(x) &= c \log |x| + o(1), \qquad |x| \to \infty. \end{aligned}$$

Using its single layer potential representation, this problem can be reformulated as the solution of the second kind boundary integral equation

$$w(s) + \int_{-\pi}^{\pi} w(\sigma) b(s, \sigma) d\sigma = -2 f(v(s)) |v'(s)|, \qquad x \in [-\pi, \pi], \tag{4.1}$$

where n denotes the unit normal vector to the boundary $\partial\Omega$ directed into the exterior of Ω, and

$$b(s, \sigma) = \frac{1}{\pi} \frac{(v(s) - v(\sigma)) \cdot n(v(s)) |v'(s)|}{|v(s) - v(\sigma)|^2}$$

with $v(s)$ a regular 2π-periodic parametric representation of the boundary $\partial\Omega$ of the form

$$v(s) = (v_1(s), v_2(s)), \qquad s \in [-\pi, \pi].$$

We approximate (4.1) by the quadrature method

$$w_j + h \sum_{k=-N/2+1}^{N/2} b(t_j, t_k) w_k = -2 f(v(t_j)) |v'(t_j)|, \qquad j = -\frac{N}{2} + 1, \ldots, \frac{N}{2} \tag{4.2}$$

for the approximate values w_j of $w(t_j)$, where $h = \frac{2\pi}{N}$ and $t_j = jh$. This quadrature method has an exponential rate of convergence when $v(s)$ and $f(v(s))$ are smooth functions (see, for example, [13], [19], [23]).

In applications of the BPRI and FCPRI methods to the linear system (4.2), we choose $M = [N^{2/3}]$, which is roughly the largest M, to ensure that the total operation count is $O(N^2)$. We iterate until the relative correction satisfies

$$\frac{\|\boldsymbol{w}^{(m+1)} - \boldsymbol{w}^{(m)}\|}{\|\boldsymbol{w}^{(m+1)}\|} \le \epsilon, \tag{4.3}$$

where $\|\cdot\|$ denotes the Euclidean vector norm. In our test example, we choose the real number ϵ by

$$\epsilon = \max\{10^{-N/8}, \delta\},$$

where the factor $10^{-N/8}$ is chosen to avoid performing unnecessary iterations when N is small and where δ is chosen close to a required precision. We choose $\delta = 10^{-15}$ or 10^{-12}

to retain and observe the exponential rate of convergence of the quadrature method. We report the errors in the $\|\cdot\|$ norm when the solution $w(s)$ is known. For this purpose, we let $r_h w = [w(t_{-N/2+1}), \ldots, w(t_{N/2})]^T$. The column IT in Tables 1–5 gives the smallest value of m for which (4.3) is satisfied. The CPU time with respect to the values of N are also reported. For comparison to known iterative methods, some results using the unpreconditioned biconjugate gradient method (BCG) are also presented. All calculations are performed in double precision on the University of Kentucky's IBM 3090-600J.

Our first example is for an elliptic boundary $\nu(s) = (\cos(s), 0.25\sin(s))$ and for the right-hand side of (4.1) equal to $1.6\cos(s)$. The solution of this equation is $w(s) = \cos(s)$. The numerical results are reported in Table 1. Since $w(s)$ is an eigenfunction of the integral equation, this example is trivial for the BCG method, and it is not necessary to present the results using BCG method.

TABLE 1
$\epsilon = \max\{10^{-N/8}, 10^{-15}\}$.

M	N	BPRI method			FCPRI method		
		$\|w^{(IT)} - r_h w\|$	IT	CPU(sec)	$\|w^{(IT)} - r_h w\|$	IT	CPU(sec)
4	8	6.15E–3	2	0.01	5.35E–3	2	0.02
4	16	1.06E–3	3	0.01	2.14E–3	3	0.02
8	32	4.13E–6	3	0.06	8.81E–6	3	0.05
16	64	2.12E–11	3	0.22	4.51E–11	3	0.17
16	128	3.97E–16	5	0.95	4.77E–16	5	0.62
32	256	8.14E–16	3	3.03	8.59E–16	3	1.94
64	512	1.70E–15	2	11.11	1.50E–15	2	6.99

The results in Table 1 show that both methods retain the exponential rate of convergence of the quadrature method in a small number of iterations, and that the FCPRI method is faster than the BPRI method when N becomes sufficiently large, which confirms the estimate of the operation counts.

The second example is for the same boundary but with the right-hand side of (4.1) replaced by a nonsmooth function $|\sin(s)|$. The numerical results are reported in Table 2.

TABLE 2
$\epsilon = \max\{10^{-N/8}, 10^{-15}\}$.

M	N	BPRI method			FCPRI method			BCG method		
		$\|w^{(IT)}\|$	IT	CPU(sec)	$\|w^{(IT)}\|$	IT	CPU(sec)	$\|w^{(IT)}\|$	IT	CPU(sec)
4	8	0.40580131	2	0.01	0.40443277	2	0.01	0.39661736	2	0.01
4	16	0.39807268	3	0.02	0.39682981	5	0.03	0.39695696	3	0.02
8	32	0.39353375	3	0.05	0.39353052	5	0.07	0.39352994	4	0.05
16	64	0.39253860	3	0.22	0.39253861	5	0.20	0.39253860	6	0.21
16	128	0.39228584	5	0.96	0.39228584	10	0.87	0.39228546	8	0.93
32	256	0.39222236	3	3.02	0.39222236	4	2.12	0.39222236	8	3.61
64	512	0.39220647	2	11.03	0.39220647	2	6.99	0.39220647	9	15.54

The number of iterations for the BPRI method is same as in Table 1, but the number of iterations for the FCPRI method is quite different. The reason is that the BPRI has a smoothing step while the FCPRI does not. This smoothing step makes the convergence behavior of the BPRI insensitive to the nonsmooth right-hand side.

The third example is for a nonconvex kite-shaped boundary $\nu(s) = (\cos(s)+0.65\cos(2s)-0.65, 1.5\sin(s))$ and for the right-hand side of (4.1) equal to $0.5\cos(s)$. The numerical results are reported in Table 3.

In the second and third examples we are not able to report the errors since the exact solutions are not known. However, it is clear that the three methods converge to the same

TABLE 3
$\epsilon = \max\{10^{-N/8}, 10^{-15}\}$.

M	N	BPRI method			FCPRI method			BCG method		
		$\|w^{(IT)}\|$	IT	CPU(sec)	$\|w^{(IT)}\|$	IT	CPU(sec)	$\|w^{(IT)}\|$	IT	CPU(sec)
4	8	0.71873226	2	0.01	0.72675829	2	0.01	0.70013867	3	0.01
4	16	0.72309351	3	0.02	0.72013247	5	0.03	0.71889127	4	0.02
8	32	0.72119042	4	0.07	0.72118769	8	0.09	0.72117184	6	0.07
16	64	0.72116795	4	0.27	0.72116795	6	0.26	0.72116795	9	0.30
16	128	0.72116795	7	1.28	0.72116795	11	1.08	0.72116795	12	1.34
32	256	0.72116795	4	4.00	0.72116795	6	3.10	0.72116795	12	5.24
64	512	0.72116795	2	13.47	0.72116795	2	9.45	0.72116795	12	20.97

digits in a small number of iterations. As expected from the theoretical estimate, the FCPRI method is faster than the BPRI method when N becomes sufficiently large. In addition, both the BPRI and FCPRI methods are faster than the BCG method for the large linear system. This is expected. For example, the second example yields a symmetric positive definite matrix, and so the convergence rate of the BCG method relies on how close the condition number of the matrix is to the unity. Although the condition number of the matrix is independent of n based on the approximation theory of Fredholm integral equations, this condition number is not close to the unity. To make the BCG method competitive with the BPRI and FCPRI methods a preconditioner as used in BPRI and FCPRI is required to ensure the condition number close to one.

Using its single layer potential representation, the solution of the exterior Dirichlet boundary value problem

$$\Delta u = 0 \quad \text{for } x \in \mathbf{R}^2 \backslash \Omega,$$
$$u = f \quad \text{for } x \in \partial\Omega,$$
$$u(x) = c \log|x| + o(1), \quad |x| \to \infty$$

can be reformulated as the solution of the first kind boundary integral equation

$$-\frac{1}{\pi}\int_{-\pi}^{\pi} w(\sigma) \log\left|2e^{-1/2}\sin\frac{s-\sigma}{2}\right| d\sigma + \int_{-\pi}^{\pi} w(\sigma) b(s,\sigma) d\sigma = f(v(s)), \qquad x \in [-\pi, \pi], \tag{4.4}$$

where

$$b(s,\sigma) = -\frac{1}{\pi}\log\left|e^{1/2}\frac{v(s)-v(\sigma)}{2\sin\frac{s-\sigma}{2}}\right|.$$

We approximate this equation by the quadrature method

$$\sum_{k=-N/2+1}^{N/2}\left[\frac{1}{N}(1+2R_{j-k}) + hb(t_j,t_k)\right] w_k = f(v(t_j)), \qquad j = -\frac{N}{2}+1, \ldots, \frac{N}{2}, \tag{4.5}$$

where

$$R_l = \frac{(-1)^l}{N} + \sum_{k=1}^{N/2-1}\frac{1}{k}\cos\frac{kl2\pi}{N}, \qquad l \in \mathbf{Z},$$

which has an exponential rate of convergence when $v(s)$ and $f(v(s))$ are smooth (see [19]). Our test example for this problem is for an elliptic boundary $v(s) = (\cos(s), 0.25\sin(s))$ and for

TABLE 4
$\epsilon = \max\{10^{-N/8}, 10^{-12}\}$.

M	N	FCPRI method		
		$\|w^{(IT)} - r_h w\|$	IT	CPU(sec)
4	8	3.35E–3	2	0.02
4	16	1.34E–3	3	0.02
8	32	5.50E–6	3	0.04
16	64	2.82E–11	3	0.14
16	128	1.26E–14	4	0.48
32	256	1.37E–14	2	1.42
64	512	2.82E–14	1	4.89

the right-hand side of (4.4) equal to $\cos(s)$. The solution of this equation is $w(s) = 2.5\cos(s)$. The numerical results are reported in Table 4. Since $w(s)$ is an eigenfunction for (4.4), the results of the BCG method are not presented.

In this example only the FCPRI method can be applied because the linear system (4.5) appears only in the form of (1.2). The exponential rate of convergence of the quadrature method is retained after few iterations.

The final example is for the same elliptic boundary but with the nonsmooth function $|\sin t|^3$ as in the right-hand side of (4.4).

TABLE 5
$\epsilon = \max\{10^{-N/8}, 10^{-12}\}$.

M	N	FCPRI method			BCG method		
		$\|w^{(IT)}\|$	IT	CPU(sec)	$\|w^{(IT)}\|$	IT	CPU(sec)
4	16	0.48809454	12	0.03	0.48912633	4	0.02
8	32	0.48954680	17	0.12	0.48954909	7	0.07
16	64	0.48953378	11	0.25	0.48953379	16	0.38
16	128	0.48953297	19	1.09	0.48953297	40	3.08
32	256	0.48953293	6	1.93	0.48953293	50	14.31
64	512	0.48953292	2	5.19	0.48953292	77	83.93

In comparison to the FCPRI method, the BCG method is a bad choice for this example because of its slow convergence. Here the condition number of the matrix is proportional to the matrix dimension n, and, again, a preconditioner is required to improve performance of the BCG method.

Acknowledgment. The author wishes to thank the referees for their constructive suggestions.

REFERENCES

[1] H. BRAKHAGE, *Über die numerische Behandlung von Integralgleichungen nach der Quadraturformelmethode*, Numer. Math., 2 (1960), pp. 183–190.

[2] K. ATKINSON, *Iterative variants of the Nyström method for the numerical solution of integral equations*, Numer. Math., 22 (1973), pp. 17–31.

[3] ———, *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1976.

[4] ———, *A discrete Galerkin method for first kind integral equations with a logarithmic kernel*, J. Integral Equations Appl., 1 (1988), pp. 343–363.

[5] K. ATKINSON AND I. GRAHAM, *Iterative solution of linear system arising from boundary integral methods*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 694–722.

[6] K. ATKINSON, *Two-grid iteration methods for linear integral equations of the second kind on piecewise smooth surfaces in* R^3, SIAM J. Sci. Comput., 15 (1994), pp. 1083–1104.

[7] I. G. GRAHAM AND Y. YAN, *Piecewise-constant collocation for first-kind boundary integral equations*, J. Austral. Math. Soc. Ser. B., 33 (1991), pp. 39–64.
[8] A. GREENBAUM, L. GREENGARD, AND A. MAYO, *On the numerical solution of the biharmonic equation in the plane*, preprint.
[9] W. HACKBUSCH, *Die schnelle Auflösung der Fredholmschen Integralgleichungen zweiter Art*, Beit. Numer. Math., 9 (1981), pp. 47–62.
[10] ———, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.
[11] F. K. HEBEKER, *On multigrid methods for the first kind for symmetric boundary integral equations of nonnegative order*, in Robust Multigrid Methods, W. Hackbusch, ed., Braunschweig, 1988, pp. 128–138.
[12] P. W. HEMKER AND H. SCHIPPERS, *Multiple grid methods for the solution of Fredholm integral equations of the second kind*, Math. Comput., 36 (1981), pp. 215–232.
[13] R. KRESS, *Linear Integral Equations*, Springer-Verlag, Berlin, Heidelberg, 1989.
[14] J. MANDEL, *On multilevel iterative methods for integral equations of the second and related problems*, Numer. Math., 46 (1985), pp. 147–157.
[15] K. NABORS, T. KORSMEYER, AND J. WHITE, *Multipole-accelerated preconditioned iterative methods for solving three-dimensional mixed first and second kind integral equations*, in Proc. 1992 Copper Mountain Conference on Iterative Methods, Copper Mountain, CO.
[16] N. NACHTIGAL, S. REDDY, AND N. TREFETHEN, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.
[17] L. REICHEL, *A method for preconditioning matrices arising from linear integral equations for elliptic boundary value problems*, Computing, 37 (1986), pp. 125–136.
[18] ———, *A matrix problem with application to rapid solution of integral equations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 263–280.
[19] L. REICHEL AND Y. YAN, *A fast numerical solution for a class of periodic pseudodifferential equations*, J. Int. Equa. Appl., to appear.
[20] V. ROKHLIN, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1985), pp. 187–207.
[21] H. SCHIPPERS, *Multigrid methods for boundary integral equations*, Numer. Math., 46 (1985), pp. 351–363.
[22] S. VAVASIS, *Preconditioning for boundary integral equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 905–925.
[23] Y. YAN, *A fast numerical solution for second kind boundary integral equation with a logarithmic kernel*, SIAM J. Numer. Anal., 31 (1994), pp. 477–498.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1201–1226, September 1994

014

THE TORUS-WRAP MAPPING FOR DENSE MATRIX CALCULATIONS ON MASSIVELY PARALLEL COMPUTERS*

BRUCE A. HENDRICKSON† AND DAVID E. WOMBLE†

Abstract. Dense linear systems of equations are quite common in science and engineering, arising in boundary element methods, least squares problems, and other settings. Massively parallel computers will be necessary to solve the large systems required by scientists and engineers, and scalable parallel algorithms for the linear algebra applications must be devised for these machines. A critical step in these algorithms is the mapping of matrix elements to processors. In this paper, the use of the *torus-wrap mapping* in general dense matrix algorithms is studied from both theoretical and practical viewpoints. Under reasonable assumptions, it is proved that this assignment scheme leads to dense matrix algorithms that achieve (to within a constant factor) the lower bound on interprocessor communication. It is also shown that the torus-wrap mapping allows algorithms to exhibit less idle time, better load balancing, and less memory overhead than the more common row and column mappings. Finally, practical implementation issues are discussed, such as compatibility with basic linear algebra subprograms (BLAS) levels 1, 2, and 3, and the results of implementations of several dense matrix algorithms are presented. These theoretical and experimental results are compared with those obtained from more traditional mappings.

Key words. LU factorization, QR factorization, Householder tridiagonalization, parallel computer, dense matrix, torus-wrap mapping

AMS subject classifications. 65Y05, 65F05

1. Introduction.

Dense linear systems of equations are quite common in science and engineering applications, appearing in boundary element methods, problems involving all-pairs interactions, and least squares problems, among others. The kernel computation for these applications usually involves some factorization of the matrix to transform it to a more convenient form. The factored matrix can then be used to solve linear systems of equations, perform least squares calculations, determine eigenvectors and eigenvalues, or whatever other computation the application requires [19].

For a dense $n \times n$ matrix, most of these factorization algorithms require $\Theta(n^3)$ floating point operations and $\Theta(n^2)$ storage. Current sequential supercomputers can store and operate on systems with tens of thousands of unknowns. For example, a single processor CRAY Y–MP with 256 megawords of memory operating at its peak theoretical speed of 333 million floating point operations per second (Mflop/s) can compute the LU factorization of a 16,000 $\times$ 16,000 matrix in about 2.3 hours. Solving substantially larger problems or sequences of moderately sized problems on even the fastest single processor machines is prohibitively time consuming. It is clear from the computational requirements that to solve such problems will require computers capable of billions of floating point operations per second. This, in turn, will require massively parallel computers based on scalable architectures. To effectively use these machines, algorithms must be devised that scale well to large numbers of processors.

Efficient use of massively parallel computers is a subject of much current research, and numerous papers have been published about dense linear algebra algorithms on these machines. Because of its importance for solving linear systems, the LU factorization (and the related triangular solve) has been the primary subject of this research [1], [2], [4], [5], [11], [12], [16], [17], [21], [26], [31], [32]. There have been far fewer papers concerned with the

*Received by the editors May 28, 1992; accepted for publication (in revised form) September 3, 1993. This work was supported by the Applied Mathematical Sciences 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 number DE-AC04-76DP00789.

†Sandia National Laboratories, Albuquerque, New Mexico 87185 (bahendr@cs.sandia.gov, dewomble@cs.sandia.gov).

efficient computation of the QR factorization, Householder tridiagonalization, or the eigenvalue problem [7], [8], [22], [37]. Fewer still have tried to address dense matrix algorithms in general.

Early implementations of dense matrix algorithms, and in particular the LU factorization, mostly used row or column decompositions in which entire rows or columns of the matrix were assigned to individual processors [17], [18], [21], [26]. The columns or rows that a processor owned were usually "wrapped" or scattered throughout the matrix to obtain good load balancing. On computers with 64–128 processors, the efficiencies of these algorithms were usually between 50% and 75% for the largest problems that could be stored on the machines, but the algorithms did not scale particularly well as the number of processors increased [17], [21].

An alternative method for assigning matrix elements to processors is the torus-wrap mapping. Variants of this assignment scheme have been independently discovered by several researchers, and consequently given a number of different names including cyclic [23], scattered [15], grid [36], and subcube-grid [8], as well as torus-wrap [30]. The mapping was first described by O'Leary and Stewart in a data-flow context [29], [30], and the synergy between the torus-wrap mapping and the hypercube topology was observed by Fox [14], [15]. Variants of the torus-wrap mapping have been used in high performance LU factorization codes on a number of different machines [3], [4], [6], [9], [27], [35], [36]. Assuming each matrix element is stored on only a single processor, Ashcraft built on work by Saad to show that for LU factorization, the torus-wrap mapping exhibits communication properties within a constant factor of optimal [1], [32]. (Ashcraft has recently devised an algorithm with lower-order communication that violates this nonduplication assumption, but it requires an impractical factor of $p^{1/3}$ additional storage, where p is the number of processors [2].) Various algorithms for QR factorization employing the torus-wrap mapping have been described that use Givens rotations [8], modified Gram–Schmidt [37], and Householder reflections [22], [27]. A torus-wrap mapping algorithm for Householder tridiagonalization is described in [7]. A triangular solve algorithm using this mapping that achieves asymptotically optimal performance is presented in [5], [28]. Because of its scaling properties, the torus-wrap mapping has been suggested as the basic decomposition for parallel dense linear algebra libraries [13].

Despite the evident recent popularity of the torus-wrap mapping for a number of different dense linear algebra implementations, a careful analysis of the strengths and weaknesses of the mapping has been lacking. One purpose of this paper is to provide such an analysis, including communication overhead, memory requirements, and load balancing issues. Our approach is to identify the critical computation and communication components of dense matrix operations on distributed memory computers and then to analyze the impact of different mappings on the performance of these components. Thus, the results in this paper are more general than much of the current literature, and we anticipate that our analysis will provide a basis for future research in this area.

Another purpose of this paper is to explore the practical aspects of implementations of dense matrix algorithms using the torus-wrap mapping. Three algorithms are actually implemented for this end, LU factorization, QR factorization, and Householder tridiagonalization. Using these implementations, we compare the performance of a range of torus-wrap mappings with that of the row-wrap and column-wrap mappings, and we examine the scalability of the torus-wrap mapping to large numbers of processors using numerical results obtained on a 1,024-processor nCUBE 2. We also present models of performance for these three implementations that allow us to examine such effects as communication/computation overlap and the effect of vector lengths on communications and on the BLAS operations. These models then allow us to predict the optimal torus-wrap decompositions.

In §2, we characterize the basic operations required for dense matrix algorithms and their implications in the parallel computing environment, deriving lower bounds on required interprocessor communication. We define the torus-wrap mapping in §3 and describe its relationship to more familiar decomposition schemes. In §4, we discuss in more detail the properties of the torus-wrap decomposition, including its communication requirements, scalability, and compatibility with the standard BLAS routines. We present data from implementations of several different dense matrix algorithms in §5. These results clearly show the advantages of the torus-wrap mapping. Conclusions are presented in §6.

2. Dense linear algebra operations and communication. Nearly all dense linear algebra algorithms consist of a sequence of fundamental operations that transform a matrix into some more desirable form. The two most important such operations are Gauss transformations and Householder reflections. These operations generally dominate the computational effort in a dense linear algebra algorithm, so their efficient execution is essential for good performance. On a message passing multiprocessor the execution time of an algorithm can depend greatly upon its communication patterns, so minimizing the communication for these fundamental operations is important for achieving good parallel performance [23], [24], [33].

To understand the communication required to perform Gauss and Householder transformations, consider Fig. 1, where A is an $m \times n$ matrix, u is an m-vector and v an n-vector. Under either a Gauss transformation or a Householder reflection (or a Gauss–Jordan transformation), each element of A is updated by the outer product of u and v; that is $A_{ij} \leftarrow A_{ij} + u_i v_j$. The difference between the algorithms is in the construction of u and v, which is a lower-order operation in both computation and communication. The outer product update of an element of A depends on the element of u directly to its left and the element of v above it. The processor that calculates the new value for A_{ij} must know the old A_{ij} as well as u_i and v_j, which may require some communication. We will establish a lower bound on the total communication volume for this operation, which is a subset of the communication required to perform a Gauss or Householder transformation.

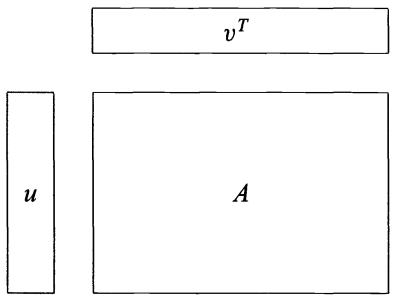


FIG. 1. *Structure of Gauss and Householder transformations.*

We denote by $N(q)$ the number of matrix elements owned by processor q, and let p be the total number of processors. We will assume that

(i) each element of A (and of u and v) is owned by a single processor, and

(ii) the matrix elements are balanced; that is, for each processor q, $N(q) \geq \alpha mn/p$ for some constant $\alpha > 0$.

We define the *communication volume* V_C of an algorithm to be the total length of all the messages the algorithm requires. For numerical algorithms, messages typically consist of floating point numbers, so the lengths are most naturally measured in terms of number of floating point values. The following theorem is a generalization of results found in [32].

THEOREM 2.1. *Under assumptions* (i) *and* (ii) *above, the communication volume required to execute a Gauss, Householder, or Gauss–Jordan transformation is at least* $2\sqrt{\alpha pmn} - (m+n)$.

Proof. Each element in the matrix needs the value of v above it and the value of u to its left. Let t_i^r denote the number of processors owning elements of row i and t_j^c be the number of processors owning elements of column j. To transmit u_i (or v_j) to all the processors in row i (or column j) requires at least $t_i^r - 1$ (or $t_j^c - 1$) messages of length 1, so the communication volume can be bounded by

$$V_C \geq \sum_{i=1}^{m}(t_i^r - 1) + \sum_{j=1}^{n}(t_j^c - 1). \tag{1}$$

Now we denote by s_q^r (and s_q^c) the number of rows (and columns) of which processor q owns at least one element. It follows from the definitions that $\sum_{q=1}^{p} s_q^r = \sum_{i=1}^{m} t_i^r$, and $\sum_{q=1}^{p} s_q^c = \sum_{j=1}^{n} t_j^c$. Substituting these identities into (1) yields

$$\begin{aligned} V_C &\geq -(m+n) + \sum_{q=1}^{p}(s_q^r + s_q^c) \\ &\geq -(m+n) + \sum_{q=1}^{p} 2\sqrt{s_q^r s_q^c}. \end{aligned}$$

Assumption (ii) ensures that for each q, $s_q^r s_q^c \geq N(q) \geq \alpha mn/p$, so

$$\begin{aligned} V_C &\geq -(m+n) + \sum_{q=1}^{p} 2\sqrt{\alpha mn/p} \\ &= 2\sqrt{\alpha pmn} - (m+n). \end{aligned}$$

□

COROLLARY 2.2. *Under assumptions* (i) *and* (ii) *above, the communication volume required to execute a Gauss, Householder, or Gauss–Jordan transformation on a square* $n \times n$ *matrix is at least* $2n(\sqrt{\alpha p} - 1)$.

Ashcraft has recently proposed an LU factorization algorithm that requires $\Theta(p^{1/3}n^2)$ communication volume, but it violates assumption (i) [2]. This algorithm is impractical in its current form, requiring an extra factor of $p^{1/3}$ storage.

The lower bound expressed in Theorem 2.1 is attainable, up to a constant factor. For simplicity we let m and n be divisible by $\sqrt{p}$, and assume that $m + n \leq \gamma \min(m, n)$ for some constant γ. If we assign each processor a dense rectangular block of the matrix of size $(m/\sqrt{p}) \times (n/\sqrt{p})$, then $\alpha = 1$, and each row and each column will be owned by only $\sqrt{p}$ of the processors. The total communication volume involved in broadcasting a row will be $n(\sqrt{p}-1)$, and for a column $m(\sqrt{p} - 1)$, implying a total of $(n+m)(\sqrt{p} - 1) \leq \gamma\sqrt{mnp} - (m+n)$, which is within a constant factor of the bound from Theorem 2.1.

We note that if each column (or row) of the matrix is owned by a single processor, then a Gauss or Householder transformation requires the broadcast of that column (or row) to all other processors. This involves a communication volume of $m(p-1)$ (or $n(p-1)$). Assuming again that $m+n \leq \gamma \min(m, n)$ for some constant γ, this volume is at least $(2/\gamma)\sqrt{mn}(p-1)$, which is larger than the lower bound by $\Theta(\sqrt{p})$.

Finally, we observe that the results in this section are a consequence of the fact that dense linear algebra operations can be formulated to require only a restricted form of communication. Values must be exchanged within each row of the matrix and within each column. Any operation that involves this communication pattern will be amenable to a similar analysis.

3. The torus-wrap mapping. Most of the previous work on parallel dense linear algebra has involved assigning elements of the $m \times n$ matrix A to processors using columns, rows, or blocks. In a column (or row) scheme, entire columns (or rows) of the matrix are assigned to a single processor. One possibility is to have columns 1 through n/p assigned to processor zero, columns $n/p+1$ through $2n/p$ assigned to processor one and so on. Since most matrix factorizations work from left to right, decreasing the number of active columns, this scheme has the disadvantage that processor q has no work left to do after column $(q+1)n/p$ is processed. For this reason, it is preferable to assign columns $1, p+1, 2p+1, \ldots$ to processor zero, columns $2, p+2, 2p+2, \ldots$ to processor one, and so forth to form what is known as a *column-wrap* mapping. Column-wrap (and row-wrap) mappings have been the most widely used choice for dense linear algebra algorithms, but as noted in §2, row and column methods require $\Theta(\sqrt{p})$ more communication volume than necessary. For machines with a small number of processors, the simplicity of these mappings may outweigh the communication drawbacks, but for massively parallel machines this factor of $\sqrt{p}$ can be very important.

In block schemes, each processor is assigned a single dense rectangular submatrix. Many different block mappings are possible involving differently shaped rectangular submatrices and different assignments of blocks to processors. As mentioned in §2, block schemes can come within a constant of achieving the lower bound on communication volume. However, they have the same problems with idle processors that the nonwrapped row and column methods have. Hybrid *block-column-wrap* or *block-row-wrap* mappings are also possible. In these mappings, instead of owning a scattered set of single columns (or rows), a processor owns a scattered set of several adjacent columns (rows).

An analogy with row and column methods suggests wrapping a block mapping in both rows and columns. The result is what we will call the *torus-wrap* mapping. As there are many different block mappings, there are correspondingly many torus-wrap mappings. If the number of processors p can be factored as a product of p_r and p_c, then we can construct a block mapping in which the blocks are of size $(m/p_r) \times (n/p_c)$. For any appropriate p_r and p_c values, we get a block mapping and its torus-wrap counterpart. We note that in the limiting cases the torus-wrap mapping reduces to a row-wrap mapping (when $p_r = p$ and $p_c = 1$), or a column-wrap mapping (when $p_c = p$ and $p_r = 1$). We call the special case in which $p_r = p_c$ a *square* torus-wrap mapping. If it is not the case that m is divisible by p_r and n by p_c then some processors will own one more row and/or column than others.

Another choice in a block mapping occurs when deciding which blocks get assigned to which processors. Because communication in dense linear algebra algorithms occurs predominantly within rows or within columns, it is convenient to assign blocks in such a way that communication among the set of processors owning a row (or column) is efficient. With mesh architectures, this is achieved by constructing a set of blocks that reflect the shape of the processor array. If the mesh is constructed as a rectangle of $p_r \times p_c$ processors, then the blocks are of size $(m/p_r) \times (n/p_c)$, which are assigned to processors in the natural way. The rows and columns are then wrapped to generate the corresponding torus-wrap assignment. This constitutes what we will call a *natural* torus-wrap mapping, and ensures that row (or column) communication occurs entirely within rows (or columns) of the processor mesh. An example of the natural torus-wrap is depicted in Fig. 2, where the mesh is of size 8×4, and the value displayed at each location is the processor that owns the corresponding matrix entry.

For hypercubes, we can exploit the fact that a d-dimensional hypercube can be viewed as the product of two hypercubes of dimensions d_r and d_c, where $d_r + d_c = d$. This is accomplished by dividing the bits of the processor identifier into two sets, b_r and b_c of cardinality d_r and d_c, respectively. The bits of b_r can be associated with the row numbering of the matrix elements, and those of b_c with the column numbering. That is, all the processors owning elements from a single row of A have the same b_r bits, and all processors with elements from

a column have the same b_c bits. This assignment scheme ensures that communication within a row involves changing only bits of b_c, while communication within a column involves only bits of b_r. So each row of the matrix lies within a subcube of dimension d_c, and each column in a subcube of dimension d_r. In this way, a total of $p_c = 2^{d_c}$ processors are assigned elements of each row, and $p_r = 2^{d_r}$ processors are assigned elements of each column.

0	1	2	3	0	1	2	3
4	5	6	7	4	5	6	7
8	9	10	11	8	9	10	11
12	13	14	15	12	13	14	15
16	17	18	19	16	17	18	19
20	21	22	23	20	21	22	23
24	25	26	27	24	25	26	27
28	29	30	31	28	29	30	31
0	1	2	3	0	1	2	3
4	5	6	7	4	5	6	7

FIG. 2. *Processors owning matrix elements in a natural torus-wrap.*

One method for assigning processor numbers on a hypercube is to take the row number of a matrix element, subtract one, express the result modulo p_r, and then take the Gray code of the resulting d_r bit number. (For a discussion of Gray codes, see [15].) Performing the analogous calculation on the column bits, and assigning matrix elements to the resulting processors generates the *Gray-coded* torus-wrap, which is discussed in detail by Chu and George in [8]. An example corresponding to Fig. 2 is depicted in Fig. 3. The advantage of the Gray-coded torus-wrap is that neighboring elements of the matrix are owned by neighboring processors. More formally, a Gray-coded torus-wrap embeds a p_r by p_c mesh into a hypercube.

0	1	3	2	0	1	3	2
4	5	7	6	4	5	7	6
12	13	15	14	12	13	15	14
8	9	11	10	8	9	11	10
24	25	27	26	24	25	27	26
28	29	31	30	28	29	31	30
20	21	23	22	20	21	23	22
16	17	19	18	16	17	19	18
0	1	3	2	0	1	3	2
4	5	7	6	4	5	7	6

FIG. 3. *Processors owning matrix elements in a Gray-coded torus-wrap.*

Figures 2 and 3 indicate another way to define the torus-wrap mapping. The assignment pattern of the leading p_r by p_c submatrix defines a tile, and the entire matrix is covered by copies of this tile in the obvious way.

An important generalization of the torus-wrap mapping is the *block-torus-wrap*, in which the matrix is first decomposed into a collection of blocks of size $\delta_1 \times \delta_2$. Each block is assigned

to a single processor in such a way that the distribution of blocks mirrors the distribution of elements in a torus-wrap mapping. We note that the torus-wrap is a special case of the block-torus-wrap in which $\delta_1 = \delta_2 = 1$. The block-torus is a natural generalization of block-row and block-column methods. Using blocks instead of single elements has both advantages and disadvantages, as will be discussed in the next section.

It is convenient to note that for all the mappings considered above, each processor is assigned precisely the matrix elements that lie in the intersection of a particular set of rows and columns. Bisseling and van de Vorst call mappings with this property *Cartesian* [4]. Merely specifying a set of row indices and column indices for each processor uniquely defines a Cartesian mapping. Under a block mapping, a processor is assigned a consecutive set of row indices and a consecutive set of column indices. In a torus-wrap mapping, the row indices assigned to a processor constitute a linear sequence separated by p_r, and the column indices a sequence with step size p_c. In a row (or column) mapping, each processor is assigned all of the column (or row) indices and a subset of the row (or column) indices.

We conclude this section by observing that a matrix distributed among processors in a torus-wrap format can be viewed as a permutation of a matrix distributed in a block scheme. Specifically, a matrix A_b distributed in a block-wrap format can be treated as $\Pi_1^T A_t \Pi_2$, where Π_1^T and Π_2 are permutation matrices and A_t is a matrix distributed in a torus-wrap format. This equivalence allows for a different interpretation of the torus-wrap mapping. As observed in [27], a standard factorization algorithm on A_t can be viewed as a factorization of A_b in which the rows and columns are eliminated in a permuted order. An important result of this observation is that it is possible to take advantage of the attractive properties of a torus-wrap mapping without necessarily having to redistribute the matrix among processors. For example, a block system can be solved by a routine that assumes a torus-wrapped system, and the result is correct up to permutations. More precisely, $A_b^{-1}x = \Pi_2^T A_t^{-1} \Pi_1^T x$, so a block mapping can be made to perform like a torus-wrap mapping, without redistributing the matrix, by merely permuting the right-hand side and the solution vectors.

4. Virtues of the torus-wrap mapping. The torus-wrap mapping has a number of distinct advantages over the more conventional assignment schemes. These virtues become increasingly important as the number of processors increases. Generally, the torus-wrap mapping requires less communication than row or column schemes and it has excellent load balancing properties. These advantages will be discussed in detail in the following subsections, and their impact on performance of a collection of linear algebra algorithms will be presented in §5. Many of these issues are familiar to researchers who have used the torus-wrap mapping and have been touched on in a number of publications describing specific implementations.

4.1. Communication volume. The square torus-wrap mapping allows Gauss and Householder transformations to be executed with a total communication volume of $\Theta(\sqrt{mnp})$, which is within a constant factor of optimal. Row and column schemes require about a factor of $\sqrt{p}$ more communication. We note, however, that if the matrix or the number of processors is small, message startup time dominates the message transmission time so that this factor of $\sqrt{p}$ is not seen. Also, when p is small, the reduced communication volume may not compensate for the increased complexity of the torus-wrap. The direction of high performance computing is toward large problems and massive parallelism, so the factor of $\sqrt{p}$ will become increasingly important.

Most linear algebra algorithms involve a sequence of about $\bar{m}$ Gauss or Householder transformations, where $\bar{m} = \min(m, n)$. This implies an overall communication volume of $\Theta(\bar{m}\sqrt{pmn})$ for torus-wrap, and $\Theta(p\bar{m}\sqrt{mn})$ for row and column mappings. The number of floating point operations (flops) is typically $\Theta(\bar{m}mn)$, which, if balanced, implies $\Theta(\bar{m}mn/p)$ per processor.

If we wish to solve larger problems by increasing the number of processors without increasing the memory of each processor, then the largest problem we can solve has $mn = cp$ for some constant c. This implies that the number of flops per processor is $\Theta(\bar{m})$ and the torus-wrap communication volume is $\Theta(p\bar{m})$, while the row or column communication volume is $\Theta(p^{1.5}\bar{m})$. All contemplated interconnection networks for massively parallel machines have $\Omega(p)$ wires, so the torus-wrap communication requirements have the potential to scale well. However, no proposed network has $\Omega(p^{1.5})$ wires, so row and column schemes will eventually be limited by communication. Similar but more detailed analyses for LU factorization can be found in [1], [4].

This advantage is not specific to Gauss and Householder transformations. The proof of Theorem 2.1 can be applied to any operation that requires communication within rows and columns, implying that such operations can be performed with near optimal communication volume using a torus-wrap mapping.

4.2. Communication parallelism. Although helpful for scaling analyses, communication volume may not be a particularly useful metric for performance modeling because it ignores any overlap in the communication operations. It is often the case that several messages can be transmitted simultaneously on different communication channels. To correctly predict performance, the times required by these overlapping messages should not be added. We define the *effective communication volume* of an algorithm to be the total length of all the messages that are not overlapped. The effective communication volume is a good estimate of the time required for communication operations when most messages are long so message startup time is negligible. This is the case for most dense linear algebra algorithms on large matrices.

With a torus-wrap mapping, the transmission of a column involves p_r overlapping broadcasts to $p_c - 1$ other processors. Assuming a logarithmic broadcast and a message length of m/p_r, this results in an effective communication volume of $d_c m/p_r$. Similarly, the transmission of a row requires an effective volume of $d_r n/p_c$. When $d_c = d_r = d/2$, the combined effective volume is $d(m+n)/(2\sqrt{p})$, but for the limiting cases of row or column mappings, it is dn or dm, respectively. If m and n are about equal, the square torus-wrap mapping has an effective communication volume of about a factor of $\sqrt{p}$ less than that of row or column mappings. When the matrix is not square, or the row and column communication loads are not exactly equal, a nonsquare torus-wrap mapping may be best. This will be the case for some of the examples we consider in §5.

Since it roughly approximates communication time, the effective communication volume allows us to investigate the proportion of execution time devoted to communication. For dense factorizations of square matrices the sequential operation count is usually $\Theta(n^3)$, so if the load is well balanced the time spent performing these operations should be $\Theta(n^3/p)$. A dense factorization typically requires $\Theta(n)$ Gauss or Householder transformations, so the ratio of effective communication volume to parallel operation time is $\Theta((d_r p_r + d_c p_c)/n)$. For row or column schemes, this implies that the ratio of communication time to compute time grows as $\Theta(dp/n)$, but for a square torus it only grows as $\Theta(d\sqrt{p}/n)$. Assuming that each processor has finite memory, n can only grow as $\sqrt{p}$. In this case, the relative cost of communication scales as $\Theta(d\sqrt{p})$ for row and column mappings, but as $\Theta(d)$ for a square torus. The proportion of time spent on communications grows much more slowly for a square torus than for row or column mappings, allowing scalability to much larger machines.

For hypercubes, there are sophisticated broadcast schemes that manage to overlap communications more effectively than simple logarithmic broadcasts [24]. These algorithms use all of the wires in a cube (or subcube) to perform a broadcast. Using these algorithms, the row and column transmissions involve a combined effective communication volume of $m/p_r + n/p_c$.

When n and m are about equal, a square torus is still better than a row or column method by about a factor of $\sqrt{p}/2$.

4.3. Message queue overhead. Row or column schemes require broadcasts of entire rows or columns of the matrix. Torus-wrap methods only require broadcasts of subsets of rows and columns with lengths n/p_c and m/p_r. To exploit the advantages of asynchronous communications, a processor that expects to receive a message must have space reserved for it. The amount of reserved space is less for a square torus than for a row or column mapping by a factor of about $\sqrt{p}$. This leaves more space that can be devoted to other things, like storing matrix elements. Consequently, the torus-wrap mapping allows larger problems to be solved than row or column mappings.

With block-torus-wrap methods, sets of blocks are broadcast together. The total communication volume is unchanged, as is the effective volume, but since sets of columns (or rows) are broadcast together, the number of startups is decreased and the lengths of messages are increased by a factor of δ_2 (or δ_1). This reduces the total communication time but increases the amount of memory required for communication.

4.4. BLAS compatibility. Much work has been done developing a standard set of basic linear algebra subprograms or BLAS, which are available as a high performance library on many machines [10], [25]. The BLAS were devised with dense vectors or matrices in mind, but with the torus-wrap mapping each row and column is scattered. Although processors do not own any consecutive portion of the matrix, the submatrix assigned to each processor is rectangular and can be stored in a dense matrix format. For all the operations required for Gauss, Householder, and Gauss–Jordan transformations, this submatrix can be treated as dense, which allows the use of levels one, two, and three BLAS operations.

4.5. Load imbalance. To achieve optimal performance from a parallel computer, it is important that each processor has nearly the same total amount of work to do; that is, the computational load must be balanced. For most dense matrix algorithms, the total number of floating point operations that have to be performed to update matrix element $A(i, j)$ is proportional to $\min(i, j)$. We will define the load of element $A(i, j)$ as $\min(i, j)$ and the load on a processor as the total load of all the elements it owns. Under a torus-wrap mapping, if A is an $m \times n$ matrix, the most heavily loaded processor will be the one owning element $A(m, n)$, and the least loaded will be the processor that would own $A(m+1, n+1)$ if A were larger. We denote the former processor by q and the latter by s, and let their loads be $W(q)$ and $W(s)$, respectively. We are interested in $\Delta = W(q) - W(s)$.

We observe that except for entries in the last row or last column of A, each element owned by q has a neighbor down and to the right that is owned by s. Similarly, with the possible exception of elements in the first row or first column, each value owned by s has a neighbor owned by q to the upper left. Each such pair of elements contributes a value of -1 to Δ. If we denote the number of pairs by N_p, then

$$\Delta = -N_p + W_{m,n}(q) - W_{1,1}(s),$$

where $W_{i,j}(x)$ is the load on processor x from elements in the row i and elements in the column j of the matrix.

We denote the first row partially owned by processor q in the torus-wrap mapping as r_0, and the corresponding first column as c_0. We assume for concreteness that $m \geq n$; the other case is analogous. The values of N_p and $W_{1,1}(s)$ are easy to compute. To compute $W_{m,n}(q)$ we separately sum the contributions from elements in the last row, elements in the last column that are above the diagonal, and the remaining elements in the last column on or below the diagonal. Some algebra gives the following result:

$$N_p = \left(\frac{m - r_0}{p_r}\right)\left(\frac{n - c_0}{p_c}\right) = (m - r_0)(n - c_0)/p,$$

$$W_{1,1}(s) = \lfloor c_0/p_c \rfloor \lfloor m/p_r \rfloor + \lfloor r_0/p_r \rfloor \lfloor n/p_c \rfloor - \lfloor c_0/p_c \rfloor \lfloor r_0/p_r \rfloor,$$

$$W_{m,n}(q) = \left(\frac{n + c_0}{2}\right)\left(\frac{n - c_0}{p_c} + 1\right) + \left\lceil \frac{n - r_0}{p_r} \right\rceil \left(r_0 + \frac{p_r}{2}\left(\left\lceil \frac{n - r_0}{p_r} \right\rceil - 1\right)\right) + n \left\lfloor \frac{m - n}{p_r} \right\rfloor.$$

In the limit as m, n, and p become large $W_{1,1}$ becomes negligible, as do r_0 and c_0, so Δ approaches $n^2/(2p_c) + n(m - n/2)/p_r - mn/p$. (If $n > m$ the corresponding result is $\Delta = m^2/(2p_r) + m(n - m/2)/p_c - mn/p$.) For a square matrix, this maximal load imbalance reduces to $n^2(p_r + p_c - 2)/(2p)$. So for a square matrix, a square torus-wrap induces less load imbalance than a row-wrap or column-wrap by a about a factor of $\sqrt{p}/2$. A similar conclusion is reached for parallel LU factorization by Bisseling and van de Vorst in [4].

Assuming that each processor has finite memory, then for square matrices n can grow as $\sqrt{p}$. Since the number of flops grows as n^3, the run time should scale roughly as n^3/p, which is proportional to $\sqrt{p}$. The maximum load imbalance scales as $p_r + p_c$, which is p for row or column mappings, but $\sqrt{p}$ for a square torus-wrap. Consequently, the proportional load imbalance increases as $\sqrt{p}$ for row or column mappings, but stays constant for a square torus-wrap.

Similar results apply for a block-torus mapping. For simplicity in the analysis, we assume that $\delta_2 = \delta_1 = \delta$, and that n and m are both divisible by δ. In this case, we can generalize the previous analysis by counting blocks instead of single elements. Letting the N_p and $W_{i,j}$ notation now apply to entire blocks, N_p scales linearly with δ because the workload imbalance associated with a pair of diagonally adjacent blocks increases as δ^3, but the number of blocks decreases as δ^2. Similarly, $W_{m,n}$ increases by about a factor of δ, while $W_{1,1}$ increases by about a factor of $\delta^2/2$. Since the the contribution of $W_{1,1}$ to Δ is negative, the growth in the load imbalance with delta is at most linear. For large m, n, and p and small δ, the N_p and $W_{m,n}$ terms will dominate $W_{1,1}$, so the load imbalance from a block-torus scheme with square blocks will approach $\delta\{n^2/(2p_c) + n(m - n/2)/p_r - mn/p\}$, where $m \geq n$. The important conclusion is that if the block sizes are small, the maximal load imbalance is proportional to the linear dimension of the blocks.

4.6. Processor idle time. Even if all the processors have the same total amount of work to do, the overall calculation will be inefficient unless each processor always has something to work on. Algorithms using Gauss or Householder transformations usually begin by computing a function of the first column (or row) of the matrix; the norm for Householder and the largest element for Gauss. If this first column (row) is not distributed, as in a column (row) mapping, then the other processors need to wait until this calculation is complete. This problem is exacerbated by using a block-column (or block-row) approach to allow level three BLAS, since several columns (rows) must be manipulated before the other processors have any work to do. A torus-wrap mapping allows some parallelism in the processing of each column or row, which reduces this potential idle time problem. With a block-torus-wrap mapping, the idle time at the beginning of the calculation grows linearly with δ_1 or δ_2.

Most factorizations eliminate the rows and/or columns of the matrix in order, from left to right and top to bottom. As the factorization nears completion, only processors that still own active matrix elements have work to do. With column (or row) schemes, processors begin to drop out when there are p columns (rows) remaining. Block methods are even worse. With the torus-wrap, however, all the processors stay active until there are only p_c columns (or p_r

rows) left to eliminate. If the matrix has p more (fewer) rows than columns then a row-wrap (column-wrap) mapping avoids this problem, but for the important special case of square and nearly square matrices, the torus-wrap mapping allows greater parallelism near the end of the computation. As at the beginning, with a block-torus-wrap the idle time at the end of the calculation grows linearly with δ_1 or δ_2.

5. Numerical results. In this section, we discuss the implementations of three dense matrix algorithms, LU factorization, QR factorization, and Householder tridiagonalization. For each of these examples, we investigate the performance implications of using different mapping schemes, both analytically and experimentally. All numerical experiments were performed in double precision C on a 1,024 node nCUBE 2 hypercube at the Massively Parallel Computing Research Laboratory at the Department of Energy's Sandia National Laboratory.

Our algorithms require only very simple communication patterns consisting of broadcast, collect, and binary exchange operations. We implemented each of these functions in a simple, generic way so that the resulting code should run on any architecture. Although our timings given are for a specific computer, the broad conclusions should be appropriate for other machines. For hypercubes there are asymptotically more efficient communication algorithms as was alluded to in §4.2, but our implementations do not exploit them.

5.1. LU factorization. Our first example is LU factorization with partial pivoting of a dense $n \times n$ matrix A. This the most important factorization in linear algebra as it is a very efficient method for solving systems of linear equations. There are several variants of LU factorization, each requiring $2n^3/3 + O(n^2)$ flops. Our algorithm uses the column-oriented, kij version as described in [19]. The algorithm is summarized in Fig. 4, where Roman subscripts denote integers and Greek subscripts denote sets of integers.

Our implementation of this algorithm incorporates double precision arithmetic and uses a Gray-coded torus-wrap mapping. As presented, this algorithm can be used with any Cartesian mapping, but in practice, block algorithms would be modified to send fewer, larger messages. The algorithm can be easily modified for a block-torus-wrap mapping, but since nCUBE only supports level one BLAS, we did not investigate this possibility. Improvements in performance have been observed on other machines using block algorithms [35]. To minimize the time spent waiting for the determination and broadcast of pivot elements our algorithm employs a compute-ahead technique. The processors owning the next column in the matrix generate and send the pivot information before updating the remainder of their elements.

The nCUBE 2 has 1,024 processors arranged in a ten-dimensional hypercube. Each processor has four Mbytes of memory, which must be divided between the operating system, code, data, and communication buffer. Using the optimal distribution of data among the processors, we can allocate about 3.8 Mbytes for data storage in our code. This means that the largest double precision, dense matrix that can be factored in core is about 22,000 $\times$ 22,000. Factoring the Linpack benchmark matrix of this size requires 3612 seconds (1.96 Gflop/s), using a processor decomposition in which $p_c = 64$ and $p_r = 16$, which is the optimal decomposition of 1,024 processors for this size problem as will be seen later in this section.

To investigate the effect of the different torus-wrap decompositions on scaling we need to use a matrix that can be stored on fewer than 1,024 processors. We factored an $8,000 \times 8,000$ matrix with different numbers of processors and the results are presented in Table 1. We note that although we can store a matrix of size $n = 11,000$ on 256 processors, the communication buffer requirements of the nonoptimal row and column distributions require that we reduce the size of the matrix, as was discussed in §4.3.

We can model the performance to gain insight into the scaling properties and the optimal balance between p_c and p_r. There are four major contributions to the run time of the LU

```
Processor q owns row set α and column set β
For j = 1 to n
    (* Find pivot row *)
    If j ∈ β Then
        (* Compute maximum of entries in column j *)
        γ^q := max_{i∈α} |A_{i,j}|
        Binary exchange to compute γ = max_q γ^q
        s := index of row containing the entry γ

    (* Generate update vector, v, from column j of A *)
    If j ∈ β Then
        A_{α,j} := A_{α,j}/γ
        v_α := A_{α,j}
        Broadcast column v_α and s to processors sharing rows α
    Else Receive v_α and s

    (* Exchange pivot row and diagonal row, and broadcast pivot row *)
    If j ∈ α Then
        Send w_β = A_{j,β} to processor owning A_{s,β}
    If s ∈ α Then
        Receive w_β
        u_β := A_{s,β}
        Broadcast row u_β to processors sharing columns β
        A_{s,β} := w_β
    Else Receive u_β
    If j ∈ α Then
        A_{j,β} := u_β

    If j ∈ α and j ∈ β Then
        A(j, j) = A(j, j) * γ    (* Restore diagonal *)

    α := α \ {j}    (* Remove j from active rows *)
    β := β \ {j}    (* Remove j from active columns *)

    A_{α,β} := A_{α,β} − v_α u_β
```

FIG. 4. *Parallel* LU *factorization for processor* q.

TABLE 1
Run times on the nCUBE 2 *for* LU *factorization of an* 8,000 × 8,000 *matrix.*

256 Processors			512 Processors			1,024 Processors		
p_c	p_r	Seconds	p_c	p_r	Seconds	p_c	p_r	Seconds
1	256	2568	1	512	2406	1	1,024	2394
2	128	1335	2	256	1064	2	512	953
4	64	930	4	128	624	4	256	483
8	32	777	8	64	462	8	128	307
16	16	719	16	32	396	16	64	237
32	8	706	32	16	372	32	32	207
64	4	729	64	8	376	64	16	201
128	2	806	128	4	412	128	8	215
256	1	996	256	2	506	256	4	260
			512	1	712	512	2	364
						1,024	1	589

factorization code: a pivot entry search and pivot column update, a row broadcast, a column broadcast, and an outer product update of the unfactored submatrix. We will need the following values taken from the nCUBE 2 manuals for our model.

Variable	Description	Microseconds
$T_{c,a}$	message startup time	100.
$T_{c,b}$	transmission time per double precision number	4.00
$T_{p,a}$	startup time for pivot operations	16.3
$T_{p,b}$	computational time for pivot operations (per element)	1.955
$T_{d,a}$	startup time for a daxpy	11.0
$T_{d,b}$	computational time for a daxpy (per element)	0.964

With these variables, the total time spent computing the outer product updates of the unfactored portion of the submatrix using the column-oriented daxpy can be modeled as

$$T_{\text{update}} = \frac{n^2}{2p_c} T_{d,a} + \frac{n^3}{3p} T_{d,b}.$$

The time spent on pivot entry searches, which includes a local search and a binary exchange by those processors containing part of a column, and pivot column updates can be approximated by

$$T_{\text{pivot}} = nT_{p,a} + \frac{n^2}{2p_r} T_{p,b} + d_r n(2T_{c,a} + 3T_{c,b}),$$

where we recall that d_r and d_c are the dimensions of the subcubes to which columns and rows are assigned, respectively. The time for a logarithmic broadcast of the pivot row is about

$$T_{\text{row}} = d_r \left(nT_{c,a} + \frac{n^2}{2p_c} T_{c,b} \right).$$

Finally, the time for a logarithmic broadcast of the pivot column is approximated by

$$T_{\text{column}} = d_c n T_{c,a} + \frac{n^2}{2p_r} T_{c,b}.$$

The total run time T_{total} is modeled by summing the four contributions above.

We note an important difference between the contributions to the overall run time of the row broadcasts and the column broadcasts. If the processor holding column $j+1$ is the first processor to complete its portion of the broadcast of the pivot column, then the search for the pivot and update of column $j+1$ can be overlapped with the remaining stages of the broadcast of column j. When a Gray-coded torus-wrap mapping is used on a hypercube, the processors holding column $j+1$ are neighbors of those owning column j. Consequently, a logarithmic broadcast can be structured in such a way that the processors owning column $j+1$ quickly receive column j and then have no further participation in the broadcast. Thus the factor d_c multiplies only the startup time in the contribution to the total run time. The row broadcast, on the other hand, cannot be overlapped with any computations (without destroying the load balance of the algorithm) so that d_r multiplies both the startup and transmission time contributions to the total run time.

The data from Table 1 is shown graphically in Fig. 5, where instead of the run time, the vertical axis is the Mflop/s rate achieved. To generate these values, we used the number of flops required by the sequential algorithm, about 3.41×10^{11}.

We observe that the optimal distribution is not achieved at $d_c = d_r$. Qualitatively, the major reason for this is the fact that column broadcasts can be overlapped with computations while row broadcasts cannot. T_{update} and T_{pivot} also contribute to this phenomenon since the startup time for column operations is reduced by increasing d_c, and the communication time required to find the pivot element (which is the dominant term in T_{pivot}) is reduced by decreasing d_r.

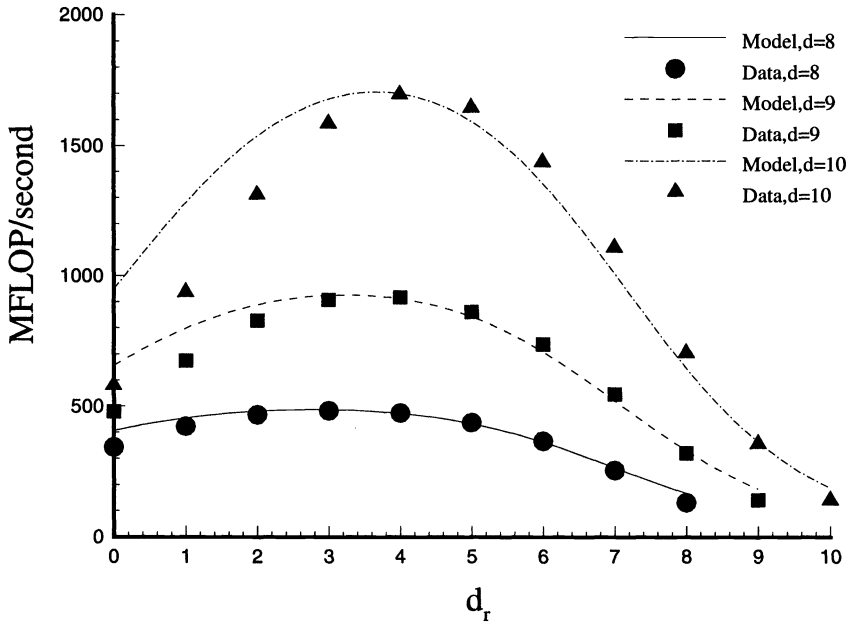


FIG. 5. *Performance of the* LU *factorization code on the* nCUBE 2.

This observation can be examined quantitatively by looking at the model. Specifically, if we write T_{total} in terms of n, d, d_c and the timing constants, differentiate with respect to d_c, and set the resulting expression equal to zero, we obtain the equation

$$2^{2d_c}\left(\frac{T_{p,b}+T_{c,b}}{2^d}\right)-2^{d_c}\left(\frac{4T_{c,a}+6T_{c,b})}{n\ln 2}\right)-\left(T_{d,a}+\left(d-d_c+\frac{1}{\ln 2}\right)T_{c,b}\right)=0.$$

We cannot solve this equation in closed form, but, in practice, simple numerical techniques can be used to compute the optimum value of d_c. Here, we derive an approximate expression for the optimum value of d_c. First, we observe that errors in the linear term involving d_c have only a small effect in the final solution and replace $(d-d_c+1/\ln 2)T_{c,b}$ with $dT_{c,b}/2$. Second, we observe that for most cases of interest $2^{2d_c-d} \gg 2^{d_c}/n$ so that the second term can be dropped from the expression. Now, solving for d_c yields

$$d_c=\frac{d}{2}+\frac{1}{2}\log_2\left(\frac{T_{d,a}+dT_{c,b}/2}{T_{p,b}+T_{c,b}}\right). \tag{2}$$

We note that even though this value is approximate, the predictions are close to the observed optima as shown in Fig. 5. The predicted peak performances (for integer values of d_c) are 487 Mflop/s for $d=8$ and $d_r=3$, 925 Mflop/s for $d=9$ and $d_r=3$, and 1700 Mflop/s for $d=10$ and $d_r=4$. We see from (2) that the optimum value of d_c is shifted from the value $d/2$ by an amount that depends on the startup time for the daxpy operation, the computation time for the pivot search operation, and the amount of row communication that cannot be overlapped with computation, but this shift is relatively small on the nCUBE 2 because the constants involved are of similar magnitudes.

One way to improve the performance of the code for large d_r would be to use the BLAS to update rows (in the outer product update) rather than columns. This would have the effect of increasing the vector lengths for the BLAS calls and reducing the number of startups. We did not incorporate this improvement into the code because the optimum performance occurs

for $d_r < d/2$ and would not be affected by the switch. This improvement will be discussed in more detail in §5.2 dealing with QR factorization.

As parallel machines are built with more and more processors, it becomes possible to solve larger and larger problems. An important metric of parallel algorithms is how they scale to larger problems on more processors [20]. We will assume the amount of memory available to a processor remains constant, so the largest dense matrix that can be stored on a machine has $n^2 = cp$ for some constant c. We define *scaled speedup* to be the ratio of computation rate to the single processor computation rate for problems in which the number of matrix elements per processor remains constant. We note that this definition of scaled speedup is based on holding the memory requirements on each processor constant, not the workload on each processor. Because the LU factorization of a dense matrix requires $O(n^3)$ flops but only $O(n^2)$ storage, holding the workload per processor constant would result in lower scaled speedup values.

The results of a set of runs to determine scaled speedups are presented in Table 2. The first column of the table gives the linear dimension of the square matrix and the second the dimension of the hypercube. The third column contains the values of d_c and d_r that proved optimal. The fourth column shows the observed total number of Mflops per second of execution time, where a flop count of $2n^3/3$ is used. The fifth column divides the fourth by the number of processors. Scaled speedups are presented in the sixth column. The last column presents *efficiencies*, which we define to be the scaled speedup divided by the number of processors. We note that the efficiencies greater than one in the second and third rows of the table are due to the fact that the BLAS operations have greater efficiency with longer vectors.

TABLE 2
Times for LU *factorizations of scaled matrices.*

Matrix size	Cube dim.	$d_r : d_c$	Mflop/s observed	Mflop/s per proc.	Scaled speedup	Eff.
500	0	0 : 0	1.94	1.94	1.00	1.00
707	1	0 : 1	3.94	1.97	2.03	1.01
1,000	2	0 : 2	7.84	1.96	4.04	1.01
1,414	3	1 : 2	15.41	1.93	7.94	0.99
2,000	4	1 : 3	30.78	1.92	15.87	0.99
2,828	5	2 : 3	60.99	1.91	31.44	0.98
4,000	6	2 : 4	121.95	1.91	62.86	0.98
5,657	7	3 : 4	242.40	1.89	124.95	0.97
8,000	8	3 : 5	483.50	1.89	249.23	0.97
11,314	9	4 : 5	963.50	1.88	496.65	0.97
16,000	10	4 : 6	1917.19	1.87	988.24	0.96

In §4.1, we discussed the complexity of the communication volume and concluded that use of the torus-wrap mapping results in better scaling of an algorithm than either the row-wrap or the column-wrap mapping. This is born out by Fig. 6, where the performance of row-wrap and column-wrap mappings are compared to the optimal torus-wrap mapping for these scaled problems.

5.2. QR factorization. Our second example is Householder QR factorization without pivoting of an $m \times n$ matrix A. After LU, QR is probably the most important factorization in linear algebra and is used in least squares problems, eigenproblems, basis generation, and other settings. The total flop count for this algorithm is $2n^2(m - n/3) + O(n^2)$ for the usual situation in which $m \geq n$ [19]. Our parallel algorithm is outlined in Fig. 7 and is described in greater detail in [22]. Alternative QR algorithms that use the torus-wrap mapping can be found in [8], [37].

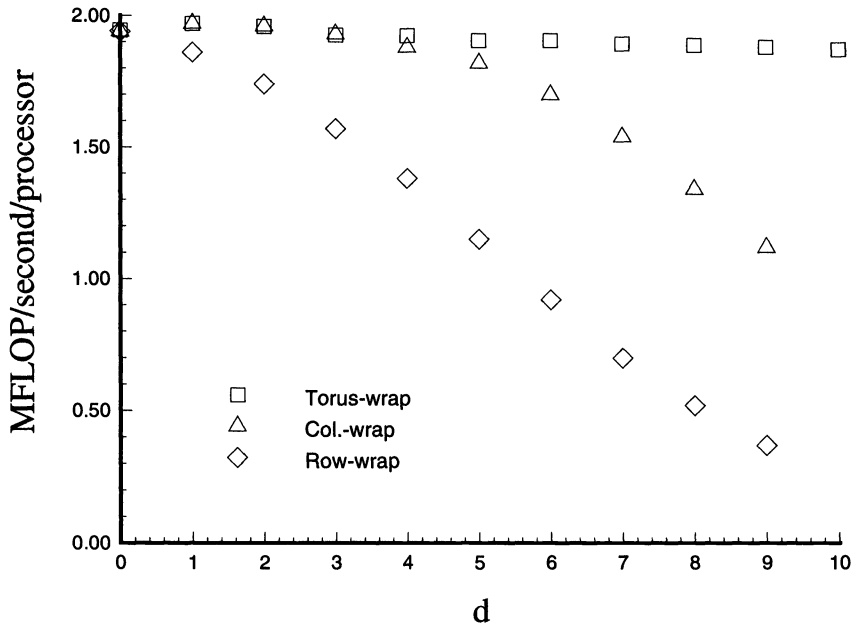


FIG. 6. *Mflop/s/processor rates for* LU *factorization with different processor mappings.*

Processor q owns row set α and column set β
For $j = 1$ to n
    ($*$ Generate Householder vector, v, from column j of A $*$)
    If $j \in \beta$ **Then**
        ($*$ Compute contribution to norm of column j $*$)
        $\gamma^q := A_{\alpha,j}^T A_{\alpha,j}$
        Binary collapse $\gamma = \sum \gamma^q$ to processor owning $A_{j,j}$
        If $j \in \alpha$ **Then**
            $\tau = 2(\gamma + |A_{j,j}|\sqrt{\gamma})$
            $A_{j,j} := A_{j,j} + \text{sign}(A_{j,j})\sqrt{\gamma}$
        $v_\alpha := A_{\alpha,j}$
        Broadcast v_α (and τ) to processors sharing rows α
    Else
        Receive v_α (and τ)

    $\beta := \beta \setminus \{j\}$ ($*$ Remove j from active columns $*$)

    $r_\beta^q := v_\alpha^T A_{\alpha,\beta}$ ($*$ Compute portion of dot–products $*$)
    Binary exchange (with appended τ) among processors sharing
        columns β to compute $r_\beta := \sum r_\beta^q$
    $A_{\alpha,\beta} := A_{\alpha,\beta} - (2/\tau) v_\alpha r_\beta$ ($*$ Update the submatrix $*$)

    $\alpha := \alpha \setminus \{j\}$ ($*$ Remove j from active rows $*$)

FIG. 7. *Parallel Householder* QR *for processor* q.

As with the LU implementation described in §5.1, to minimize time spent waiting for a Householder vector to be computed and broadcast, the processors that share the next column to be processed generate and broadcast their elements of the Householder vector before updating

the rest of their matrix elements. Run times for the same set of problem sizes considered in §5.1 are presented in Table 3.

TABLE 3
Run times on the nCUBE 2 *for* QR *factorization of an* 8,000 × 8,000 *matrix.*

256 Processors			512 Processors			1,024 Processors		
p_c	p_r	Seconds	p_c	p_r	Seconds	p_c	p_r	Seconds
1	256	3105	1	512	2612	1	1,024	2515
2	128	2087	2	256	1532	2	512	1324
4	64	1643	4	128	1042	4	256	779
8	32	1463	8	64	834	8	128	537
16	16	1428	16	32	756	16	64	436
32	8	1381	32	16	729	32	32	408
64	4	1384	64	8	713	64	16	381
128	2	1437	128	4	736	128	8	383
256	1	1559	256	2	814	256	4	418
			512	1	996	512	2	507
						1,024	1	715

The total run time for this algorithm can be modeled as the sum of the times for computing Householder vectors, broadcasting Householder vectors, generating inner-products and performing outer product updates. In addition to those parameters from §5.1, our model requires the following values from the nCUBE 2 manuals.

Variable	Description	Microseconds
$T_{n,a}$	startup time for a 2-norm	13.0
$T_{n,b}$	computational time for 2-norm (per element)	0.863

To generate a Householder vector the processor owning the top element must know the vector's 2-norm. The total time spent performing this task consists of the time for the local computations followed by the time for a binary collapse to combine the partial sums, which can be approximated by

$$T_{\text{norm}} = nT_{n,a} + \frac{n(2m-n)}{2p_r} T_{n,b} + d_r n(T_{c,a} + T_{c,b}).$$

The total time spent broadcasting Householder vectors can be modeled as

$$T_{\text{bcast}} = d_c n T_{c,a} + \frac{n(2m-n)}{2p_r} T_{c,b}.$$

As with the LU factorization column broadcast, the generation of the next Householder vector overlaps with all but the first stage of the broadcast of the current one. Thus the factor d_c does not appear in the per element term of the broadcast.

The time for computing all the dot-products is the sum of the time required for the numerics, $T_{\text{dot-calc}}$, and the time for combining all the partial sums within each column $T_{\text{dot-comm}}$. Although for hypercubes there are asymptotically more efficient alternatives [15], [34], we perform this communication using a binary exchange that requires redundant numerical operations. The gains from the more sophisticated algorithms are of a low order and so insignificant for large problems, and by not exploiting hypercube specifics we can reach broader conclusions about the performance of torus-wrap algorithms.

The numerical operations associated with the dot-products can be performed as a daxpy within rows, or as a ddot within columns (which has about the same startup and per element

cost as a daxpy). Our implementation dynamically chooses whichever of these operations involves fewer startups. For simplicity of analysis, we ignore the fact that the optimal choice can change during a run for nonsquare matrices, in which case the dot-product terms are about

$$T_{\text{dot-calc}} = \frac{n}{2}\min\left(\frac{n}{p_c}, \frac{2m-n}{p_r}\right)T_{d,a} + \frac{n^2(m-n/3)}{2p}T_{d,b},$$

$$T_{\text{dot-comm}} = d_r n T_{c,a} + \frac{d_r n^2}{2p_c}T_{c,b}.$$

Finally, the outer product updates on submatrices can be performed using daxpys within either columns or rows. We again choose whichever leads to fewer startups, so the total update time can be approximated as

$$T_{\text{update}} = \frac{n}{2}\min\left(\frac{n}{p_c}, \frac{2m-n}{p_r}\right)T_{d,a} + \frac{n^2(m-n/3)}{2p}T_{d,b}.$$

The data from Table 3 is shown graphically in Fig. 8, where the vertical axis is the Mflop rate achieved. To compute rates, we used the sequential flop count, which for this problem is about 6.83×10^{11} flops.

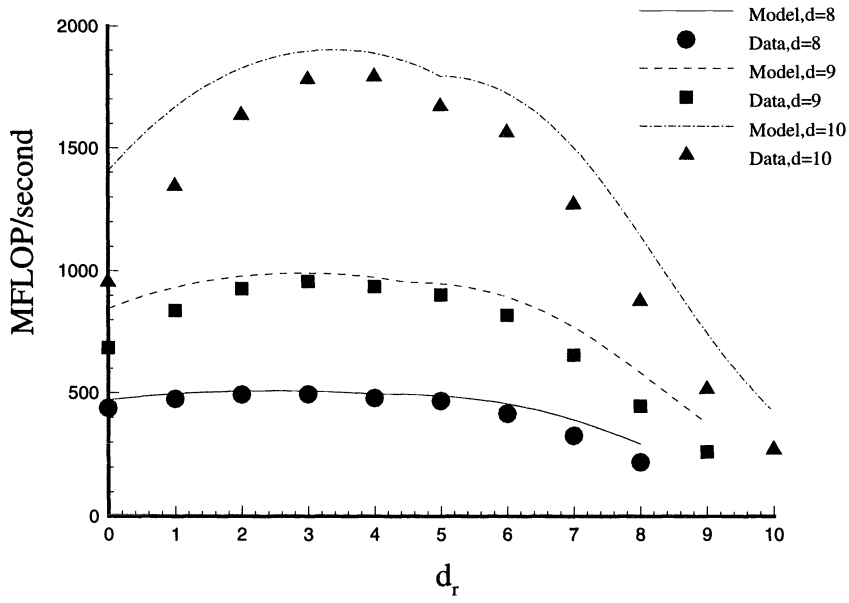


FIG. 8. *Performance of the* QR *factorization code on the* nCUBE 2.

The cusp in the curves is due to switching between a row oriented and a column oriented application of the level one BLAS. We expect the model to overpredict performance since by only including communication and BLAS it neglects any overheads. In the extreme case of column-wrapping, the model clearly overestimates the overlap of computation with communication. The model can be used to predict the optimal decomposition of processors among rows and columns by setting the derivative of the expression for run time to zero. However, as in §5.1, no-closed form expression results. The peak performance is achieved when d_r is somewhat less than d_c due primarily to the greater communication within columns than within rows. A nonsquare torus also allows for fewer BLAS startups.

TABLE 4
Times for QR *factorizations of square matrices.*

Matrix size	Cube Dim.	$d_r : d_c$	Mflop/s observed	Mflop/s per proc.	Scaled speedup	Eff.
500	0	0 : 0	1.97	1.97	1.00	1.00
707	1	0 : 1	3.98	1.99	2.02	1.01
1,000	2	0 : 2	7.95	1.99	4.04	1.01
1,414	3	0 : 3	15.83	1.98	8.05	1.00
2,000	4	1 : 3	31.42	1.96	15.97	1.00
2,828	5	1 : 4	62.76	1.96	31.90	1.00
4,000	6	2 : 4	124.56	1.95	63.31	0.99
5,657	7	2 : 5	249.06	1.95	126.59	0.99
8,000	8	3 : 5	494.32	1.93	251.24	0.98
11,314	9	3 : 6	989.71	1.93	503.03	0.98
16,000	10	3 : 7	1963.91	1.92	998.01	0.97

As in §5.1, we ran a sequence of factorizations in which the memory required per processor remained constant, allowing us to compute scaled speedups. The results are presented in Table 4. As before, the efficiencies greater than one are due to longer vectors in the BLAS routines.

Computation rates per processor for this set of problems are plotted in Fig. 9, comparing the optimal torus-wrap to row- and column-wrap mappings. As with LU, the torus-wrap mapping allows for much greater scalability than row or column schemes.

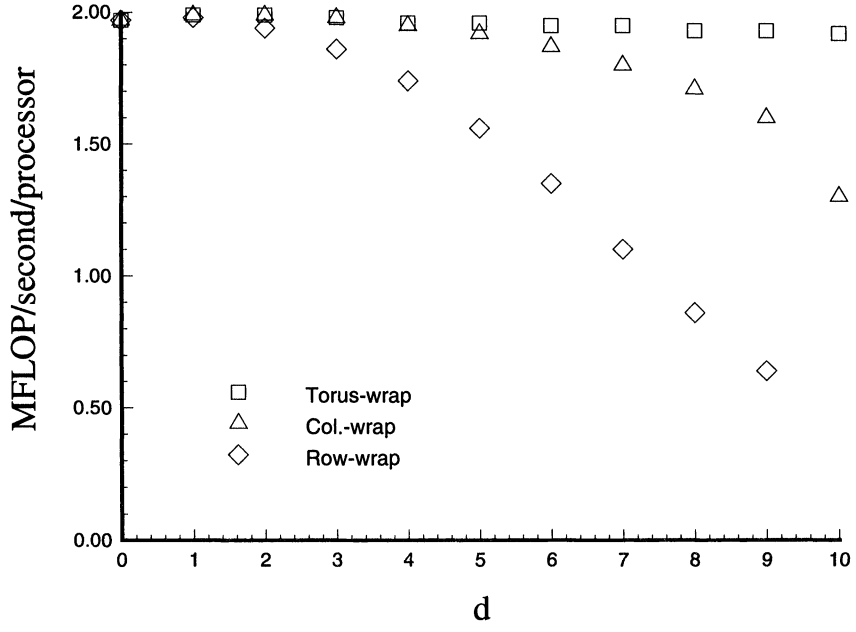


FIG. 9. *Mflop/s/processor rates for* QR *factorization with different processor mappings.*

5.3. Householder tridiagonalization. Our third example, Householder tridiagonalization of a symmetric $n \times n$ matrix A, is used in eigenvector calculations of symmetric or Hermitian matrices. The best sequential algorithm requires $4n^3/3 + O(n^2)$ flops and $n^2/2 + O(n)$ storage [19]. An algorithm that fails to exploit the symmetry of the matrix will require ad-

ditional computation and memory, so it is better to store and manipulate only a lower (or upper) triangular part of the matrix. This causes problems for row or column oriented mappings since a particular row (or column) of the matrix is now stored partially as a row and partially as a column. The same issues arise with other algorithms for symmetric matrices, like Cholesky decomposition. However, since a square torus-wrap mapping treats rows and columns symmetrically, it is well suited to deal with this problem.

Our algorithm for square tori is outlined in Fig. 10. Each processor stores its portion of the lower triangular part of A in a column major format. One property of torus-wrap mappings as we have defined them is that for square tori the processors owning diagonal matrix elements are assigned a set of row indices equal to their set of column indices. This is also true for block torus-wrap mappings with square blocks, but it is not generally true of Cartesian mappings. This property makes transposing vectors easy, since these diagonal processors have the appropriate elements of both a vector and its transpose. We exploit this convenience in our implementation, which limits us to square tori. A similar algorithm is described in [7].

As evidenced by Fig. 10, exploiting symmetry makes this algorithm more complicated than those for LU and QR. First a Householder vector v must be generated and broadcast. The appropriate elements of the transpose of v (and later w) are then communicated to each processor. Next comes the calculation of $s = \bar{A}v$, where $\bar{A}$ is the remaining submatrix. This is followed by the computation of $w = 2(s - (v^T s)v/\tau)/\tau$, where $\tau = ||v||^2$. Finally, an outer product update of the matrix elements is performed.

We implemented this algorithm on the nCUBE 2 and used the code to factor an $8,000 \times 8,000$, double precision matrix, which required 1653 and 556 seconds on cubes of dimension eight and ten, respectively.

The algorithm in Fig. 10 can be generalized to apply to nonsquare torus mappings, but this involves substantial complexity in performing the transpose operations and the matrix-vector multiplications. Although we did not implement this more general algorithm, we can develop a performance model to investigate the tradeoffs associated with different mappings. Our model will require the parameters introduced in the previous two sections and the following values from the nCUBE 2 manuals.

Variable	Description	Microseconds
$T_{s,a}$	startup time for a dscal	7.8
$T_{s,b}$	computational time for dscal (per element)	0.554
$T_{t,a}$	startup time for a ddot	10.0
$T_{t,b}$	computational time for ddot (per element)	0.984

Computing and broadcasting the Householder vectors is very similar to the operation required in Householder QR as described in §5.2.

$$T_{\text{norm}} = nT_{n,a} + \frac{n^2}{2p_r}T_{n,b} + d_r n(T_{c,a} + T_{c,b}),$$

$$T_{\text{bcast}} = d_c nT_{c,a} + \frac{d_c n^2}{2p_r}T_{c,b}.$$

Unlike the models for LU and QR, we include the time for each stage of the broadcast. This is because the binary exchanges in the algorithm keep the processors tightly synchronized, which reduces the potential to overlap computation with communication.

Processor q owns row set α and column set β of lower triangular A
For $j = 1$ to $n - 1$
  $\alpha := \alpha \setminus \{j\}$ (* Remove j from active rows *)

  (* Generate Householder vector, v, from column j of A *)
  If $j \in \beta$ **Then**
    $\gamma^q := A_{\alpha,j}^T A_{\alpha,j}$
    Binary collapse $\gamma = \sum \gamma^q$ to processor owning $A_{j+1,j}$
    If $j + 1 \in \alpha$ **Then**
      $\tau = 2(\gamma + |A_{j+1,j}|\sqrt{\gamma})$
      $A_{j+1,j} := A_{j+1,j} + \text{sign}(A_{j+1,j})\sqrt{\gamma}$
    $v_\alpha := A_{\alpha,j}$
    Broadcast v_α (and τ) to processors sharing rows α
  Else Receive v_α (and τ)

  $\beta := \beta \setminus \{j\}$ (* Remove j from active columns *)

  (* Get elements of v^T to the correct processors *)
  If $\alpha = \beta$ **Then**
    $v_\beta := v_\alpha$
    Broadcast v_β to processors sharing columns β
  Else Receive v_β

  (* Compute Av *)
  $r_\beta^q := A_{\alpha,\beta}^T v_\alpha$
  Binary collapse among processors sharing columns β
    to diagonal processor to form $r_\beta := \sum r_\beta^q$
  If $\alpha = \beta$ **Then**
    $s_\alpha^q := r_\alpha + A_{\alpha,\beta} v_\beta$ (* excluding diagonal contribution *)
  Else $s_\alpha^q := A_{\alpha,\beta} v_\beta$
  Binary exchange among processors sharing rows α
    to form $s_\alpha := \sum s_\alpha^q$
  (* Compute $v^T s$ and generate w *)
  $\eta^q := \sum_{i \in \alpha} v_i s_i$
  Binary exchange among processors sharing columns β
    to form $\eta := \sum \eta^q$ (and append τ)
  $w_\alpha := \frac{2}{\tau}(s_\alpha - \frac{\eta}{\tau} v_\alpha)$

  (* Get elements of w^T to the correct processors *)
  If $\alpha = \beta$ **Then**
    $w_\beta := w_\alpha$
    Broadcast w_β to processors sharing columns β
  Else Receive w_β

  $A_{\alpha,\beta} := A_{\alpha,\beta} - v_\alpha w_\beta^T - w_\alpha v_\beta^T$ (* Update the submatrix *)

FIG. 10. *Parallel Householder tridiagonalization for processor q.*

For square tori, transposing v and w just requires broadcasts from the diagonal processors, but for nonsquare tori it is more complicated. We denote $p_{\max} = \max(p_r, p_c)$, and $d_{\max} = \max(d_r, d_c)$, with the obvious $p_{\min}$ and $d_{\min}$ counterparts. Transposition can be accomplished with $d_{\min}$ stages of a broadcast with message length about $n/p_{\max}$, followed by $d_{\max} - d_c$ stages of a binary exchange in which the message length doubles after each stage. We note that for nonsquare tori, some copying of data is also required. The total time spent performing these operations is about

$$T_{\text{trans}} = 2d_r n T_{c,a} + \frac{d_{\min} n^2}{p_{\max}} T_{c,b} + \frac{n^2}{p_{\max}} \left(\frac{p_{\max}}{p_c} - 1\right) T_{c,b}.$$

This formula is assymetric in rows and columns because the recursive doubling stage in the transpose need only occur if $d_r > d_c$. Otherwise, the last term in the expression reduces to zero.

Computing $\bar{A}v$ is somewhat problematic since only the lower triangular portion of the matrix is stored. We denote this triangular portion as L_1, and the portion of L_1 below the diagonal as L_2. We observe that $\bar{A}v = L_1^T v + L_2 v$, which requires communication in both rows and columns. Our algorithm first performs a ddot to determine the contribution from $L_1^T v$. These values are combined and sent to the processors owning the diagonal matrix elements using precisely the opposite of the communication pattern used above for transposition. Next, the contribution from $L_2 v$ is computed using a daxpy, and these values are combined across rows using a binary exchange. As a side effect, the binary exchange synchronizes the processors within each row and a total of about $(p_c - 1)n^2/2$ redundant flops are performed. As with our implementation of QR factorization, for hypercubes there are asymptotically more efficient alternatives to the binary exchange [15], [34], but our implementation does not use them. The calculation and communication time for this operation can be approximated as

$$T_{\bar{A}v-\text{calc}} = \frac{n^2}{2p_c}(T_{t,a} + T_{d,a}) + \frac{n^3}{6p}(T_{t,b} + T_{d,b}),$$

$$T_{\bar{A}v-\text{comm}} = (d_r + d_c) n T_{c,a} + \frac{n^2}{2p_{\max}}\left(d_{\min} + \frac{p_{\max}}{p_c} - 1\right) T_{c,b} + \frac{d_c n^2}{2p_r} T_{c,b}.$$

Forming $v^T s$ involves local computation, followed by a binary exchange among the processors sharing a set of column indices. This serves to synchronize the processors within each column, and requires a total of about $(p_r - 1)n^2/2$ redundant flops. Also, the same local computations are repeated by each column of processors, implying an overall additional $(p_c - 1)n^2$ flops beyond those in the sequential algorithm. The time for this computation and communication can be modeled as

$$T_{\text{dot-calc}} = n T_{t,a} + \frac{n^2}{2p_r} T_{t,b}$$

$$T_{\text{dot-comm}} = d_r n (T_{c,a} + T_{c,b}).$$

Each processor can now generate its own elements of w, using a dscal followed by a daxpy. As above, this calculation is duplicated p_c times, resulting in about $2(p_c - 1)n^2$ extra flops. The time spent in this step is about

$$T_{\text{genw}} = n(T_{d,a} + T_{s,a}) + \frac{n^2}{2p_r}(T_{d,b} + T_{s,b}).$$

Updating $\bar{A}$ by $-vw^T - wv^T$ is now a local operation performed by each processor on its own data. Each column of $\bar{A}$ can be updated with two daxpys, so the time for this operation can be modeled as

$$T_{\text{update}} = \frac{n^2}{p_c} T_{d,a} + \frac{n^3}{3p} T_{d,b}.$$

The total time is modeled as the sum of the terms above. The predictions of the model for an $8,000 \times 8,000$ matrix are plotted in Fig. 11, with the observed values for square tori

included for comparison. In computing rates, we use the sequential flop count, which is about 6.83×10^{11} for this problem.

As expected, the model indicates that a square torus is better than row or column methods for this problem. The cusp in the model is due to the different communication patterns that apply depending on the relative sizes of p_r and p_c. We note that for these problems, the model predicts a slight improvement in performance when $d_r = (d/2) - 1$, but the model neglects the additional copying required for nonsquare tori. We can use the model to predict the optimal tradeoff between p_r and p_c in general, but as in §5.1 and §5.2 no closed form expression exists.

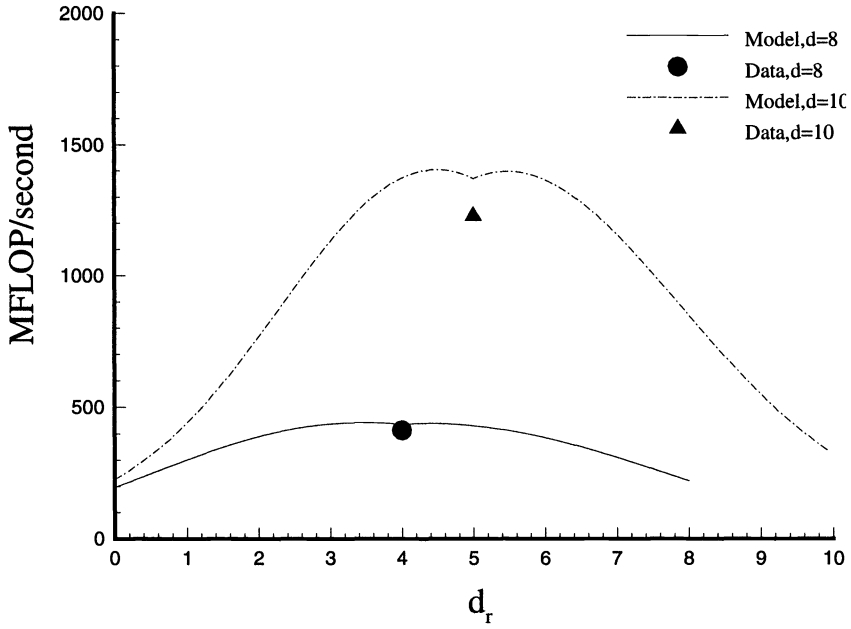


FIG. 11. *Performance of the tridiagonalization code on the* nCUBE 2.

As in §5.1 and §5.2, to investigate scaled speedup we ran the code on problems in which the local memory requirements remained constant. The results are presented in Table 5.

TABLE 5
Times for Householder tridiagonalization.

Matrix size	Cube dim.	Mflop/s observed	Mflop/s per proc.	Scaled speedup	Eff.
500	0	1.87	1.87	1.00	1.00
1,000	2	7.25	1.81	3.88	0.97
2,000	4	27.84	1.74	14.89	0.93
4,000	6	107.30	1.68	57.40	0.90
8,000	8	413.01	1.61	220.95	0.86
16,000	10	1596.33	1.56	853.97	0.83

On a single processor, the performance of the tridiagonalization code is about 5% less than that for LU or QR. This is a consequence of exploiting the symmetry of the matrix and can be expected in other algorithms that work on symmetric matrices like Cholesky factorization. The short columns in the rightmost portion of the lower triangular matrix and the short rows at

the top result in many short vectors in the BLAS. Also, index calculations are more complex with a triangular matrix, adding some overhead to the calculation.

In addition, the performance of the tridiagonalization code scales less well than either LU or QR. The efficiency on 1,024 processors is about 83%, while for QR and LU it was in the upper 90s. This is a consequence of three factors. First, the repeated computations add about $n^2(7p_c + p_r)/2$ flops to the sequential algorithm. Second, exploiting symmetry requires a greater amount of communication. Third, the two binary exchanges effectively synchronize the processors, which reduces the potential for hiding communication with computation. The second factor will also influence other algorithms on symmetric matrices. Having said this, it is still true that the tridiagonalization code performs well, achieving greater than 75% of the peak BLAS performance on 1024 processors.

6. Conclusions. We have presented analytical and empirical evidence that for many dense linear algebra algorithms, the torus-wrap mapping is better than row or column mappings. The primary advantage of the torus-wrap is that it requires less communication, leading to better scalability, but there are a number of additional advantages including better load balancing, reduced processor idle time, and shorter message queues.

After factoring a matrix, one typically wishes to use it, for example, to solve linear systems or least squares problems. This requires using the factored products to modify one or more vectors. If only a few vectors are involved, then the flop count is $\Theta(n^2)$, an order of magnitude less than the factorization. In this case, since the cost of the factorization dominates, a torus-wrap mapping for the factorization is likely to give the best overall performance. In addition, algorithms exist for doing a single triangular solve using the torus-wrap mapping that run in the asymptotically optimal time of $n^2/p + O(n)$ [5], [28].

If many vectors must be modified, then they can be combined to form a matrix that can be assigned to processors in a torus-wrap fashion. The same techniques that were employed in the factorization can now be used in the triangular solves, and good performance should result. For instance, the LU factorization code described in §5.1 has been used to invert a matrix by solving n linear equations on the nCUBE 2 at an overall computational rate of 1.96 Gflop/s.

Acknowledgments. We are indebted to Cleve Ashcraft and Andy Cleary for bringing the torus-wrap mapping to our attention, to Courtenay Vaughan for help with the implementations, and to Ernie Brickell for useful discussions. We also wish to acknowledge the extremely conscientious effort on the part of an anonymous referee whose suggestions significantly improved this paper.

REFERENCES

[1] C. C. ASHCRAFT, *The Distributed Solution of Linear Systems Using the Torus Wrap Data Mapping*, Tech. Report ECA-TR-147, Boeing Computer Services, Seattle, WA, October 1990.

[2] ———, *A taxonomy of distributed dense LU factorization methods*, Tech. Report ECA-TR-161, Boeing Computer Services, Seattle, WA, March 1991.

[3] R. H. BISSELING AND L. D. J. C. LOYENS, *Towards Peak Parallel* LINPACK *Performance on* 400 *Transputers*, Supercomputer, 45 (1991), pp. 20–27.

[4] R. H. BISSELING AND J. G. G. VAN DE VORST, *Parallel* LU *decomposition on a transputer network*, in Lecture Notes in Comput. Sci. No. 384, G. A. van Zee and J. G. G. van de Vorst, eds., Springer-Verlag, New York, Berlin, 1989, pp. 61–77.

[5] ———, *Parallel triangular system solving on a mesh network of transputers*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 787–799.

[6] R. P. BRENT, *The* LINPACK *benchmark on the* AP 1000 in Proc. Frontiers 1992, McLean, VA, October 1992, pp. 128–135.

[7] H. Y. Chang, S. Utku, M. Salama, and D. Rapp, *A parallel Householder tridiagonalization stratagem using scattered square decomposition*, Parallel Comput., 6 (1988), pp. 297–311.

[8] E. Chu and A. George, *QR factorization of a dense matrix on a hypercube multiprocessor*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 990–1028.

[9] J. J. Dongarra, *Performance of various computers using standard linear equations software*, Supercomputing Rev., 4 (1991), pp. 45–54.

[10] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff, *A set of level 3 basic linear algebra subprograms*, TOMS, 16 (1990), pp. 1–17.

[11] J. J. Dongarra, I. S. Duff, D. C. 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.

[12] J. J. Dongarra, A. H. Sameh, and D. C. Sorensen, *Implementation of some concurrent algorithms for matrix factorization*, Parallel Comput., 3 (1986), pp. 25–34.

[13] J. J. Dongarra, D. Walker, and R. van de Geijn, *A look at scalable dense linear algebra libraries*, in Proc. Scalable High Performance Computing Conf., Williamsburg, VA, April 1992, pp. 372–379.

[14] G. C. Fox, *Square Matrix Decomposition – Symmetric, Local, Scattered*, Tech. Report, CalTech Publication Hm-97, California Institute of Technology, Pasadena, CA, 1985.

[15] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker, *Solving problems on concurrent processors: Volume* 1, Prentice Hall, Englewood Cliffs, NJ, 1988.

[16] K. A. Gallivan, M. T. Heath, E. Ng, J. M. Ortega, B. W. Peyton, R. J. Plemmons, C. H. Romine, A. H. Sameh, and R. G. Voigt, *Parallel Algorithms for Matrix Computations*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1990.

[17] G. A. Geist and M. T. Heath, *Matrix factorization on a hypercube multiprocessor*, in Hypercube Processors, 1986, M. T. Heath, ed., Society for Industrial and Applied Mathematics, 1986, pp. 161–180.

[18] G. A. Geist and C. H. Romine, *LU factorization algorithms on distributed-memory multiprocessor architectures*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 639–649.

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

[20] J. L. Gustafson, G. R. Montry, and R. E. Benner, *Development of parallel methods for a 1024–processor hypercube*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 609–638.

[21] M. T. Heath and C. H. Romine, *Parallel solution of triangular systems on distributed–memory multiprocessors*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 558–588.

[22] B. Hendrickson, *Parallel QR Factorization on a Hypercube Using the Torus Wrap Mapping*, Tech. Report SAND91-0874, Sandia National Laboratories, Albuquerque, NM, October 1991.

[23] S. L. Johnsson, *Communication efficient basic linear algebra computations on hypercube architectures*, J. Par. Distr. Comput., 4 (1987), pp. 133–172.

[24] S. L. Johnsson and C. T. Ho, *Optimum broadcasting and personalized communication in hypercubes*, IEEE Trans. Computers, 38 (1989), pp. 1249–1268.

[25] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic linear algebra subprograms for fortran usage*, TOMS, 5 (1979), pp. 308–323.

[26] G. Li and T. F. Coleman, *A parallel triangular solver for a distributed–memory multiprocessor*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 485–502.

[27] W. Lichtenstein and S. L. Johnsson, *Block cyclic dense linear algebra*, SIAM J. Sci. Comput., 14 (1993), pp. 1257–1286.

[28] L. D. J. C. Loyens and R. H. Bisseling, *The formal construction of a parallel triangular system solver*, in Lecture Notes in Comput. Sci. No. 375, J. van de Snepscheut, ed., Springer-Verlag, New York, Berlin, 1989, pp. 325–334.

[29] D. P. O'Leary and G. W. Stewart, *Data-flow algorithms for parallel matrix computations*, Comm. ACM, 28 (1985), pp. 840–853.

[30] ———, *Assignment and scheduling in parallel matrix factorization*, Linear Algebra Appl., 77 (1986), pp. 275–299.

[31] J. M. Ortega and C. H. Romine, *The ijk forms of factorization methods* ii. *parallel systems*, Parallel Comput. 7 (1988), pp. 149–162.

[32] Y. Saad, *Communication complexity of the Gaussian elimination algorithm on multiprocessors*, Linear Algebra Appl., 77 (1986), pp. 315–340.

[33] G. W. Stewart, *Communication and matrix computations on large message passing systems*, Parallel Comput., 16 (1990), pp. 27–40.

[34] R. van de Geijn, *Efficient global combine operations*, in Proc. 6th Distributed Memory Computing Conf., IEEE Computer Society Press, 1991, pp. 291–294.

[35] R. VAN DE GEIJN, *Massively Parallel* LINPACK *Benchmark on the Intel Touchstone Delta and iPSC/860 Systems*, Tech. Report Computer Science Report TR–91–28, University of Texas, Austin, TX, 1991.

[36] J. G. G. VAN DE VORST, *The formal development of a parallel program performing* LU*-decomposition*, Acta Inform., 26 (1988), pp. 1–17.

[37] E. L. ZAPATA, J. A. LAMAS, F. F. RIVERA, AND O. G. PLATA, *Modified Gram-Schmidt* QR *factorization on hypercube* SIMD *computers*, J. Par. Distr. Comput., 12 (1991), pp. 60–69.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1227–1242, September 1994

015

ERRORS WHEN SHOCK WAVES INTERACT DUE TO NUMERICAL SHOCK WIDTH*

RALPH MENIKOFF†

Abstract. A simple test problem proposed by Noh, a cold gas with uniform initial particle velocity directed towards a rigid wall, demonstrates a generic problem with numerical shock capturing algorithms at boundaries that Noh called "excess wall heating." The same type of numerical error is shown to occur when shock waves interact. The underlying cause is due to the numerical shock profile. The error can be understood from an analysis of the asymptotic solution of the partial differential equations when an artificial viscosity is added. The position of the front for a numerical shock wave can be defined by matching the total mass in the profile to that of a discontinuous shock. There is then a difference in the total energy of the numerical wave relative to a discontinuous shock. Moreover, the relative energy depends on the strength of the shock. The error when shock waves interact results from the difference in the relative energies between the incoming and outgoing shock waves. A conservative differencing scheme correctly describes the Hugoniot jump conditions for a steady propagating shock. The error implied by the asymptotic energy shift occurs in the entropy generated by the numerical dissipation in the transient when the waves interact. The entropy error remains localized and does not dissipate. A scaling argument shows that as the viscosity coefficient approaches zero, the error shrinks in spatial extent but the peak pointwise error is constant in magnitude. Consequently, the convergence of the inviscid limit to the hyperbolic solution is nonuniform in regions where shocks have interacted.

Key words. hyperbolic conservation laws, shock interactions, viscous profiles

AMS subject classifications. 35L65, 35L67, 65M12, 76M20

1. Introduction.

The equations for ideal fluid flow form a hyperbolic system of conservation laws

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ \rho(\frac{1}{2}u^2 + E) \end{pmatrix} + \partial_x \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho(\frac{1}{2}u^2 + E)u + Pu \end{pmatrix} = \vec{0}, \tag{1.1}$$

where ρ is the density, u is the particle velocity, E is the specific energy, $P(V, E)$ is the pressure, and $V = 1/\rho$ is the specific volume. Dissipation only occurs across a shock wave and physically is accounted for by imposing the Rankine–Hugoniot jump relations across the shock discontinuity. Finite difference shock-capturing algorithms are frequently used to obtain a numerical solution to the fluid flow equations. These schemes have a numerical dissipation that gives a shock wave a small width measured in grid cells, but an artificially large spatial width compared to the typical shock width that physically occurs. The effect of the artificial shock width is largest during a transient when a shock wave forms or when shock waves interact. To determine the effect of the numerical shock width, we analyze the asymptotic solution for a simple shock interaction when a viscous dissipative term is added to the ideal fluid equations.

The problem we consider in detail is a strong shock in an ideal gas reflecting from a rigid wall. This is equivalent to the interaction between equal strength shocks of the opposite family. It is similar to a test problem Noh [6] introduced that exemplifies errors in numerical calculations due to artificial viscosity. In Noh's problem the initial data consists of a uniform state of cold gas with a constant velocity directed toward a rigid wall. Its solution has a strong outgoing shock. Because of the zero initial sound speed, an analytic solution exists in planar, cylindrical, and spherical geometry. Typically, numerical solutions have an entropy

*Received by the editors May 10, 1993; accepted for publication September 7, 1993. This work was supported by the U. S. Department of Energy.

†Theoretical Division, Los Alamos National Laboratory, Los Alamos, New Mexico 87545 (rtm@lanl.gov).

error at the boundary. The shock interaction problem considered here is less singular than the Noh problem. The initial state is assumed to have a smooth viscous profile rather than a discontinuity in the velocity. Furthermore, the Mach number of the outgoing shock is finite. Nevertheless, the same type of entropy error occurs in the numerical solution of the shock interaction problem.

The hyperbolic solution of the shock interaction problem consists of an outgoing shock wave. Because the flux at the boundaries is constant, the total mass, momentum, and energy in the viscous solution can be compared with those in the hyperbolic solution. We define the shock position of the viscous wave to have the same total mass and momentum as the hyperbolic shock wave. An important quantity in the asymptotic analysis is the energy of the viscous shock relative to the energy of hyperbolic shock.

We show that there is a shift in the relative energy between the incoming and outgoing waves. This implies that an entropy error must occur during the transient shock interaction. For the shock reflection problem, the transient takes place when the shock profile overlaps with the boundary. After the transient, the entropy is frozen in place, i.e., convects along particle trajectories, and the error does not dissipate.

A scaling argument due to Noh shows that under mesh refinement the entropy error decreases in spatial extent but the peak pointwise error is constant in magnitude. Since the coefficient of viscosity decreases with the mesh size, the scaling implies that the convergence of the inviscid limit to the hyperbolic solution is nonuniform in regions where shocks have interacted.

In particular, for a von Neumann–Richtmyer artificial viscosity the shift in the relative energies is calculated analytically. A numerical example illustrates how the transient shock interaction gives rise to an entropy error. From the general energy arguments we are led to conjecture that the numerical shock width in any shock-capturing algorithm without significant heat conduction will give rise to an entropy error when shock waves interact.

2. Asymptotics. When the fluid equations are regularized by a viscous dissipation term, a shock wave very rapidly approaches its asymptotic steady state profile. In fact this is a necessary condition for shock capturing algorithms to satisfy the Hugoniot jump conditions.

Let us consider a steady state viscous shock wave. Suppose the wave is right facing and propagating with velocity σ. Furthermore, let the reference points x_a and x_b be in the ahead and behind state, respectively, with $x_b < x_a$. The position of the wave can be defined by comparing the viscous profile with a discontinuous shock and adjusting the discontinuity such that the two waves have the same total mass.

The condition that the waves have the same mass is given by

$$0 = \int_{x_b}^{x_s} dx\,(\rho - \rho_b) + \int_{x_s}^{x_a} dx\,(\rho - \rho_a). \tag{2.1}$$

Relative to x_b the shock position based on the mass is given by

$$x_s = x_b + (\rho_b - \rho_a)^{-1} \int_{x_b}^{x_a} dx\,(\rho(x) - \rho_a). \tag{2.2}$$

Similarly, the position of the wave could be defined by matching the total momentum. The shock position based on momentum is obtained from (2.2) by replacing the mass density ρ with the momentum density ρu.

In steady state the mass flux is everywhere constant, $\rho(u - \sigma) = m$. Hence, there is a linear relation between mass density and momentum density $\rho u = \sigma\rho + m$. Consequently the shock position, based on either the mass or momentum of the wave, is the same.

We could also base the shock position on the total energy. However, the energy density $\mathcal{E} = (\frac{1}{2}u^2 + E)\rho$ is not Galilean invariant. This would lead to a nonuniqueness in the shock position. Instead, we define the relative energy between the viscous profile and the discontinuous shock with the shock position based on mass

$$\begin{aligned}\delta\mathcal{E}^T &= \int_{x_b}^{x_s} dx\,(\mathcal{E} - \mathcal{E}_b) + \int_{x_s}^{x_a} dx\,(\mathcal{E} - \mathcal{E}_a) \\ &= \int_{x_b}^{x_a} dx\,(\mathcal{E} - \mathcal{E}_a) - (x_s - x_b)\,(\mathcal{E}_b - \mathcal{E}_a).\end{aligned} \tag{2.3}$$

We note that $\delta\mathcal{E}^T > 0$ corresponds to an excess energy in the viscous profile over the discontinuous shock.

We next show that the relative energy is Galilean invariant and hence well defined. In a reference frame moving with relative velocity u' the energy density is transformed to $\mathcal{E}' = \mathcal{E} + \rho u u' + \frac{1}{2}\rho(u')^2$. Substituting $\mathcal{E}'$ into (2.3) we find that the additional terms are proportional to the mass and momentum density and have the same form as in (2.1). Hence, the additional terms vanish when x_s is chosen to be the shock position based on mass or equivalently momentum.

3. Von Neumann–Richtmyer viscosity. Viscosity can be incorporated into the planar one-dimensional (1-D) fluid equations by adding a viscous pressure to the fluid pressure, $P \to P + Q$ in (1.1). We analyze the viscous fluid equations using a von Neumann–Richtmyer viscosity [5] and an ideal gas equation of state. The von Neumann–Richtmyer viscosity is defined by the viscous pressure

$$Q = \begin{cases} C_v\,\rho\,\ell^2(\partial_x u)^2, & \text{if } \partial_x u < 0; \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

where C_v is a dimensionless viscosity and ℓ is a length scale proportional to the shock width. We note that Q is Galilean invariant. Without loss of generality we can set $C_v = 1$. For an ideal gas, with $\gamma > 1$, the equation of state is

$$PV = (\gamma - 1)E. \tag{3.2}$$

In this case, there is an exact analytic formula for the viscous profile of a shock wave. Let σ be the shock velocity and the variable

$$w = \left(\frac{\gamma + 1}{2}\right)^{1/2} \cdot \left(\frac{x - \sigma t}{\ell}\right) \tag{3.3}$$

be a scaled length relative to the shock front. The viscous profile is given by [7]

$$V(w) = \tfrac{1}{2}(V_a + V_b) + \tfrac{1}{2}(V_a - V_b)\sin(w), \tag{3.4}$$

$$P(w) = \tfrac{1}{2}(P_a + P_b) - \tfrac{1}{2}(P_b - P_a)\left[\sin(w) + \frac{\frac{1}{2}(\gamma + 1)\,(V_a - V_b)\cos^2(w)}{(V_a + V_b) + (V_a - V_b)\sin(w)}\right], \tag{3.5}$$

$$Q(w) = \tfrac{1}{4}\,(\gamma + 1)\,(P_b - P_a)\left[\frac{(V_a - V_b)\cos^2(w)}{(V_a + V_b) + (V_a - V_b)\sin(w)}\right], \tag{3.6}$$

$$u(w) = \sigma - m\,V(w), \tag{3.7}$$

where $m = \rho_a(\sigma - u_a)$ is the mass flux through the shock front. From the Hugoniot jump conditions $m^2 = (P_b - P_a)/(V_a - V_b)$. We note the shock profile is of finite width extending from the ahead state at $w_a = \frac{1}{2}\pi$ to the behind state at $w_b = -\frac{1}{2}\pi$.

The shock position based on mass is given by

$$w_s = w_b + (\rho_b - \rho_a)^{-1}\left[\int_{-\pi/2}^{\pi/2} dw\, \frac{2}{(V_a + V_b) + (V_a - V_b)\sin(w)} - \pi\rho_a\right]. \tag{3.8}$$

The integral is of the form evaluated in the appendix. It can be simplified to give

$$w_s = w_b + \frac{\pi}{\eta(\eta^{1/2} + 1)}, \tag{3.9}$$

where $\eta = V_a/V_b$ is the compression ratio of the shock. We note that the compression ratio is in the bounded interval $1 \le \eta \le (\gamma+1)/(\gamma-1)$. For the limiting case of a weak shock $\eta \to 1$ and $w_s \to 0$, and for a strong shock $\eta \to (\gamma+1)/(\gamma-1)$ and $w_s \ne 0$ with $-\frac{1}{2}\pi < w_s < 0$.

It is convenient to calculate the relative energy of the shock profile in the rest frame of the shock front, i.e., $\sigma = 0$. In this case the kinetic energy is $\frac{1}{2}\rho u^2 = \frac{1}{2}m^2 V$ and the energy density can be expressed as

$$\mathcal{E} = \tfrac{1}{2}\, m^2\, V + (\gamma - 1)^{-1} P. \tag{3.10}$$

Substituting this expression into (2.3) for the relative energy we obtain

$$\delta\mathcal{E}^T = \left(\frac{2}{\gamma+1}\right)^{1/2} \ell\left[\tfrac{1}{2}m^2 \int_{-\pi/2}^{\pi/2} dw\, V(w) + \frac{1}{\gamma-1}\int_{-\pi/2}^{\pi/2} dw\, P(w) - \pi\mathcal{E}_a - (w_s - w_b)(\mathcal{E}_b - \mathcal{E}_a)\right]. \tag{3.11}$$

The integrals can be evaluated with the formulae in the appendix

$$\int_{-\pi/2}^{\pi/2} dw\, V(w) = \tfrac{1}{2}\pi(V_a + V_b),$$

$$\int_{-\pi/2}^{\pi/2} dw\, P(w) = \tfrac{1}{2}\pi\left[(P_a + P_b) - \tfrac{1}{2}(\gamma+1)\left(\frac{\eta^{1/2}-1}{\eta^{1/2}+1}\right)(P_b - P_a)\right].$$

After straightforward algebraic manipulation, we obtain for the relative energy

$$\delta\mathcal{E}^T = \tfrac{1}{2}\pi\frac{1}{\gamma-1}\left(\frac{2}{\gamma+1}\right)^{1/2}\ell(P_b - P_a)\left\{(\gamma-3)\left[\frac{1}{\eta(\eta^{1/2}+1)} - \tfrac{1}{2}\right] - \tfrac{1}{2}(\gamma+1)\frac{\eta^{1/2}-1}{\eta(\eta^{1/2}+1)}\right\}. \tag{3.12}$$

We note that the relative energy is in general nonzero and has the following three properties:

(i) $\delta\mathcal{E}^T$ is a function of the shock width. In particular, $\delta\mathcal{E}^T \to 0$ as the shock width goes to zero.

(ii) $\delta\mathcal{E}^T$ is a function of the shock strength. For weak shocks $\delta\mathcal{E}^T/\ell \sim (P_b - P_a)^2$ and for strong shocks $\delta\mathcal{E}^T/\ell \sim (P_b - P_a)$. Moreover, $\delta\mathcal{E}^T \to 0$ as the shock strength goes to zero.

(iii) $\delta\mathcal{E}^T$ is a function of the equation of state (EOS). The relative energy depends on the EOS through the shock profile. Because the viscous pressure depends only on the density and velocity, the profile is expected to vary with the EOS. In the weak shock limit, $\eta \to 1$ and the factor in curly brackets goes to $(7-3\gamma)(\eta-1)/4$. Thus, depending on the EOS, the relative energy can have either sign. Numerical evaluation of (3.12) shows that $\delta\mathcal{E}^T > 0$ when $1 < \gamma < 2+$, $\delta\mathcal{E}^T < 0$ when $\gamma > 3-$, and the sign of $\delta\mathcal{E}^T$ depends on shock strength for intermediate values such as $\gamma = 2.5$.

These important properties are expected to be true for any reasonable viscosity.

4. Effect of shock width on a shock interaction. In the limit of weak shocks, the numerical shock width has a negligible effect because both the relative energy and the entropy change of a shock wave approach zero. As the strength of a shock wave increases, the shock width has an important effect on a shock interaction.

We consider the simple case of a strong shock reflecting from a rigid wall. Figure 1 shows the results of a numerical calculation, discussed later in more detail. It illustrates the effect described by Noh [6]. The pressure and particle velocity are in good agreement with the analytic solution but there are large errors localized near the wall in the density and specific energy. We explain below how this error is a consequence of the shift in the relative energy of the shock profiles resulting from the shock interaction.

To compare the viscous solution with the hyperbolic solution, we compute the difference in the relative energy between the incoming shock and the outgoing shock

$$\Delta\mathcal{E}^T = \delta\mathcal{E}^T_{rs} - \delta\mathcal{E}^T_{s}. \tag{4.1}$$

We note that $\Delta\mathcal{E}^T > 0$ corresponds to a net excess energy in the viscous shock profiles compared to the hyperbolic shocks. Let the pressure behind the incoming shock be P_s. The compression ratio of a strong shock is $\eta_s = (\gamma+1)/(\gamma-1)$. The reflected shock is characterized by its pressure ratio, $P_{rs}/P_s = 1 + 2\gamma/(\gamma-1)$, and its compression ratio, $\rho_{rs}/\rho_s = \gamma/(\gamma-1)$.

The scale for the relative energies is

$$\epsilon = \tfrac{1}{2}\pi \frac{1}{\gamma-1}\left(\frac{2}{\gamma+1}\right)^{1/2} \ell\, P_s.$$

Substituting the values for the pressure and compression ratio into (3.11) we obtain for the relative energies

$$\delta\mathcal{E}^T_s/\epsilon = \tfrac{1}{2}(\gamma-3)\left\{\frac{(\gamma-1)^2}{\gamma+1}\left[\left(\frac{\gamma+1}{\gamma-1}\right)^{1/2}-1\right]-1\right\} - \tfrac{1}{4}(\gamma-1)^2\left[\left(\frac{\gamma+1}{\gamma-1}\right)^{1/2}-1\right]^2,$$

$$\delta\mathcal{E}^T_{rs}/\epsilon = (\gamma-3)\left\{2(\gamma-1)\left[\left(\frac{\gamma}{\gamma-1}\right)^{1/2}-1\right]-\frac{\gamma}{\gamma-1}\right\} - (\gamma+1)(\gamma-1)\left[\left(\frac{\gamma}{\gamma-1}\right)^{1/2}-1\right]^2.$$

The above formulae determine $\Delta\mathcal{E}^T$ as a function of γ. The difference of the relative energy, shown in Fig. 2, has the following general properties:

(i) $\Delta\mathcal{E}^T \to \infty$ as $\gamma \to 1$. This singularity is due to the singularity in the compression ratio at $\gamma = 1$.

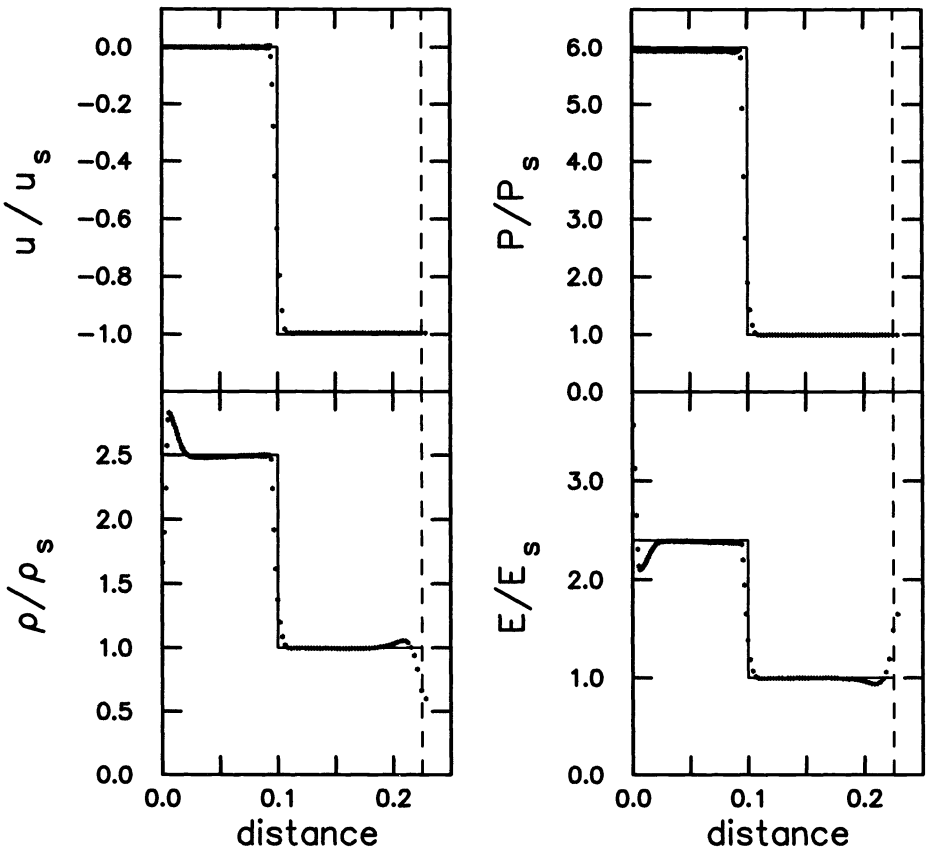


FIG. 1. *Numerical profiles for planar Noh problem: Piston driving strong left facing shock that reflects from a rigid wall at* $x = 0$. *Numerical profiles are at time when reflected shock is at* $x = 0.1$. *The fluid has an ideal gas* EOS *with* $\gamma = 5/3$. *The solid line is the hyperbolic solution and the dashed line is position of the piston.*

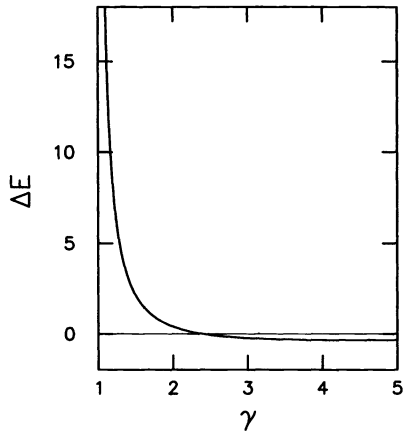


FIG. 2. *Difference of the scaled relative energy for strong shock reflection with Eulerian viscous length scale;* $E = \mathcal{E}^T/\epsilon$.

(ii) $\Delta\mathcal{E}^T/\epsilon = 0$ at $\gamma \approx 2.4$.
(iii) The minimum value of $\Delta\mathcal{E}^T \approx -0.34$ occurs at $\gamma \approx 4.65$.
(iv) $\Delta\mathcal{E}^T \to 0$ as $\gamma \to \infty$.
We note that in general $\Delta\mathcal{E}^T$ is not zero.

The analytic formulae for $\delta\mathcal{E}_s^T$ and $\delta\mathcal{E}_{rs}^T$ describe the continuum steady viscous shock profiles of the partial differential equations (PDEs). They would approximate the values for a numerical calculation with a uniform Eulerian grid to the extent that the artificial viscous shock profile is resolved. In a Lagrangian calculation the length scale for the artificial viscosity is typically chosen to be the grid spacing $\ell = \Delta x$. Consequently, the viscous length scale changes with the flow. In one space dimension, for a real viscosity $Q = \nu\partial_x u$, the usual choice for an Eulerian grid corresponds to a constant kinematic viscosity ν and for a Lagrangian grid to a constant dynamic viscosity $\eta = \rho\nu$.

For the shock reflection problem, the reflected shock length scale is decreased by the compression ratio of the incident shock. Neglecting the change in length scale within the shock profile, for a Lagrangian calculation the difference in the relative energies would be

$$(\Delta\mathcal{E}^T)_L \approx \delta E_s - \frac{\gamma - 1}{\gamma + 1}\delta E_{rs}. \tag{4.2}$$

The relative energy of the reflected shock compared to the incident shock is increased by its higher pressure and density but decreased by its lower Mach number. The decreased viscous length scale in a Lagrangian calculation cancels the effect of the increased density on the relative energy of the reflected shock compared to the incident shock. The difference in the Lagrangian relative energies for the shock reflection is shown in Fig. 3. We note that the singularity at $\gamma = 1$ in $\Delta\mathcal{E}^T/\epsilon$ has been removed but a singularity remains in the energy scale ϵ.

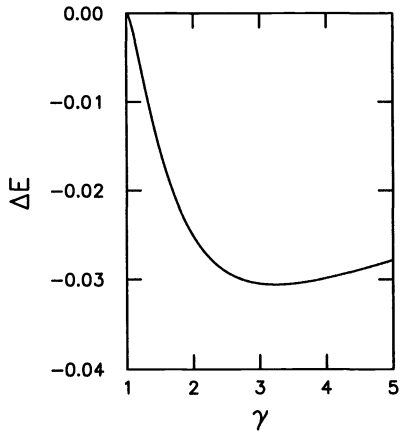


FIG. 3. *Difference of the scaled relative energy for strong shock reflection with Lagrangian viscous length scale;* $E = \mathcal{E}^T/\epsilon$.

For a typical case, $\gamma = 5/3$, the approximate Lagrangian relative energy difference equation (4.2) is a factor of 100 smaller than the Eulerian difference equation (4.1) and has the opposite sign. This suggests the effect of the change in shock profile from the incoming to outgoing wave would be smaller for a Lagrangian algorithm than for an Eulerian algorithm. However, the large change in magnitude may be fortuitous and due to the approximation which neglects the change in length scale within the shock profile.

4.1. Consequence of shift in relative shock energy. The numerical error shown in Fig. 1 can be understood as follows. The constant flux ahead of the outgoing wave can be accounted for by comparing the position of the shock in the viscous solution to that of the hyperbolic solution. The shift in the energy of the viscous shock profiles implies that a uniform state behind a steady state outgoing wave can not simultaneously satisfy the flux relations for mass, momentum, and energy. Instead, the shock interaction must result in a transient. The transient occurs on both a fast and slow time scale and results in an entropy error when comparing the viscous solution to the hyperbolic solution. The two time scales can be seen in the time sequence of numerical profiles shown in Fig. 4. The numerics are discussed later in more detail.

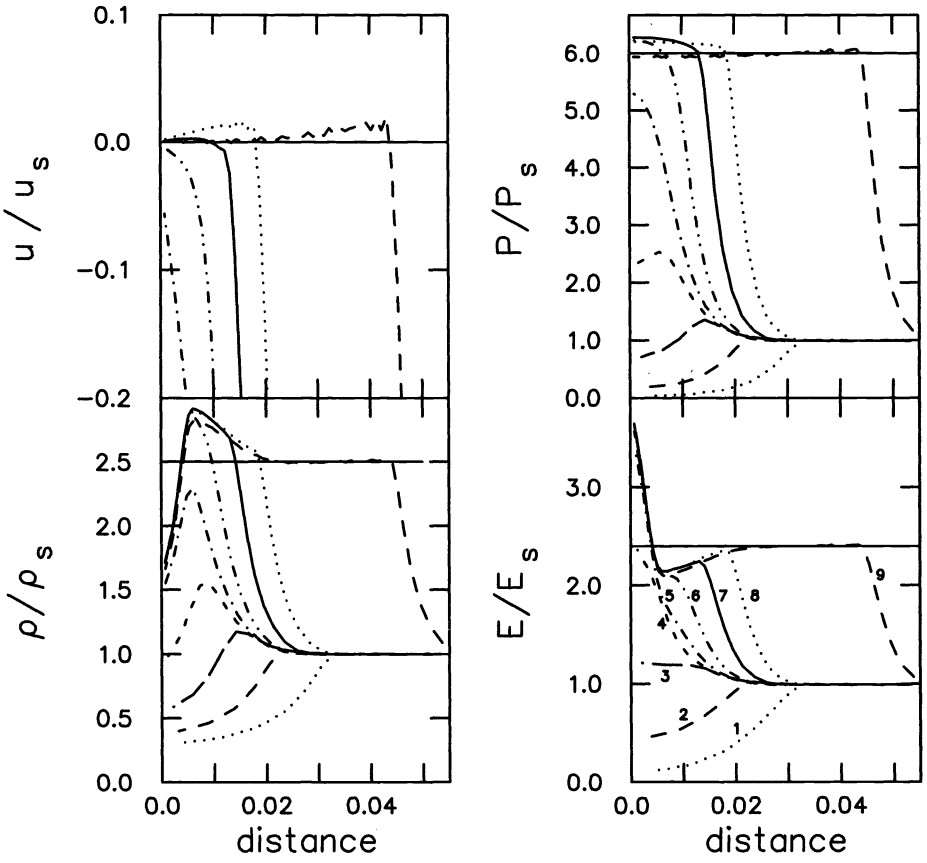


FIG. 4. *Time sequence of numerical profiles for planar Noh problem. The range of the sequence covers the transient in which the shock interaction takes place. Curves* 1 *and* 2 *are the profiles of the incident left-facing shock. In curves* 3 *and* 4 *the profile is in transition from the incoming to outgoing shock. Curves* 5–9 *are the profiles for the right-facing reflected shock. Curve* 9 *is essentially the asymptotic wave profile. The horizontal solid line corresponds to the value behind the reflected shock.*

Over the fast time scale, (shock width)/(shock velocity), the viscous pressure smooths out any discontinuity in the nondegenerate or acoustic modes. This is important when the positions of the incoming and outgoing shock waves are within a few shock widths of the wall. The pressure and particle velocity rapidly equilibrate to their respective values behind the reflected shock as the incoming shock profile changes to the outgoing profile. On the slow

time scale, the wave profile of the viscous solution approaches the steady state profile and the wave structure of the viscous solution asymptotically approaches the outgoing waves in the solution to the Riemann problem.

On the slow time scale, the shift in energy is small compared to the total energy behind the shock. The energy mismatch in the shock profiles can be distributed over the region between the wall and the shock front by acoustic waves. The entropy error at the shock front is small and further decreases rapidly for large t. This is a consequence of the fact that the Hugoniot jump conditions give the correct entropy jump across a steady state shock profile independent of the form of dissipation.

On the fast time scale, the energy shift is significant compared to the total energy in the shock profile. This results in a significant entropy error in the interaction region during the transient in which the shock profiles change. After the pressure and particle velocity have equilibrated, the viscous pressure in the interaction region approaches zero and the subsequent change in entropy is negligible. Without heat conduction, which would give rise to diffusion of entropy, the entropy error is frozen into the particle trajectories. Thus, the bulk of the entropy error from the interaction is confined to within a few shock widths of the wall.

Let us consider in more detail the interaction region for the case when $\Delta\mathcal{E}^T > 0$. Near the wall the outgoing viscous wave must have a deficit in energy equal to $\Delta\mathcal{E}^T$ to compensate for the energy difference in the shock profiles. Because the wall causes the particle velocity to approach zero, the energy density reduces to $\mathcal{E} = \rho E = P/(\gamma - 1)$ and is proportional to P. When the reflected wave has propagated a couple of shock widths, the pressure has approximately equilibrated to the value behind the outgoing hyperbolic shock. To conserve total energy, the viscous shock front must be slightly behind the hyperbolic shock front. Then to conserve mass, on average ρ must be above the value for the hyperbolic shock. Since P is approximately constant, a high value for ρ implies on average the entropy $S \propto \log(P/\rho^\gamma)$ is low.

At the wall, the pressure rise is more characteristic of a single strong shock than a double shock. Since the entropy is greater for a single strong shock then for two sequential shocks to the same final pressure, right at the wall we expect the entropy to be high and the density to be low. This implies there is an oscillation in the density and entropy in the vicinity of the wall. The pressure and density determine the specific energy through the equation of state. At the wall, a low value of ρ results in a high value of E. This agrees with the results of numerical calculations and is what Noh [6] called "excess wall heating," even though there is a damped oscillation in the energy about the value behind the hyperbolic shock.

Finally, to conserve total momentum the velocity profile overshoots and becomes slightly negative immediately behind the viscous shock front. We note at the wall that the mass and energy flux are identically zero. However, the momentum flux depends on the wall pressure and varies with the shock profile during the transient interaction. This affects the rate at which the asymptotic shock position based on mass and momentum equilibrate. Since the transient wall pressure depends on the form of the dissipation, the entropy oscillation at the wall is expected to vary with the numerical algorithm.

As the wave moves further away from the wall, the viscous profile more closely approaches that of a steady state shock wave. Consequently, the entropy jump across the viscous wave rapidly approaches the value for the hyperbolic shock. As time progresses, further errors in entropy outside the interaction region are negligible.

To understand the small distance it takes for the shock to form and the pressure and velocity to equilibrate we estimate the magnitude of $\delta\mathcal{E}_s^T$ relative to the energy in the shock profile. For illustrative purposes we assume $\gamma = 5/3$. From (3.3) the shock width is $\Delta x = 2.72\ell$. The compression ratio of a strong shock is $\eta = (\gamma + 1)/(\gamma - 1) = 4$. From (3.12), the energy

ratio is $\delta\mathcal{E}_s^T/(\Delta x\mathcal{E}_s) = 1/9$. Thus the energy in the shock profile will have a small effect on the shock interaction after the outgoing shock has propagated a couple of shock widths. This also suggests that an entropy oscillation, related to the transient wall pressure that is algorithm dependent, can greatly increase the L^1 norm of the entropy error.

We note that the initial data for Noh's test problem corresponds in effect to taking the relative energy of the incoming wave to be zero. Furthermore, the outgoing wave is a strong shock. In this case, the energy difference for the interaction is $\Delta\mathcal{E}^T = \delta\mathcal{E}_s^T$. Again, in general $\Delta\mathcal{E}^T$ is not zero and an entropy error occurs from the transient interaction that forms the outgoing shock. For a typical numerical calculation, Lagrangian shock-capturing algorithm, and $\gamma = 5/3$, the difference in the relative energy is larger for an initial velocity with a discontinuity than for an initial velocity with a smooth profile. Consequently, the entropy error in Noh's test problem on shock formation is larger than the error for the shock interaction problem we are considering. The Noh problem is a very good test problem for numerical algorithms because it exaggerates a generic type of error that occurs for shock interactions.

4.2. Results of numerical computation. It is unlikely that the details of the transient interaction region can be calculated analytically. Here we present the results of a numerical calculation that illustrates the general features of the solution discussed in the preceding section.

The calculation was performed with the HYDROX code [9]. It uses an old Lagrangian algorithm based on control volumes that is conservative and formally second-order accurate. It was chosen merely to demonstrate the effect with an "off the shelf" code. We expect other codes using newer algorithms that can resolve a shock with fewer cells to have a quantitatively smaller but qualitatively similar entropy error.

For the numerical example, a piston boundary condition is used to generate a strong left facing shock wave in an ideal gas with $\gamma = 5/3$. To avoid numerical problems from a zero sound speed in the ambient state, a Mach number of $\sigma/c_0 = 100$ instead of an infinite strength shock was used. The calculation used 150 cells to ensure that the incoming shock had a well-developed profile before reflecting from a rigid wall boundary. A purely quadratic artificial viscosity with a relatively large coefficient of $C_v = 10$ and correspondingly small time step, Courant–Friedrichs-Lewy (CFL) factor of 0.1, was used to minimize post shock oscillations so that the dominant error is from the shock interaction. The large numerical viscosity results in a partially resolved shock profile over several cells. This enhances the radial extent of the entropy error at the wall. Though computationally inefficient, it is convenient for illustrative purposes to exaggerate the effect.

The profiles after the transient has decayed away are shown in Fig. 1. Each dot corresponds to a grid point. The rigid wall is at the left and the piston is at the right. We note that the pressure and particle velocity are in good agreement with the hyperbolic solution. There are errors in density and specific energy in localized regions about both boundaries. The error at the piston boundary is identical to Noh's original problem. Because of Galilean invariance, an impulsively driven piston is equivalent to an initially constant velocity directed towards a stationary wall.

A time sequence of profiles of the shock interaction at the rigid wall is shown in Fig. 4. We note that the incoming shock width is about $\Delta x \approx 0.02$. Hence the plots have zoomed in on a region within a couple of shock widths of the wall. We see that the pressure and particle velocity equilibrate almost immediately to their respective values behind the reflected shock. The shock profiles have stabilized and the density and specific energy have almost equilibrated to their steady state values after the reflected shock has propagated only two widths of the incoming shock. For an ideal gas the entropy is proportional to $\log(P/\rho^\gamma)$.

Since $P/\rho^\gamma \propto E^\gamma/P^{\gamma-1}$ and P is constant behind the reflected shock, the entropy profile has the same qualitative shape as the specific energy profile. Consequently, we see that before the entropy equilibrates to its value behind the reflected shock there is an excess followed by a deficit in entropy within one to two shock widths of the wall.

Finally, the results of another calculation with an algorithm that resolves a shock with fewer cells is shown in Fig. 5. The numerical algorithm uses a linear plus quadratic artificial viscosity, $Q = C_{v1}\rho c J[u] + C_{v2}\rho J[u]^2$, where $J[u]$ is a monotonicity limited velocity jump within a cell. The parameters used were $C_{v1} = 0.5$, $C_{v2} = 2/3$ and CFL factor of 0.25. The figures shows that within two cells the shock profile varies from 10%–90% of the asymptotic values ahead of and behind the shock. Qualitatively, the results are the same as in Fig. 1. As expected, the error at the wall involves fewer cells. We also note that the first cell next to the wall boundary has a deficit rather than an excess of energy. This illustrates that the transient wall pressure is algorithm dependent and quantitatively effects the entropy error.

Both these calculations show a qualitatively similar effect; nonzero entropy error localized to the interaction region. Quantitatively, the entropy error varies in magnitude, spatial extent, and even in sign. We believe the entropy error is due to the change in the relative energy of the shock profiles and conjecture that any shock capturing algorithm without significant heat conduction will have a qualitatively similar error.

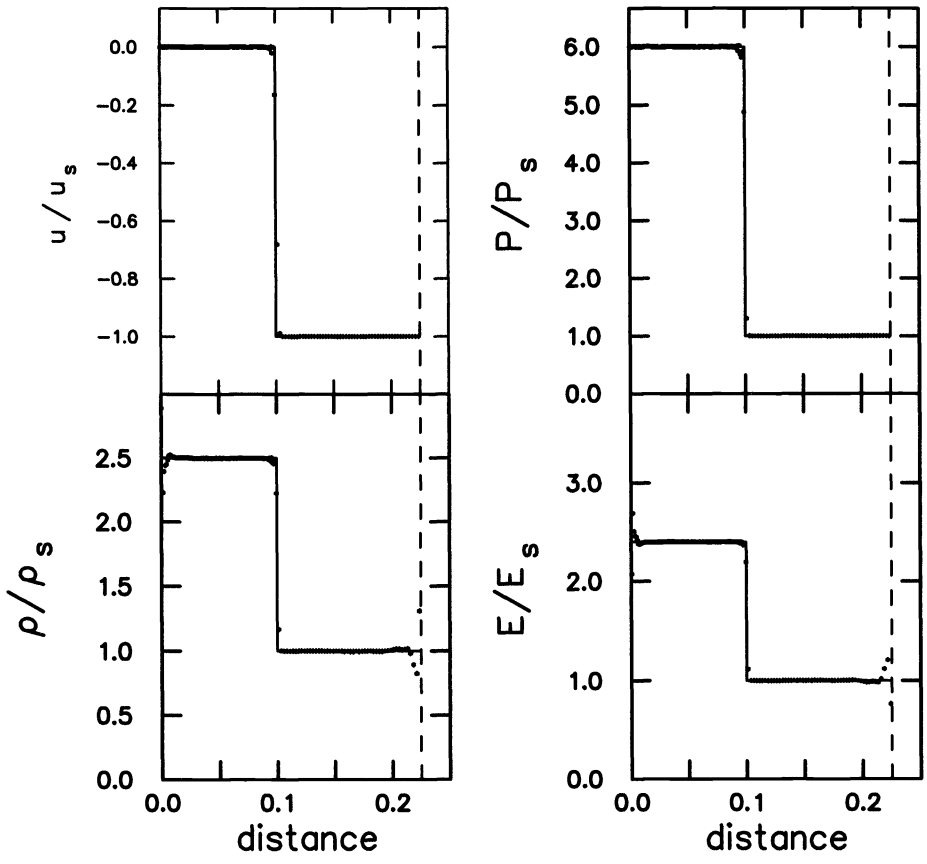


FIG. 5. *Numerical profiles for planar Noh problem. Same as Fig.* 1 *but with numerical algorithm that resolves shock with fewer cells.*

5. Nonuniform convergence of inviscid limit. For a shock reflecting from a rigid wall, a scaling argument due to Noh [6] shows that under mesh refinement the entropy error decreases in spatial extent but the peak pointwise error is constant in magnitude. Since the coefficient of viscosity decreases with the mesh size, the scaling implies that the convergence of the inviscid limit to the hyperbolic solution is nonuniform. Here, we extend this result on nonuniform convergence to a general shock interaction.

The inviscid fluid equations are scale invariant. Scaling space and time amounts to a choice of units. Viscosity introduces a length scale that breaks the invariance. However, under scaling the viscous pressure is multiplied by a constant. Therefore, by scaling the viscosity coefficient along with the length and time scales, the equations are again invariant. This scaling invariance applies to both the PDEs and the finite difference equations in either Eulerian or Lagrangian coordinates provided that the artificial viscous length scale is proportional to the grid spacing.

A solution to the fluid equations with the von Neumann–Richtmyer viscosity is invariant under the transformation $x' = \alpha x$, $t' = \alpha t$ and $\ell' = \alpha \ell$; i.e., for fixed ℓ if $U_{C_v}(x, t)$ is a solution to the equations with viscosity coefficient C_v then $\widetilde{U}_{\alpha^2 C_v}(x, t) = U_{C_v}(x/\alpha, t/\alpha)$ is also a solution. Furthermore, this transformation preserves velocity and hence the initial value data. Therefore, the inviscid limit corresponds to $\alpha \to 0$. In this limit, the entropy error at the wall decreases in spatial extent but is constant in magnitude. Hence the inviscid limit for the shock reflection problem converges in L^1 or L^2 but not in L^∞.

A shock reflecting from a rigid wall is equivalent to the symmetric collision of two shocks, i.e., equal strength shocks of the opposite family. The argument that the cause of the error is due to the asymptotic shift in the relative energy between the incoming waves and the outgoing waves implies that the equal strength of the incoming waves is not important. Consequently, shock interactions in general will result in nonuniform convergence of the inviscid limit.

Numerical shock-capturing algorithms differ from each other and the continuum PDEs with regard to three properties that are important for the shock interaction problems. The mesh spacing introduces two dimensionless ratios. The first ratio is the CFL number $c\Delta t/\Delta x$. The second ratio is the shock width divided by the mesh spacing, or the number of cells over which the shock is spread out. In addition, numerical algorithms can use different forms of dissipation. This affects both the stability and minimum number of cells required for a shock. Moreover, the form of the dissipation determines the shock profile and hence the relative energy $\delta\mathcal{E}^T$.

Once the CFL number is small enough for stability, it is not expected to affect the behavior of the solution. The argument based on the change in relative energy of the shock profiles suggests that all shock-capturing algorithms without significant heat conduction will have the same qualitative entropy error when shock waves interact. Since the relative energy difference scales with the shock width, we expect the entropy error from a shock interaction to be smaller when the shock width is narrower.

The entropy error from a shock interaction can be affected by an adaptive mesh. Coarsening a mesh is a smoothing operation. Behind the shock front it has an effect similar to heat conduction. Smoothing out the energy oscillation in the interaction region can greatly reduce both the spatial extent and the magnitude of the entropy error.

6. Effect of source terms. Noh [6] also has a version in cylindrical and spherical geometry of his test problem on the formation of a shock from a velocity discontinuity. This introduces an additional effect on the outgoing shock wave due to the geometrical source term. Depending on the form of the artificial viscosity, some numerical algorithms have much larger errors for the test problem in cylindrical and spherical geometry than in the planar case.

The geometrical source term is singular at the origin. Consequently, when a shock approaches the origin, within the shock profile the geometric source term becomes comparable

in magnitude to that of the viscous term. When this occurs, the conservation form of the equations no longer imply the usual Hugoniot jump relations across a shock; i.e., the nonzero shock width becomes important.

A real effect in which the Hugoniot jump relations are modified occurs for detonation waves [2]. In this case, the competition between a chemical reaction and a geometrical source term gives rise to the curvature effect in which the detonation velocity depends on the curvature of the detonation front [10], [4]. Quantitatively, the magnitude of the curvature effect depends on the details of the reaction rate. To first order, the Hugoniot jump conditions are modified by the addition of terms proportional to the ratio of the reaction zone width to the front curvature. An artificially large numerical shock width and a geometric source term can have a similar effect near the origin. Furthermore, the magnitude of the effect depends on the dissipation rate within the shock profile.

From the Gurdeley similarity solution, an ideal converging shock is singular at the origin. The shock width provides a length scale that regularizes the singularity when the shock reflects from the origin. After reflection there are large gradients behind the shock front. The shock has to propagate a sufficient distance from the origin for the gradients behind the shock to be small compared to those within the shock profile. This is a necessary condition for the usual Hugoniot jump relations to apply across the shock independent of the form of dissipation.

Thus, when source terms or gradients behind the shock front are large compared to the dissipation within the shock profile, the viscous solution can differ significantly from the hyperbolic solution. In the absence of a shock wave, the geometric source term does not result in any dissipation. Consequently, an entropy error due to the interaction of a shock wave and a geometric source term is convected and remains localized. Since the error is expected to be proportional to the shock width over the radius, algorithms with the smallest shock width should have the smallest error.

For the 1-D fluid equations in cylindrical and spherical geometry, there are two natural possibilities for the artificial viscous dissipation. The first form results from projecting the inviscid three-dimensional (3-D) fluid equations onto the radial coordinate and then adding a scalar artificial viscous pressure. The second form adds an artificial tensor viscosity to the 3-D fluid equations [8], [11] and then projects onto the radial coordinate. The resulting form of the dissipation in the two cases is different. A traceless stress tensor corresponds to a shear viscosity. In contrast to a scalar viscosity, there is no artificial dissipation for a uniform compression with a shear viscosity. Numerical experiments have shown that the traceless tensor form of artificial viscosity gives a smaller entropy error for the Noh problem [1].

7. Conclusion. We have analyzed the problem of a strong viscous shock reflecting from a rigid wall. For the von Neumann–Richtmyer artificial viscosity, we have shown that the same type of error occurs as for the Noh problem [6]. The phenomena is much more general than a boundary effect, "excess wall heating." It is an entropy error that occurs either when a shock is formed or during the transient when shock waves interact. The error is due to the difference in energy relative to the hyperbolic solution of the viscous profiles for the incoming and outgoing shock waves. After a shock has passed entropy is convected. Consequently, the entropy error from a shock interaction remains localized and does not dissipate. Noh's scaling argument has been extended to show for shock interactions that under mesh refinement the entropy error decreases in spatial extent but the peak pointwise error is constant in magnitude.

From the energy argument based on the shock profiles, we conjecture that the same behavior occurs for an arbitrary shock interaction with any dissipative mechanism that results in a nonzero shock width, provided there is no heat conduction to diffuse entropy. The dissipation may correspond to a term added to the hyperbolic PDEs, e.g., an artificial viscosity, or can be numerical in nature, e.g., resulting from truncation errors in the differencing scheme,

or a Riemann solver used in the Godunov method. The fact that hyperbolic finite difference schemes deliberately underresolve the shock profile is not critical as long as the algorithm is stable. The truncation errors merely introduce an oscillation in the shock profile as the position of the shock front propagates between grid points.

The entropy error when shock profiles interact implies a nonuniform convergence of the inviscid limit to the hyperbolic solution. Nonuniform convergence can be expected at the shock front. An additional nonuniformity can occur in a region in which the solution is smooth resulting from a shock interaction that occurred in the region's past history.

A more severe form of this entropy error occurs when a shock wave is incident on a material interface or contact. For materials with different equations of state or when the contact is a discontinuous change in zoning, there can be a large transient resulting from the change in profiles for the incident, transmitted, and reflected shock waves. In Lagrangian algorithms the effect is partially ameliorated by choosing the grid such that the wave speed in units of zones per time step is the same for the outgoing shocks on each side of the interface. However, the minimal error is similar to that which occurs for the shock interaction discussed here.

In more complicated 1-D fluid flows, additional errors can result from the inhomogeneities caused by the entropy error from shock interactions. For example, subsequent shock waves will scatter off the inhomogeneities and spread the spatial extent of the error. This effect is partially ameliorated by the fact that shock heating raises the sound speed. Hence, subsequent reflected shocks have a lower Mach number and the additional entropy errors they cause decrease as the shocks weaken.

In two dimensions, the interaction of two incoming shocks gives rise to wave patterns [3]. A steady interaction region in two space dimensions is qualitatively similar to the time dependent interaction region in one dimension. Hence, we expect an entropy error for two-dimensional (2-D) wave patterns in the region in which the shock profiles overlap. Errors from numerical shock profiles in two dimensions may have a more dramatic effect on the subsequent flow than in one dimension. For example, as the strength or angle between the incoming shock waves evolves in time, the wave patterns can bifurcate. One well-known bifurcation is the transition from regular to Mach reflection. Because the transition point of a bifurcation is a threshold phenomena, it can be very sensitive to parameters such as the viscous shock profile. Another example occurs in an unstable 2-D flow. The inhomogeneities from entropy errors caused by shock interactions can be the seed of a perturbation which subsequently leads to or enhances the growth of an instability. This raises the important question as to whether in two dimensions the inviscid limit always converges to the hyperbolic solution.

For some applications, the nonuniform convergence is important. One example is when comparing the calculated temperature at a wall to experimental data. The numerical entropy error from a reflected shock results in a high wall temperature that does not dissipate in time. Moreover, the calculated wall temperature does not improve under mesh refinement. Having understood the cause, we can compensate for this error, e.g., with sufficient resolution by averaging over a small region in the vicinity of the wall. Another example is when the material is chemically reactive. In particular, for an explosive a numerical hot spot caused by a shock interaction can initiate a detonation and greatly affect the fluid flow.

The spatial extent of the entropy error when shocks interact is proportional to the shock width. Thus, this error is smallest for those numerical schemes that minimize the artificial shock width. In particular, this type of error can be eliminated by using a front tracking algorithm.

Appendix: Evaluation of integral. The needed integrals can be evaluated by contour integration as follows. Suppose $a > b > 0$ and n is a nonnegative integer. Let $z = e^{ix}$. Then the basic integral of interest can be expressed as

$$\int_0^\pi dx \frac{\cos(nx)}{a + b\cos(x)} = \mathrm{Re} \int_{C_0} \frac{-idz}{z} \cdot \frac{z^n}{a + \frac{1}{2}b(z + 1/z)}$$
$$= \mathrm{Im} \int_{C_0} dz \frac{z^n}{\frac{1}{2}bz^2 + az + \frac{1}{2}b},$$

where C_0 is the arc of a unit circle in the upper half of the complex plane.

The denominator of the integrand on the right-hand side has two zeros located at

$$z_\pm = \left[-a \pm (a^2 - b^2)^{1/2}\right]/b.$$

These lie along the real axis with $z_- < -1$ and $-1 < z_+ < 0$. Let C be the path formed by closing the path C_0 along the x axis but going around the pole at z_+ in the upper half plane. By applying Cauchy's residue formulae we obtain

$$\int_0^\pi dx \frac{\cos(nx)}{a + b\cos(x)} = \mathrm{Im}\left(\int_C dz \frac{z^n}{\frac{1}{2}bz^2 + az + \frac{1}{2}b} - \mathrm{PV}\int_{-1}^1 dx \frac{x^n}{\frac{1}{2}bx^2 + ax + \frac{1}{2}b} + i\pi\ \mathrm{Residue}(z_+)\right)$$
$$= \pi\ \mathrm{Residue}(z_+)$$
$$= \frac{\pi}{(a^2 - b^2)^{1/2}} \left(\frac{(a^2 - b^2)^{1/2} - a}{b}\right)^n.$$

Using the symmetry of the sin and cos functions over a half cycle we note two special cases of the above formula

$$\int_{-\pi/2}^{\pi/2} dx \frac{1}{a + b\sin(x)} = \int_0^\pi dx \frac{1}{a + b\cos(x)}$$
$$= \frac{\pi}{(a^2 - b^2)^{1/2}}$$

$$\int_{-\pi/2}^{\pi/2} dx \frac{\cos^2(x)}{a + b\sin(x)} = \int_0^\pi dx \frac{\sin^2(x)}{a + b\cos(x)}$$
$$= \frac{1}{2}\int_0^\pi \frac{1 - \cos(2x)}{a + b\cos(x)}$$
$$= \frac{\pi}{b^2}\left[a - (a^2 - b^2)^{1/2}\right].$$

Acknowledgments. I thank my colleague Klaus Lackner for many stimulating and enlightening discussions on this subject. In addition, I thank Paul Whalen for sharing his insight on the Noh problem from the many calculations he has performed using a wide variety of different numerical algorithms.

REFERENCES

[1] J. R. BUCHLER AND P. WHALEN, *Experiments with artificial viscosity*, in The Numerical Modeling of Nonlinear Stellar Pulsations, J. R. Buchler, ed., NATO ASI Series C 302, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1990, pp. 315–322.

[2] B. BUKIET AND R. MENIKOFF, *Detonation waves and the front tracking method*, Phys. Fluids A, 4 (1992), pp. 2070–2081.

[3] L. F. HENDERSON, *On the Refraction of Longitudinal Waves in Compressible Media*, Report UCRL-53853, Lawrence Livermore National Laboratory, Livermore, CA, 1988.

[4] J. JONES, *The spherical detonation*, Adv. Appl. Math., 12 (1991), pp. 147–186.

[5] J. VON NEUMANN AND R. D. RICHTMYER, *A method for the numerical calculation of hydrodynamic shocks*, J. Appl. Phys., 21 (1950), pp. 232–237.

[6] W. F. NOH, *Errors for calculations of strong shocks using artificial viscosity and an artificial heat flux*, J. Comput. Phys., 72 (1987), pp. 78–120.

[7] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, Wiley Interscience, New York, 1967.

[8] W. D. SCHULTZ, *Tensor artificial viscosity for numerical hydrodynamics*, J. Math. Phys., 5 (1964), pp. 133–138.

[9] M. S. SHAW AND G. K. STRAUB, *HYDROX: A One-Dimensional Lagrangian Hydrodynamics Code*, Report LA-8642-M, Los Alamos National Laboratory, Los Alamos, NM, 1981.

[10] S. STEWART AND J. B. BDZIL, *The shock dynamics of stable multidimensional detonation*, Combust. Flame, 72 (1988), pp. 311–323.

[11] W. M. TSCHARNUTER AND K.–H. WINKLER, *A method for computing self-gravitating gas flows with radiation*, Comput. Phys. Comm., 18 (1979), pp. 171–199.

SIAM J. SCI. COMPUT.
Vol. 15, No. 5, pp. 1243–1250, September 1994

016

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.

GAUSS–SEIDEL ITERATION FOR STIFF ODES FROM CHEMICAL KINETICS*

J. G. VERWER†

Abstract. A simple Gauss–Seidel technique is proposed that exploits the special form of the chemical kinetics equations. Classical Aitken extrapolation is applied to accelerate convergence. The technique is meant for implementation in stiff solvers that are used in long range transport air pollution codes using operator splitting. Splitting necessarily gives rise to a great deal of integration restarts. Because the Gauss–Seidel iteration works matrix free, it has much less overhead than the modified Newton method. Start-up costs therefore can be kept low with this technique. Preliminary promising numerical results are presented for a prototype of a second order backward differentiation formula (BDF) solver applied to a stiff ordinary differential equation (ODE) from atmospheric chemistry. A favourable comparison with the general purpose BDF code DASSL is included. The matrix free technique may also be of interest for other chemically reacting fluid flow problems.

Key words. numerical stiff ODEs, chemical kinetics, air pollution modelling

AMS subject classifications. Primary: 65L05; Secondary: 80A30, 80A32

1. Introduction. Large scale, long range atmospheric air pollution models are computationally very expensive [10]. Usually the computational work is heavily dominated by the numerical treatment of the stiff ODE systems describing the chemical kinetics model in use. These ODE systems are of the nonlinear form

$$\frac{d}{dt}y = f(t, y) := P(t, y) - L(t, y)y, \qquad y(t) = (y_1(t), \ldots, y_m(t))^T, \tag{1.1}$$

where $P(t, y)$ is a vector and $L(t, y)$ is a diagonal matrix. The components $P_k(t, y)$ and $L_k(t, y)y_k$ are nonnegative and represent, respectively, production and loss terms. The reciprocal of L_k is the physical time constant or characteristic reaction time for compound y_k. Generally the range of time constants is large, which implies that in most applications the ODE system is stiff. For example, if we assume that the popular operator splitting approach is used, then a common situation is that (1.1) must be solved repeatedly, over several hundreds of split-step time intervals, at any of thousands of grid points. As a rule the length of this time interval is the stepsize used in the advection step, which for part of the chemical species occurring in atmospheric applications is much larger than the time constant. This introduces the stiffness. When the chemistry is nonlinear and many species are involved, say, 20 to 50, it is clear that a highly efficient stiff solver, tailored to the application under consideration, is indispensable. Due to the great number of restarts, one for each split-step time interval, it is

* Received by the editors October 20, 1993; accepted for publication (in revised form) March 4, 1994. This paper is one of a series on the development of algorithms for long range transport air pollution models (projects EUSMOG and CIRK). Financial support was provided by The Dutch National Institute of Public Health and Environmental Protection (RIVM).

† CWI, Research Institute of the Stichting Mathematisch Centrum, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (janv@cwi.nl).

particularly important to use integrators that keep the inevitable start-up costs low, in comparison with a common stiff ODE integration. In addition, the integrator must be able to change stepsize rapidly with little costs because in practice rate coefficients in atmospheric chemistry models are temperature or time dependent. This dependency can cause sudden changes in the concentrations, which can be followed accurately and efficiently only if stepsizes can be adjusted efficiently.

2. The Gauss–Seidel iteration. The purpose of this note is to present some preliminary, but promising, results of a simple Gauss–Seidel iterative technique for solving implicit relations. Because this technique works matrix free, a change of stepsize involves no numerical algebra overhead at all. This results in low start-up cost compared to Newton-type iteration. The Gauss–Seidel technique is very cheap in the start-up phase where usually a few iterations are sufficient. When the stepsize increases, the convergence will normally slow down. In the experiments reported in this note we have successfully applied the classical Aitken extrapolation to accelerate convergence over a wide range of stepsizes. So far only a single extrapolation step has been considered. In the near future more sophisticated acceleration techniques will be the subject of further investigation. Moreover, switching between Gauss–Seidel and the modified Newton iteration will be examined. We also experimented with the Jacobi iteration, but less successfully. In the experiments reported here, the Gauss–Seidel technique significantly improved convergence, especially for larger stepsize values. Yet the Jacobi iteration may be of interest in the initial transient phase, because one Jacobi iteration is always cheaper than one Gauss–Seidel iteration. For simplicity of presentation, in this note we focus on the Gauss–Seidel iteration.

We have implemented the Gauss–Seidel iteration process in a prototype of a new solver. This prototype uses as the main integrator the variable step, second order BDF formula

$$y^{n+1} = Y^n + \gamma\tau f(t_{n+1}, y^{n+1}), \qquad \tau = t_{n+1} - t_n, \tag{2.1}$$

where $\gamma = (c+1)/(c+2)$, $c = (t_n - t_{n-1})/(t_{n+1} - t_n)$ and

$$Y^n = \left((c+1)^2 y^n - y^{n-1}\right)/(c^2 + 2c). \tag{2.2}$$

We emphasize that second order is sufficient because for reactive flow problems a low level of accuracy of, say, 1% is good enough. A higher level is thought to be superfluous, due to errors made in other (operator splitting) processes and uncertainties in the reaction constants of the chemistry models.

The Gauss–Seidel technique exploits the chemical kinetics form (1.1), by which (2.1) can be written as

$$y^{n+1} = F(y^{n+1}) := \left(I + \gamma\tau L(t_{n+1}, y^{n+1})\right)^{-1} \left(Y^n + \gamma\tau P(t_{n+1}, y^{n+1})\right). \tag{2.3}$$

The Gauss–Seidel iteration is now straightforwardly applied to the nonlinear system of equations $y = F(y)$ in the standard way. The diagonal form of L makes this process essentially an explicit one. Note that the technique can be implemented in a completely similar way into a diagonally implicit Runge–Kutta code, say, into one of order two with two stages. Due to the one-step nature, much larger differences between successive stepsizes can then be realized. A disadvantage is that more iterations are required per integration step. This is attributable to the computation of the second stage. In our future work, we will explore the efficiency of a two-stage Runge–Kutta method versus that of our BDF method.

Note that for components for which both P_k and L_k are constant in y, the solution of (2.3) is obtained in one iteration. This means that when individual components rapidly approach their steady state values P_k/L_k, they are handled very efficiently. In this connection the current

iterative approach bears a resemblance to the explicit pseudo-steady approximation approach evaluated in [9]. The schemes evaluated in [9] show a very good stability behaviour, but proved to be so inaccurate that generally they cannot compete with state-of-the-art, general purpose codes like DASSL [1] and RADAU5 [6], even in the low accuracy range requested. A comparison with DASSL, presented in §4, will show that the iterative Gauss–Seidel technique offers better prospects for the development of fast, special purpose solvers for stiff ODEs in chemically reacting flows. Because the comparison is based on a single test problem, it is obvious that more experimental work is needed to justify completely the use of the Gauss–Seidel technique for chemical sub-models.

3. The prototype solver. We now present a brief outline of our prototype solver. This outline contains all the information needed to appreciate the experimental results presented in the next section. We begin with the variable stepsize strategy for the BDF method. Let E^{n+1} be a local error indicator and consider the weighted error norm

$$||E^{n+1}||_w = \max(|E_k^{n+1}|/W_k^n), \qquad W_k^n = ATOL + RTOL\,|y_k^n|, \tag{3.1}$$

where *ATOL* and *RTOL* are the absolute and relative error tolerance. If $||E^{n+1}||_w \leq 1.0$, then the integration step is accepted and otherwise rejected. The new stepsize τ_{new} is estimated by

$$\tau_{\text{new}} = \max(0.5, \min(2.0, 0.8/\sqrt{||E^{n+1}||_w}))\tau_{\text{old}}. \tag{3.2}$$

The square root appears here since our local error indicator is

$$E^{n+1} = \frac{2}{c+1}\left(cy^{n+1} - (1+c)y^n + y^{n-1}\right), \tag{3.3}$$

which yields $\tau^2 y''(t^n) + O(\tau^3)$ upon substitution of the exact solution. Hence we do not estimate the true local error of the BDF method, which is $O(\tau^3)$. The main reason is that we wish to avoid the explicit use of derivative values in E^{n+1}. As is well known, this would amplify small insignificant errors in the solution because of the stiffness, which hinders the prediction of the new stepsize. In codes using modified Newton iteration, this amplification is suppressed by an additional forward-backward substitution. Because we use Gauss–Seidel iteration, we cannot do this and therefore prefer to use the conservative estimate (3.3), which, in our experience, works well in combination with (3.2) and the iteration strategy described below. It is obvious, though, that more extensive tests are needed to justify (3.3) completely.

The missing starting value is computed with the implicit Euler method. To obtain a safe guess for the initial stepsize, we replace E^{n+1} in (3.1) by $\tau f(t_0, y^0)$ and define τ such that the weighted error norm is equal to one, i.e.,

$$\tau = \min(W_k^0/|f_k(t_0, y^0)|). \tag{3.4}$$

Hence the initial step is chosen so that the first Taylor series term $\tau f(t_0, y^0)$ satisfies the absolute/relative tolerance requirement. The two-step scheme is then applied with the same stepsize and after that the variable stepsize mechanism is activated. Normally, (3.4) will lead to a rather small initial guess, which will be accepted and subsequently rapidly increased according to (3.2). This is also the case in the experiments reported here.

Let $y^{(i)}$ denote the approximation to y^{n+1} after i iterations with the Gauss–Seidel method for (2.3) or its counterpart for the implicit Euler method. Let *ITOL* be a tolerance value. As an initial guess we use $y^{(0)} = y^n$. The first iterate $y^{(i)}$ satisfying

$$||y^{(i)} - y^{(i-1)}||_w \leq ITOL, \qquad i \geq 2, \tag{3.5}$$

is accepted as the new approximation y^{n+1}. Hence a minimum of two iterations is used at each timestep. The iteration is interrupted if $||y^{(i)} - y^{(i-1)}||_w > ||y^{(i-1)} - y^{(i-2)}||_w$. In the experiments reported here, such failure has not occurred.

Aitken extrapolation is applied for $i \geq 3$. Let $z^{(i)}$ be the Aitken extrapolated value at the ith iteration. Then, if (3.5) does not hold, $z^{(i)}$ is accepted as the new approximation y^{n+1} if

$$||z^{(i)} - z^{(i-1)}||_w \leq ITOL, \qquad i \geq 4. \tag{3.6}$$

Aitken extrapolation is applied to all components and consequently involves overhead, such as componentwise checks on zero division. For simple models the overhead may therefore annihilate the resulting speed in convergence. For the model example used below it has proven to be attractive, especially for larger stepsize values. For small stepsize values in a transient phase where only a few Gauss–Seidel iterations are used, the extrapolation has little effect, of course.

4. Numerical results. Following [8], [9], we present results for an air pollution model from atmospheric chemistry with 20 components and 25 reactions (see the Appendix). The initial data is such that an initial transient is present, while the Lipschitz constant is about $1.5\,10^7$. This was estimated in [9] by means of an explicit Runge–Kutta integration. Hence the ODE system is very stiff, provided the integration interval is sufficiently large, which is true here. We will give results for $t = 1$ minute, which is slightly after the initial transient, and $t = 60$ minutes, at which time the solution gets close to its steady state.

We also include a comparison with DASSL (see [1]; we have used the double precision version DDASSL available from Netlib [5]). This general purpose BDF solver has been applied as a black box using only default options, except that the initial stepsize was also determined by (3.4). Both DASSL and the new BDF solver can produce negative solution values. However, in the experiments reported here we have not noticed this. It should be noted that the reaction constants in the model example are constant, so that outside the initial transient no sudden large changes in the concentrations occur. This slightly favours DASSL in our comparison. It should also be noted that DASSL is a differential algebraic equation (DAE) solver and hence carries some overhead needed to solve nonlinearly implicit DAEs, even when it is applied to ODEs in normal form, such as (1.1). When this overhead becomes truly noticeable, then it is likely that BDF codes developed for ODEs, like VODE [2] or LSODE [7], will be faster than DASSL. The interested reader is referred to [6, §V.5], where several codes are briefly compared. More specific results obtained with VODE for chemical kinetics problems can be found in [3], [4].

For the prototype solver, Table 1 yields the following information at the specified time $t = T$. SD = the number of significant digits for the maximum relative error, defined by

$$SD = -\log_{10}\left(\max_k \frac{|y_k^n - y_k(T)|}{|y_k(T)|}\right), \tag{4.1}$$

$STEPS$ = the number of integration steps, $ITER$ = the total number of Gauss–Seidel iterations, and CPU = cpu time in seconds. Although CPU is an approximate value and implementation and machine dependent, the given times are indicative for comparison purposes (with an accuracy of at most 0.01 sec. on a Silicon Graphics Indigo workstation, using the Fortran77 Compiler Options -c -r8 -O). In Table 1 results are given for the following two coupled values for $ATOL$ and $RTOL$,

$$ATOL = 10^{-6}TOL, \qquad RTOL = TOL, \qquad TOL = 10^{-1}, 10^{-2}. \tag{4.2}$$

Recall that for reactive flow problems a low level of accuracy suffices, say, 1% ($SD = 2$). For these two values of TOL, the initial stepsize τ_1 determined by (3.4) is equal to $4.7\,10^{-7}$

and $4.7\,10^{-8}$, approximately. These small initial stepsizes reveal the initial transient and arise because we take *ATOL* rather small, which is desirable since a number of the concentrations are zero at the initial time and remain small for evolving time. To illustrate the convergence of the Gauss–Seidel iteration, two values for *ITOL* were used, viz., 10^{-2} and 10^{-3}. We emphasize that for the air pollution model considered here, $ITOL = 10^{-2}$ is sufficiently small to come close near the accuracy of the second order BDF formula. Note that inequalities (3.5) and (3.6) are based on the weighted error norm that contains *ATOL* and *RTOL*.

TABLE 1
Values (SD, CPU, STEPS, ITER) using Aitken extrapolation.

		$ITOL = 10^{-2}$	$ITOL = 10^{-3}$
$TOL = 10^{-1}$	$t = 1$	(1.87, 0.03, 42, 153)	(1.87, 0.03, 42, 183)
	$t = 60$	(2.11, 0.04, 56, 273)	(2.40, 0.05, 57, 351)
$TOL = 10^{-2}$	$t = 1$	(2.68, 0.06, 94, 369)	(2.68, 0.07, 94, 438)
	$t = 60$	(3.10, 0.09, 132, 663)	(3.08, 0.11, 132, 773)

We see from Table 1 that for the present example problem Gauss–Seidel iteration accelerated with Aitken extrapolation works very well. The rather high accuracy the prototype code yields for the tolerance values used is due to the conservative error indicator (3.3). Note that for $TOL = 10^{-1}$, $ITOL = 10^{-2}$, we are near the 1% error level. The average number of Gauss–Seidel iterations over the entire interval [0, 60] for this tolerance combination is approximately five. To illustrate the stepsize variation and convergence behaviour of the accelerated Gauss–Seidel iteration, Fig. 1 shows for this tolerance combination a plot of the stepsize sequence τ_n and of the associated number of iterations. We see that over a large range of stepsizes the number of Gauss–Seidel iterations remains limited. Only near the end of the interval, where we get close to the steady state and τ_n becomes quite large, the number of iterations starts to grow.

To show the effect of the Aitken extrapolation, we refer to Table 2, which gives the same information as Table 1, but without application of Aitken extrapolation. As to be expected, Aitken extrapolation becomes truly effective for the smaller tolerances $TOL = 10^{-2}$ and $ITOL = 10^{-3}$, while the convergence acceleration is largest for the larger stepsize values used when we approach steady state.

TABLE 2
Values (SD, CPU, STEPS, ITER) without using Aitken extrapolation.

		$ITOL = 10^{-2}$	$ITOL = 10^{-3}$
$TOL = 10^{-1}$	$t = 1$	(1.87, 0.03, 42, 171)	(1.87, 0.04, 42, 288)
	$t = 60$	(2.10, 0.05, 57, 450)	(2.39, 0.08, 57, 669)
$TOL = 10^{-2}$	$t = 1$	(2.68, 0.06, 94, 484)	(2.68, 0.09, 94, 754)
	$t = 60$	(3.07, 0.12, 132, 1016)	(3.08, 0.18, 132, 1537)

DASSL was also applied for the two coupled tolerances (4.2) and solved the problem without error test and convergence failures. The results of DASSL, in terms of *SD*, *CPU*, *STEPS*, *ITER*, and *JEVS*, are contained in Table 3. Now, *ITER* = the number of modified Newton iterations (or backsolves) and *JEVS* = the number of Jacobian evaluations (or LU decompositions). Note that DASSL computes the Jacobians by numerical differencing. DASSL yields results near the 1% error level for $TOL = 10^{-2}$. Comparing the results for $TOL = 10^{-1}$,

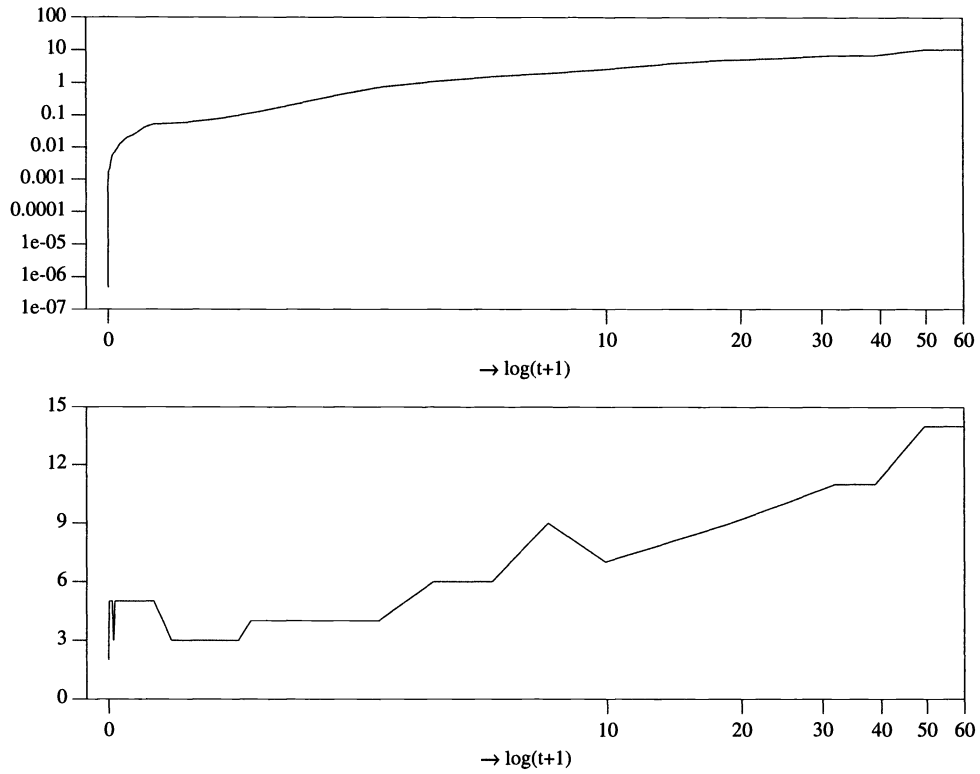


FIG. 1. *Stepsizes (upper plot) and number of Gauss–Seidel iterations (lower plot) for the tolerances TOL* $= 10^{-1}$, *ITOL* $= 10^{-2}$ *for* $0 \leq t \leq 60$. *Aitken extrapolation was applied.*

ITOL $= 10^{-2}$ from Table 1 with those in Table 3 for *TOL* $= 10^{-2}$, we see that near the 1% error level the prototype code runs approximately three times as fast as DASSL.

TABLE 3
Values (SD, CPU, STEPS, ITER, JEVS) for DASSL.

$TOL = 10^{-1}$	$t = 1$	(0.84, 0.07, 30, 42, 19)
	$t = 60$	(1.20, 0.09, 42, 60, 25)
$TOL = 10^{-2}$	$t = 1$	(2.17, 0.09, 49, 69, 19)
	$t = 60$	(2.09, 0.12, 69, 99, 25)

5. Conclusion. When solving atmospheric flow problems with operator splitting, stiff ODE integrations like the one discussed here must be carried out at thousands of grid points many times in succession. It is therefore of great practical interest to develop special purpose solvers which for this application run considerably faster than very efficient, general purpose codes like DASSL, of course without sacrificing accuracy and reliability. The preliminary results presented here show that our prototype solver, which as yet merely differs from a general purpose code in that a Gauss–Seidel iteration is applied instead of a Newton-type iteration, offers very good prospects for this purpose. An additional advantage of the Gauss–Seidel technique is that it reduces the storage requirements considerably. In large air pollution models the chemistry has to be carried out at thousands of grid points and, therefore, the

storage requirement can be a restrictive factor. We will therefore continue our efforts towards the development of a fast stiff ODE solver for chemically reacting atmospheric flow problems along the lines proposed in this paper.

Appendix: Description of the example problem. We give the chemical model with the reaction constants r_k and define the ODE system through the reaction rates v_k, the production terms P_k, and the loss terms L_k. Also a highly accurate reference solution at $t = 1$ and $t = 60$ minutes with the corresponding initial values is given. This reference solution shows the order of the chemical species as used in the ODE system. The units for the rate constants are min^{-1} for first order reactions and $ppm^{-1}min^{-1}$ for the second order ones.

The chemical reactions for the model example.			
01. NO2	→	NO + O3P	.350E+00
02. NO + O3	→	NO2	.266E+02
03. HO2 + NO	→	NO2 + OH	.120E+05
04. HCHO	→	2HO2 + CO	.860E–03
05. HCHO	→	CO	.820E–03
06. HCHO + OH	→	HO2 + CO	.150E+05
07. ALD	→	MEO2 + HO2 + CO	.130E–03
08. ALD + OH	→	C2O3	.240E+05
09. C2O3 + NO	→	NO2 + MEO2 + CO2	.165E+05
10. C2O3 + NO2	→	PAN	.900E+04
11. PAN	→	C2O3 + NO2	.220E–01
12. MEO2 + NO	→	CH3O + NO2	.120E+05
13. CH3O	→	HCHO + HO2	.188E+01
14. NO2 + OH	→	HNO3	.163E+05
15. O3P	→	O3	.480E+07
16. O3	→	O1D	.350E–03
17. O3	→	O3P	.175E–01
18. O1D	→	2OH	.100E+09
19. O1D	→	O3P	.444E+12
20. SO2 + OH	→	SO4 + HO2	.124E+04
21. NO3	→	NO	.210E+01
22. NO3	→	NO2 + O3P	.578E+01
23. NO2 + O3	→	NO3	.474E–01
24. NO3 + NO2	→	N2O5	.178E+04
25. N2O5	→	NO3 + NO2	.312E+01

$$v_1 = r_1 y_1,\ v_2 = r_2 y_2 y_4,\ v_3 = r_3 y_5 y_2,\ v_4 = r_4 y_7,\ v_5 = r_5 y_7,\ v_6 = r_6 y_7 y_6,\ v_7 = r_7 y_9,$$

$$v_8 = r_8 y_9 y_6,\ v_9 = r_9 y_{11} y_2,\ v_{10} = r_{10} y_{11} y_1,\ v_{11} = r_{11} y_{13},\ v_{12} = r_{12} y_{10} y_2,\ v_{13} = r_{13} y_{14},$$

$$v_{14} = r_{14} y_1 y_6,\ v_{15} = r_{15} y_3,\ v_{16} = r_{16} y_4,\ v_{17} = r_{17} y_4,\ v_{18} = r_{18} y_{16},\ v_{19} = r_{19} y_{16},$$

$$v_{20} = r_{20} y_{17} y_6,\ v_{21} = r_{21} y_{19},\ v_{22} = r_{22} y_{19},\ v_{23} = r_{23} y_1 y_4,\ v_{24} = r_{24} y_{19} y_1,\ v_{25} = r_{25} y_{20};$$

$$P_1 = v_2 + v_3 + v_9 + v_{11} + v_{12} + v_{22} + v_{25},\ P_2 = v_1 + v_{21},\ P_3 = v_1 + v_{17} + v_{19} + v_{22},$$

$$P_4 = v_{15},\ P_5 = v_4 + v_4 + v_6 + v_7 + v_{13} + v_{20},\ P_6 = v_3 + v_{18} + v_{18},\ P_7 = v_{13},$$

$$P_8 = v_4 + v_5 + v_6 + v_7,\ P_9 = 0.0,\ P_{10} = v_7 + v_9,\ P_{11} = v_8 + v_{11},\ P_{12} = v_9,\ P_{13} = v_{10},$$

$$P_{14} = v_{12},\ P_{15} = v_{14},\ P_{16} = v_{16},\ P_{17} = 0.0,\ P_{18} = v_{20},\ P_{19} = v_{23} + v_{25},\ P_{20} = v_{24};$$

$$L_1 = r_1 + r_{10} y_{11} + r_{14} y_6 + r_{23} y_4 + r_{24} y_{19},\ L_2 = r_2 y_4 + r_3 y_5 + r_9 y_{11} + r_{12} y_{10},$$

$$L_3 = r_{15},\ L_4 = r_2 y_2 + r_{16} + r_{17} + r_{23} y_1,\ L_5 = r_3 y_2,\ L_6 = r_6 y_7 + r_8 y_9 + r_{14} y_1 + r_{20} y_{17},$$

$$L_7 = r_4 + r_5 + r_6 y_6,\ L_8 = 0.0,\ L_9 = r_7 + r_8 y_6,\ L_{10} = r_{12} y_2,\ L_{11} = r_9 y_2 + r_{10} y_1,$$

$$L_{12} = 0.0,\ L_{13} = r_{11},\ L_{14} = r_{13},\ L_{15} = 0.0,\ L_{16} = r_{18} + r_{19},\ L_{17} = r_{20} y_6,$$

$$L_{18} = 0.0,\ L_{19} = r_{21} + r_{22} + r_{24} y_1,\ L_{20} = r_{25}.$$

Concentration values in ppm at, respectively, $t = 0$ min., $t = 1$ min., and $t = 60$ min.				
1.	[NO2]	0	0.37326304298606E–01	0.56462554800124E–01
2.	[NO]	0.2	0.16251325412704E+00	0.13424841304232E+00
3.	[O3P]	0	0.27344389305923E–08	0.41397343310918E–08
4.	[O3]	0.04	0.32994065756620E–02	0.55231402074779E–02
5.	[HO2]	0	0.31151619385738E–06	0.20189772623092E–06
6.	[OH]	0	0.26534918500427E–06	0.14645418635004E–06
7.	[HCHO]	0.1	0.99423103666384E–01	0.77842491190039E–01
8.	[CO]	0.3	0.30061731277555E+00	0.32450753533953E+00
9.	[ALD]	0.01	0.99269949383466E–02	0.74940133838905E–02
10.	[MEO2]	0	0.29529601825322E–07	0.16222931573089E–07
11.	[C2O3]	0	0.20994901153721E–07	0.11358638332624E–07
12.	[CO2]	0	0.65714929538122E–04	0.22305059757112E–02
13.	[PAN]	0	0.59742964642105E–05	0.20871628827982E–03
14.	[CH3O]	0	0.27858639499927E–04	0.13969210168610E–04
15.	[HNO3]	0	0.13959464032840E–03	0.89648848569121E–02
16.	[O1D]	0	0.26002979092156E–17	0.43528463693250E–17
17.	[SO2]	0.007	0.69973974657447E–02	0.68992196962640E–02
18.	[SO4]	0	0.26025342552954E–05	0.10078030373603E–03
19.	[NO3]	0	0.38171954506700E–06	0.17721465139664E–05
20.	[N2O5]	0	0.72454590089901E–05	0.56829432922952E–04

REFERENCES

[1] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, North-Holland, Amsterdam, 1989.

[2] P. N. Brown, G. D. Byrne, and A. C. Hindmarsh, *VODE: A variable step ODE solver*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1038–1051.

[3] G. D. Byrne, *The taming of a co-polymerization problem with VODE*, IMPACT of Computing in Science and Engineering, 5 (1993), pp. 318–344.

[4] G. D. Byrne and A. M. Dean, *The numerical solution of some chemical kinetics models with VODE and CHEMKIN II*, Computers Chem., 17 (1993), pp. 297–302.

[5] J. J. Dongarra and E. Grosse, *Distribution of software via electronic mail*, Comm. ACM, 30 (1987), pp. 403–407. (Available through netlib@research.att.com.)

[6] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II. *Stiff and Differential-Algebraic Problems*, Springer-Verlag, Berlin, 1991.

[7] A. C. Hindmarsh, *LSODE and LSODI, two new initial value ordinary differential equation solvers*, ACM-SIGNUM Newsletter, 15 (1980), pp. 10–11.

[8] F.A.A.M. de Leeuw, *Numerical Solution of Ordinary Differential Equations Arising from Chemical Kinetics*, Report 228603005, National Institute of Public Health and Environmental Protection (RIVM), Bilthoven, The Netherlands, 1988.

[9] J. G. Verwer and M. van Loon, *An Evaluation of Explicit Pseudo-Steady State Approximation Schemes for Stiff ODE Systems from Chemical Kinetics*, CWI Report NM-R9312, Centre for Mathematics and Computer Science, Amsterdam, 1993; J. Comput. Phys., to appear.

[10] Z. Zlatev and J. Wasniewsky, *Large Scale Computations in Air Pollution Modelling*, Report UNIC-92-05, Scientific Computing Group, Danmarks EDB-Center for Forskning og Uddannelse, 1992.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1251–1279, November 1994

QUASI-RANDOM SEQUENCES AND THEIR DISCREPANCIES*

WILLIAM J. MOROKOFF† AND RUSSEL E. CAFLISCH†

Abstract. Quasi-random (also called low discrepancy) sequences are a deterministic alternative to random sequences for use in Monte Carlo methods, such as integration and particle simulations of transport processes. The error in uniformity for such a sequence of N points in the s-dimensional unit cube is measured by its discrepancy, which is of size $(\log N)^s N^{-1}$ for large N, as opposed to discrepancy of size $(\log\log N)^{1/2} N^{-1/2}$ for a random sequence (i.e., for almost any randomly chosen sequence). Several types of discrepancies, one of which is new, are defined and analyzed. A critical discussion of the theoretical bounds on these discrepancies is presented. Computations of discrepancies are presented for a wide choice of dimension s, number of points N, and different quasi-random sequences. In particular for moderate or large s, there is an intermediate regime in which the discrepancy of a quasi-random sequence is almost exactly the same as that of a randomly chosen sequence. A simplified proof is given for Woźniakowski's result relating discrepancy and average integration error, and this result is generalized to other measures on function space.

Key words. Monte Carlo, quasi-random, discrepancy, Brownian sheet

AMS subject classifications. 65C05, 10K30

1. Introduction.

Since the beginning of the computer age, Monte Carlo methods have been used to evaluate integrals, solve integral equations, and simulate physical processes [7]. These methods use a sequence of points, usually a deterministic pseudo-random approximation to a randomly chosen sequence, to sample the values of the integrand function or the possible steps in a process. Over the years a number of techniques, such as variance reduction through stratification, have been developed to improve the accuracy of these methods. An alternative technique is to replace the pseudo-random sequence with a deterministic sequence having better uniformity properties.

Uniformity of a sequence is measured by its discrepancy, which is the error in representation of the volume of subsets of the unit cube by the fraction of points in the subsets. Several different definitions of discrepancy can be formulated [8], [14], [17], including a sup over rectangles or an L_1 or L_2 integral over rectangles using either all rectangles or only those with one vertex at the origin. Integration error can be related to discrepancy either through the Koksma–Hlawka inequality [8], [14], [17] or Woźniakowksi's identity [26], which states that the discrepancy is equal to the average integration error with respect to the Brownian sheet measure. We present a critical discussion of the various definitions of discrepancy, one of which is new, and a simplified proof of Woźniakowksi's result is given, which allows the result to be generalized to other measures on function space.

A quasi-random (or low discrepancy) sequence in the s-dimensional cube is a sequence for which the discrepancy is roughly of size $(\log N)^s N^{-1}$ for large N, which is the minimum size possible. These sequences are more uniform than random sequences because randomly chosen points tend to clump, leading to discrepancy of size $(\log\log N)^{1/2} N^{-1/2}$. Evidence of this clumping is shown in a planar projection of a pseudo-random sequence in Fig. 12; while the top graph of Fig. 13 shows the uniformity that can be achieved with quasi-random points. At the other extreme, regular lattices of points work well in low dimension, but in high dimension they are not very useful. Points cannot be added to a lattice incrementally. Instead, a given s-dimensional lattice can only be refined by increasing the number of points by a factor 2^s; i.e., the discrepancy of a lattice is of size $O(1)$, except at special values of N

*Received by the editors July 17, 1991; accepted for publication (in revised form) August 30, 1993. Research was supported in part by the Air Force Office of Scientific Research grant AFOSR 90-0003.

†Mathematics Department, University of California Los Angeles, Los Angeles, California 90024-1555 (morokoff@math.ucla.edu, caflisch@math.ucla.edu).

at which the lattice is completely refined. Moreover for large s it is usually impossible to put down enough lattice points to get good resolution.

Quasi-random sequences combine the advantage of a random sequence (points can be added incrementally) with the advantage of a lattice (no clumping of points). Examples of such sequences that will be considered below include the Halton sequence, Sobol' sequence, and Faure sequence. Bounds on the discrepancy of these sequences, as well as other analytic properties, have been previously derived using number theoretic techniques [8], [14]–[17]. An alternative method for generating quasi-random sequences and bounds on integration error using a dynamical systems approach is presented in [19].

The main portion of this paper consists of computations and critical discussion of the discrepancy for these sequences over a large range of values of N and s. In particular for large dimension s, the theoretical bound $(\log N)^s N^{-1}$ is only meaningful for extremely large values of N; i.e., $N = O(e^s)$.

In an attempt to directly understand the uniformity properties of quasi-random sequences in high dimension, two-dimensional projections are presented for a variety of quasi-random sequences. These can show considerable clumping in high dimension. Finally we present timing results for the generation of the different quasi-random sequences.

Previous computational studies of quasi-random sequences (as well as scrambled quasi-random sequences) and their discrepancy by Braaten and Weller [1], Bratley and Fox [2], Bratley, Fox, and Niederreiter [3], Levitan [9], and Pages and Xiao [20] presented some useful, but less complete, results. Some of their results were due to transient effects, such as those described below for the T_N^* discrepancy, and some of the multidimensional integration tests were only performed for product functions. Sarkar and Prasad [22] have given a comparison of Halton, scrambled Halton, and Faure sequences as applied to an absorption problem. In a cogent article, Press and Teukolsky [21] discussed the Sobol' sequences and computational methods for generating them and showed how discontinuous integrand functions decrease the effectiveness of Monte Carlo integration with quasi-random sequences. In two companion papers [12], [13] we present computational studies and some analysis for quasi-Monte Carlo methods applied to integration and simulation of some simple transport processes.

2. Discrepancy and integration error for quasi-random sequences. Given that pseudo-random sequences work well in Monte Carlo integration, it seems reasonable to ask if other deterministic sequences might also work. More precisely, it seems that the independence of random numbers plays a secondary role to their uniformity in Monte Carlo calculations; so sequences with better uniformity properties may lead to smaller errors. In order to develop this idea it is necessary to define a uniform sequence and some measure of its uniformity. The following is based on Niederreiter's development of the topic in [14].

Let I^s denote the s-dimensional unit cube. An infinite sequence $\{x_n\}$ in I^s is called uniformly distributed if for all Jordan measurable subsets J of I^s,

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathcal{X}_J(\boldsymbol{x}_n) = m(J)$$

holds, where $\mathcal{X}_J$ is the characteristic function of J, and $m(J)$ is the volume of J. Thus in the limit of an infinite number of points, every region in I^s has proportionally the right number of points. From this definition it follows that a sequence $\{\boldsymbol{x}_n\}$ is uniformly distributed if for all Riemann integrable functions f defined on I^s it holds that

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{x}_n) = \int_{I^s} f(\boldsymbol{x})\, d\boldsymbol{x}\,.$$

It follows from the Central Limit Theorem that a sequence of independent, random points chosen from the interval I^s with probability density one is indeed a uniformly distributed sequence with probability one.

Practically, it is only possible to deal with a finite number of integration nodes, so it is necessary to define some measure of uniformity for finite point sets. Such a quantity is known as discrepancy. For a set $J \subseteq I^s$ and a sequence of N points $\{\boldsymbol{x}_n\}$ in I^s, define

$$R_N(J) = \frac{1}{N} \sum_{n=1}^{N} \chi_J(\boldsymbol{x}_n) - m(J) .$$

Various kinds of discrepancies can be defined then by restricting J to a certain class of sets and taking a norm of R_N over this class. If E is the set of all subrectangles of I^s, then the L_∞ and L_2 norms are defined as

$$D_N = \sup_{J \in E} | R_N(J) |, \tag{1}$$

$$T_N = \left[\int_{(\boldsymbol{x},\boldsymbol{y}) \in I^{2s}, x_i < y_i} (R_N(J(\boldsymbol{x},\boldsymbol{y})))^2 \, d\boldsymbol{x} \, d\boldsymbol{y} \right]^{\frac{1}{2}} . \tag{2}$$

Here $J(\boldsymbol{x},\boldsymbol{y})$ indicates the rectangle with opposite corners at $(\boldsymbol{x},\boldsymbol{y})$. If E^* is the set of subrectangles with one corner at $\mathbf{0}$, then the star discrepancies are defined as

$$D_N^* = \sup_{J \in E^*} | R_N(J) |, \tag{3}$$

$$T_N^* = \left[\int_{I^s} (R_N(J(\boldsymbol{x})))^2 \, d\boldsymbol{x} \right]^{\frac{1}{2}} . \tag{4}$$

Here $J(\boldsymbol{x})$ is the rectangle with a corner at $\mathbf{0}$ and a corner at $\boldsymbol{x}$. It should be noted that this is not the standard notation as used by Niederreiter [14] and others. In the past, T_N has denoted what is here referred to as T_N^*. The new notation is necessary because the L_2 discrepancy over all rectangles T_N had not been previously defined nor used. To be consistent with the sup discrepancy, it makes sense to relabel the original L_2 discrepancy over rectangles with a corner at zero as T_N^*, and call the new L_2 discrepancy T_N. Another kind of discrepancy, J_N, is obtained by taking the sup over all convex sets. No L_2 analog exists for this class. The infinite sequence $\{\boldsymbol{x}_n\}$ being uniformly distributed is equivalent to $\lim_{N\to\infty} D_N = 0$, where D_N refers to the discrepancy of the first N terms of the sequence. The statement is true for all of the above discrepancies.

The importance of discrepancy as an error bound for Monte Carlo integration can be seen from the Koksma–Hlawka inequality, which in one dimension for smooth functions f of bounded variation reads

$$\varepsilon(f) = \left| \int_0^1 f(x) \, dx - \frac{1}{N} \sum_{n=1}^{N} f(x_n) \right| \leq V(f) \, D_N^*, \tag{5}$$

where D_N^* is the discrepancy of the sequence $\{x_n\}$ and $V(f) = \int_0^1 |df|$ is the variation of f.

Inequality (5) can be extended to higher dimensions; however, the definition of variation must be modified. Assume for the moment that f is sufficiently smooth on I^s so that

$$V^{(s)}(f) = \int_{I^s} \left| \frac{\partial^s f}{\partial t_1 \ldots \partial t_s} \right| dt$$

exists. Define for all $k \leq s$ and all sets of k integers $1 \leq i_1 < i_2 < \cdots < i_k \leq s$ the quantity

$$V^{(k)}(f; i_1, \ldots, i_k) = \int_{I^k} \left| \frac{\partial^k f}{\partial t_{i_1} \ldots \partial t_{i_k}} \right|_{t_j = 1, j \neq i_1, \ldots, i_k} dt_{i_1} \ldots dt_{i_k} .$$

Similarly define $D_N^*(i_1, \ldots, i_k)$ to be the star discrepancy of the orthogonal projection of the sequence $\{\boldsymbol{x}_n\}$ on to the appropriate k-dimensional subspace of I^s. Then

$$\left| \int_{I^s} f(\boldsymbol{x})\, d\boldsymbol{x} - \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{x}_n) \right| \leq \sum_{k=1}^{s} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k)\, D_N^*(i_1, \ldots, i_k) .$$

It is possible to obtain a similar bound involving $T_N^*(i_1, \ldots, i_k)$ and an L_2 version of $V^{(k)}$ defined by

$$W^{(k)}(f; i_1, \ldots, i_k) = \left[\int_{I^k} \left(\frac{\partial^k f}{\partial t_{i_1} \ldots \partial t_{i_k}} \Big|_{t_j = 1, j \neq i_1, \ldots, i_k} \right)^2 dt_{i_1} \ldots dt_{i_k} \right]^{\frac{1}{2}} .$$

This requires a stronger condition on the integrand f than just being of L_1 bounded variation. The resulting bound is

$$\left| \int_{I^s} f(\boldsymbol{x})\, d\boldsymbol{x} - \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{x}_n) \right| \leq \sum_{k=1}^{s} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s} W^{(k)}(f; i_1, \ldots, i_k)\, T_N^*(i_1, \ldots, i_k) .$$

This argument may be made rigorous for a broader class of f known as functions of bounded variation in the sense of Hardy and Krause. See [14] for details.

It is easy to see that $D_N^* \geq D_N^*(i_1, \ldots, i_k)$ for all $k \leq s$. If the variation of f in the sense of Hardy and Krause is defined as

$$V(f) = \sum_{k=1}^{s} \sum_{1 \leq i_1 < i_2 < \cdots < i_k \leq s} V^{(k)}(f; i_1, \ldots, i_k) ,$$

then inequality (5) follows immediately. This relationship does not follow when the T_N^* discrepancy is used; in fact, in our computational tests, $T_N^* \leq T_N^*(i_1, \ldots, i_k)$.

The inequality (5) shows that if sequences exist with lower discrepancy than random sequences, better than random convergence may be possible. Several such low discrepancy sequences are discussed in the next section. However, first it is helpful to examine some of the basic properties of discrepancy.

For a random sequence it can be shown from the law of iterated logarithms that

$$D_N = \mathcal{O}\left(\left(\frac{\log \log N}{N} \right)^{\frac{1}{2}} \right)$$

with probability one (see [14, p. 971]). This is true for any dimension s. Calculations carried out in §6 show that $E(T_N^2) = C_s/N$, where $E(\cdot)$ is the expectation, taken to be an integral over the space I^{sN} of possible values of the s coordinates of the N random points. C_s is a constant depending on dimension. Both of these estimates show $N^{-.5}$ type convergence, which corresponds to the standard Monte Carlo error behavior for integration with random nodes.

Various relationships exist among the different notions of discrepancy, which allow a sequence to be termed low discrepancy without specifying the measure. Perhaps the simplest relationship is

$$D_N^* \leq D_N \leq 2^s D_N^* .$$

This is clear from the fact that E^* is a subset of E, while any set in E can be written as a combination of 2^s sets in E^*. From the basic fact that L_∞ norms are larger than L_2 norms, it follows that

$$T_N^* \leq D_N^* .$$

The relationship between T_N^* and T_N is discussed in §5; in general, the star discrepancy is larger. In [16] Niederreiter establishes the relationship

$$C_s (D_N)^{\frac{s+2}{2}} \leq T_N^* .$$

In his book with Kuipers [8] he also shows that the isotropic discrepancy over convex sets satisfies

$$J_N \leq 4s(D_N)^{\frac{1}{s}} .$$

This bound is improved in [13] under the assumption suggested by Press and Teukolsky [21].

Other properties of discrepancy include the lower bound established by Roth and discussed in [8];

$$T_N^* > C_s \frac{(\log N)^{\frac{s-1}{2}}}{N} .$$

Halton [6] showed the existence of infinite sequences in any dimension which satisfy

$$D_N = \mathcal{O}\left(\frac{(\log N)^s}{N}\right) .$$

This bound is regarded as the best possible. This is an important result because it offers hope that the standard Monte Carlo $N^{-1/2}$ convergence can be improved considerably. Sequences with this property are the topic of the §4.

3. Average integration error. A direct relation between the integration error $\varepsilon(f)$ and the L^2 discrepancy has been derived by Woźniakowski [26]. He showed that $(T_N^*)^2$ is equal to the average integration error, i.e.,

$$(T_N^*)^2 = E(\varepsilon(f)^2) \tag{6}$$

in which $\varepsilon(f)$ denotes the integration error on the left side of (5) above. The average is taken with respect to the "Brownian sheet" measure, which is a generalization of Brownian motion with s-dimensional "time." In particular the measure is concentrated on functions that are roughly "half-differentiable" (i.e., they have Hölder exponent nearly equal to $\frac{1}{2}$), so that they have infinite variation. This shows that the Koksma–Hlawka inequality (5) is a vast overestimate, at least for this class of functions.

The Brownian sheet measure is a measure on function space. It is a natural generalization of the simple Brownian motion $b(x)$ to multidimensional "time" $\boldsymbol{x}$. Denote $\boldsymbol{x}' = (x_i')_{i=1}^s$ in which $x_i' = 1 - x_i$ and (with the usual abuse of notation) denote $f(\boldsymbol{x}') = f(\boldsymbol{x})$; also

denote the finite difference operator $D_i f(\boldsymbol{x}') = f(\boldsymbol{x}' + \Delta_i \hat{e}_i) - f(\boldsymbol{x}')$ in which $\hat{e}_i$ is the ith coordinate vector. The Brownian sheet is based at the point $\boldsymbol{x}' = 0$, i.e., $\boldsymbol{x} = (1, \ldots, 1)$, and has $f(\boldsymbol{x}' = 0) = f(1, \ldots, 1) = 0$. For any point $\boldsymbol{x}'$ in I^s and any set of positive lengths Δ_i (with $x_i' + \Delta_i < 1$), the multidimensional difference $D_1 \ldots D_s f(\boldsymbol{x})$ is a normally distributed random variable with mean zero and variance

$$E((D_1 \ldots D_s f(\boldsymbol{x}))^2) = \Delta_1 \ldots \Delta_s. \tag{7}$$

This implies that

$$E(df(\boldsymbol{x})df(\boldsymbol{y})) = \delta(\boldsymbol{x} - \boldsymbol{y})d\boldsymbol{x}d\boldsymbol{y} \tag{8}$$

in which df is understood in the sense of the Ito calculus [10]. Moreover $f(\boldsymbol{x}') = 0$ if $x_i' = 0$ for any i, and for any $\boldsymbol{x}$ in I^s, $f(\boldsymbol{x})$ is normally distributed with mean zero and variance

$$E(f(\boldsymbol{x})^2) = \prod_{i=1}^{s} x_i'. \tag{9}$$

The Brownian sheet has the same covariance properties as the product of independent Brownian motions $\tilde{f}(\boldsymbol{x}) = \prod_{i=1}^{s} x_i b_i(x_i)$, but $f \neq \tilde{f}$ since the product is not normally distributed.

The derivation of (6) in [26] was simply a calculation of each side of the equation. Here we present a new derivation that follows naturally the properties of the Brownian sheet measure. First rewrite the integration error $E(f)$ using integration by parts, following the steps of the proof of the Koksma–Hlawka inequality [14]. Note that

$$dR(\boldsymbol{x}) = \left\{ \frac{1}{N} \sum_{n=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_n) - 1 \right\} d\boldsymbol{x} \tag{10}$$

in which $R(\boldsymbol{x}) = R_N(J(\boldsymbol{x}))$ as defined in §2. Also $R(\boldsymbol{x}) = 0$ if $x_i = 0$ and $f(\boldsymbol{x}) = 0$ if $x_i = 1$ for any i, which implies that the boundary terms all disappear in the following integration by parts:

$$\begin{aligned} \varepsilon(f) &= \left| \int_{I^s} f(\boldsymbol{x})d\boldsymbol{x} - \frac{1}{N} \sum_{n=1}^{N} f(\boldsymbol{x}_n) \right| \\ &= \left| \int_{I^s} \left\{ 1 - \frac{1}{N} \sum_{n=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_n) \right\} f(\boldsymbol{x})d\boldsymbol{x} \right| \\ &= \left| \int_{I^s} R(\boldsymbol{x})df(\boldsymbol{x}) \right|. \end{aligned}$$

The quantity df in this identity is defined here through the Ito calculus, even though $V(f) = \infty$ with probability one.

It follows from (8) that the average square error is

$$\begin{aligned} E(\varepsilon(f)^2) &= E\left(\left(\int_{I^s} R(\boldsymbol{x})df(\boldsymbol{x}) \right)^2 \right) \\ &= \int_{I^s \times I^s} R(\boldsymbol{x})R(\boldsymbol{y})E(df(\boldsymbol{x})df(\boldsymbol{y})) \\ &= \int_{I^s} R(\boldsymbol{x})^2 d\boldsymbol{x} \\ &= (T_N^*)^2. \end{aligned} \tag{11}$$

One unnatural feature of the Brownian sheet measure used above is that the functions f are all required to vanish on the boundaries $x_i' = 0$ (i.e., $x_i = 1$) for all i. This restriction can be removed by a generalization of the Brownian sheet that puts the values on the boundaries $x_i' = 0$ to be generalized Brownian sheets.

First set $f(\boldsymbol{x}' = 0) = f(1, \ldots, 1) = 0$. Then on the s coordinate lines emanating from the origin, let f be given by s independent Brownian motions. Next on the two-dimensional boundaries, where all but two of the x_i' are zero, define f by the Brownian sheet property with the given boundary conditions on the two sides. Continue this procedure until all boundaries and finally the interior have been defined. The resulting measure still satisfies the equation (8) and has the following covariance:

$$(E(f(\boldsymbol{x}')f(\boldsymbol{y}')) = \prod_{i=1}^{s} \min(x_i', y_i') + \sum_{j=1}^{s} \prod_{i\neq j, i=1}^{s} \min(x_i', y_i') + \cdots + \sum_{i=1}^{s} \min(x_i', y_i'). \tag{12}$$

In particular note that

$$E(df(\boldsymbol{x}')f(\boldsymbol{y}')) = 0 \tag{13}$$

if $\boldsymbol{x}'$ is an interior points of I^s and $y_i' = 0$ for some i.

Now compute the average integration error as before. The boundary terms do not vanish but are all independent of each other. Use integration by parts to find that the integration error is

$$\begin{aligned} \varepsilon &= \left| \int_{I^s} f(\boldsymbol{x}) dR(\boldsymbol{x}) \right| \\ &= \left| \int_{I^s} R(\boldsymbol{x}) df(\boldsymbol{x}) + \sum_{i=1}^{s} \int_{I^s \cap x_i = 1} f(\boldsymbol{x}) dR(\boldsymbol{x}) \right| \end{aligned}$$

since $R(\boldsymbol{x}) = 0$ if $x_i = 0$ for any i. The first term and the sum are independent according to (13); also integration by parts can be performed on each term in the sum. The result is similar to the covariance equation (12); i.e.,

$$\begin{aligned} E(\varepsilon^2) &= \int_{I^s}\int_{I^s} R(\boldsymbol{x})R(\boldsymbol{y})E(df(\boldsymbol{x})df(\boldsymbol{y})) \\ &\quad + \sum_{i=1}^{s} \int_{I^s \cap x_i = 1} \int_{I^s \cap y_i = 1} R(\boldsymbol{x})R(\boldsymbol{y})E(df(\boldsymbol{x})df(\boldsymbol{y})) \\ &\quad + \cdots + \sum_{i=1}^{s} \int_{0<x_i<1, x_j=1 \text{ if } j\neq i} \int_{0<x_i<1, y_j=1 \text{ if } j\neq i} R(\boldsymbol{x})R(\boldsymbol{y})E(df(\boldsymbol{x})df(\boldsymbol{y})) \qquad (14) \\ &= \int_{I^s} R(\boldsymbol{x})^2 d\boldsymbol{x} + \sum_{i=1}^{s} \int_{I^s \cap x_i = 1} R(\boldsymbol{x})^2 d\boldsymbol{x} + \cdots + \sum_{i=1}^{s} \int_{0<x_i<1, x_j=1 \text{ if } j\neq i} R(\boldsymbol{x})^2 d\boldsymbol{x} \\ &= T_n^* + \sum_{i=1}^{s} T_n^*(i) + \cdots + \sum_{i=1}^{s} T_n^*(1, \ldots, i-1, i+1, \ldots, s) \end{aligned}$$

in which $T_n^*(i_1, \ldots, i_k)$ is the L_2^* discrepancy for the sequence projected onto the boundary $x_j = 1$ for $j = i_1, \ldots, i_k$.

The point 0 still plays a special role in this measure. A still more uniform measure would be to center the Brownian sheet at a random point $\boldsymbol{y}$ inside I^s. In this case the first term in $E(\varepsilon^2)$ is just the L^2 discrepancy T_N (without *), but there are correlations between the boundary terms and the interior, as well as between different boundary terms, so that no simple equation results.

4. Low discrepancy sequences. Numerous sequences have been shown to have $\mathcal{O}((\log N)^s/N)$ behavior for their discrepancies. Included among these are sequences of the form

$$\boldsymbol{x}_n = ([n\alpha_1], \ldots, [n\alpha_s])$$

where the α_i are irrational numbers, which are linearly independent over the rationals, and $[\cdot]$ denotes the fractional portion of the number. Another sequence that has been suggested involves using a pseudo-random generator. If $\boldsymbol{x}_1 = (\rho_1, \ldots, \rho_s)$ is a random point, then the next term in the quasi-random sequence is given by $\boldsymbol{x}_2 = (\rho_2, \ldots, \rho_s, \rho_{s+1})$, where ρ_{s+1} is the next number produced by the generator. Various techniques have been applied to determine the discrepancy bounds for these sequences.

The sequences that have generated the most attention, and which have been studied in the current research, are those based on the p-adic expansion of the integers. For any integer n, let $(n)_p = a_k a_{k-1} \ldots a_0$ be the base p expansion of n with $0 \le a_i < p$. Define

$$S_p(n) = \frac{a_0}{p} + \frac{a_1}{p^2} + \cdots + \frac{a_k}{p^{k+1}}.$$

Then $0 < S_p(n) < 1$ for all n, and the sequence $S_p(n)$ is a one-dimensional uniformly distributed sequence. For $p = 2$ this is known as the van der Corput sequence. An s-dimensional generalization of this sequence $\{\boldsymbol{x}_n\}$, known as the Halton sequence, is given by

$$\boldsymbol{x}_n = (S_{p_1}(n), \ldots, S_{p_s}(n)),$$

where $(p_1, \ldots, p_s)$ are relatively prime integers, usually taken to be the first s primes. Several authors have derived bounds for the discrepancy of the Halton sequence. Meijer [11] shows that

$$D_N^* \le C_s^H \frac{(\log N)^s}{N} + \mathcal{O}\left(\frac{(\log N)^{s-1}}{N}\right) \tag{15}$$

where

$$C_s^H = \prod_{k=1}^{s} \frac{p_k - 1}{2 \log p_k}. \tag{16}$$

An unfortunate aspect of this bound is that the constant in the leading term grows super-exponentially with dimension. The difficulties of the Halton sequence in high dimension are discussed further in §6 and in [13].

The Sobol' [23] and the Faure [4] sequences are also based, at least indirectly, on p-adic expansions of the integers. Niederreiter [17], [18] has developed and expanded a general theory of (t, s)-sequences, which encompasses the theory behind both of these sequences. The Sobol' sequence is an example of a (t, s)-sequence with $p = 2$ independent of s, and with t growing with s. The Faure sequence is another example, but it requires that $t = 0$ by setting $p = p(s)$, where $p(s)$ is the smallest prime greater than or equal to s. In each case there is a discrepancy bound equivalent to (15). For Sobol' the coefficient of the $(\log N)^s N^{-1}$ term takes the form

$$C_s^S = \frac{2^t}{s!(\log 2)^s}.$$

Sobol' [23] gives the bound for $t = t(s)$ in the bound for the Sobol' sequence as

$$K \frac{s \log s}{\log \log s} \le t(s) \le \frac{s \log s}{\log 2} + \mathcal{O}(s \log \log s),$$

which shows that $t(s)$ grows super-linearly. Like the Halton bound, this constant grows super-exponentially with s, although it is not nearly as large as the Halton constant. For the Faure sequence, the coefficient can be written

$$C_s^F = \frac{1}{s!}\left(\frac{p(s)-1}{2\log p(s)}\right)^s .$$

This has the desirable property that $\lim_{s\to\infty} C_s = 0$. As Faure's calculations show [4], C_s^F is smaller than both C_s^S and C_s^H, and it goes to zero as dimension increases while the others go to infinity. Because of this smaller bound, it has been claimed that the Faure sequence is superior. A comparison of these sequences in actual computation is made in §6 below.

The actual construction of these sequences is rather complicated, and it is best to check the papers of Sobol', Faure, and Niederreiter for a complete description. Press and Teukolsky [21] and Bratley and Fox [2] give detailed descriptions of the implementation of the Sobol' sequence. For periodic integrands, the method of good lattice points, described in [14], also offers promise, although this has not been studied in the current work.

5. Theoretical bounds on discrepancy. The Halton, Sobol', and Faure sequences discussed in §4 are now studied in detail. First the nature of the error bounds for discrepancy of these sequences is examined. Then actual calculations of the L_2 discrepancies are presented as a means of comparison and as a method of predicting performance in integration. This is followed by a discussion of certain properties of the sequences that are revealed through studying the two-dimensional orthogonal projections. Finally some computational aspects of the sequences are examined, and recommendations are made for their use.

As described in §4, the Halton, Sobol', and Faure sequences all have discrepancy bounds of the form

$$D_N^* \le C_s \frac{(\log N)^s}{N} + \mathcal{O}\left(\frac{(\log N)^{s-1}}{N}\right) .$$

The difficulty with basing any conclusions on this bound for discrepancy is illustrated through the following considerations. Only the bound for the Faure sequence will be considered, as it is the smallest. Let $b_F^s(N) = C_s^F(\log N)^s N^{-1}$ denote the leading term of the bound on the Faure sequence including the constant given above. The best way to examine the behavior of discrepancy with respect to N is to consider a log log plot. Thus let $x = \log N$, which gives

$$\log(b_F^s(N)) = \log(C_s^F) + s\,\log x - x .$$

This function has a maximum, which can be found by setting the derivative

$$\frac{d\log(b_F^s(N))}{dx} = \frac{s}{x} - 1$$

equal to zero. Then the maximum occurs at $x = s$, or when $N = e^s$. Because the general trend of discrepancy for a uniform sequence should be to decrease with increasing N, it follows that the bound cannot be a useful measure of performance until after its maximum has been attained. Thus in high dimensions, the bound gives no information until a very large number of points is used. Moreover, in order to get the same rate of decay of error as with random numbers, $N = e^{2s}$ points are required. The bound has a rate of convergence of $N^{-.95}$ only when $N = e^{20s}$. Even in low dimensions, an extraordinary number of points is required for the bound to indicate near $\frac{1}{N}$ performance.

Not only is the convergence rate predicted by the bound somewhat questionable, but the actual value of the bound is rather large, and grows with dimension, despite the fact that

C_s^F goes to zero. Figure 1 shows the growth as a function of dimension of $\log(b_F^s(N))$ at $N = e^s$ where it attains its maximum. As discrepancy is bounded by one, and thus the log of discrepancy must be negative, a large positive value for the log of the bound is another indication that for N near the maximum, the bound is not accurate.

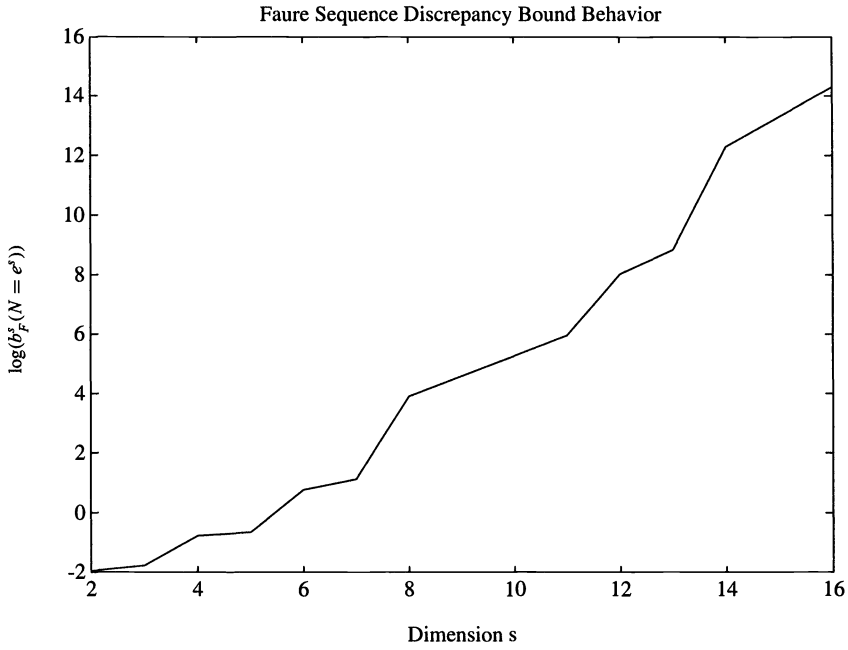


FIG. 1. *Maximum value of discrepancy bound.*

It should be noted that for fixed N, $b_F^s(N)$ is also an increasing function of s until it achieves its maximum at a value of s which is somewhat larger than N (asymptotically in N, the maximum occurs at $s = N^{e/2}$). Thus for fixed N the leading order term does go to zero as dimension increases, but only after passing through a large maximum whose value is super-exponential in N.

It should also be noted that the influence of the terms other than the leading order one has been neglected here. The effect of including these extra terms on the converge rate in N is minor, at most a factor of $\log(N)/(\log(N) - 1)$. However, the leading order constant C_s may be an underestimate. This further illustrates the inadequacies of the discrepancy bound.

Figure 2 shows a plot of $\log b_F^s(N)$ as a function of $\log N$ for dimensions 4 and 16. Also plotted is the actual value of $\log T_N$ for each sequence. The calculation of this quantity is described in the next section. These graphs show that even in the lower dimension, the bound does not accurately predict the behavior of the L_2 discrepancy in slope or in magnitude. Of course the bound was derived for D_N^* and not T_N, and the two measures of discrepancy do not necessarily have the same convergence properties. However, both measure the uniformity of a sequence and thus the convergence of T_N should be related to how well the sequence performs in practice. The difference becomes even greater as dimension increases, as illustrated by the plot for 16 dimensions. This may be somewhat deceptive because D_N^* increases with dimension, while T_N decreases; however, both are still bounded by one. No bound has been specifically derived for T_N, but it is worthwhile to consider this quantity, because as described next, it seems to indicate what one can expect from actual calculations.

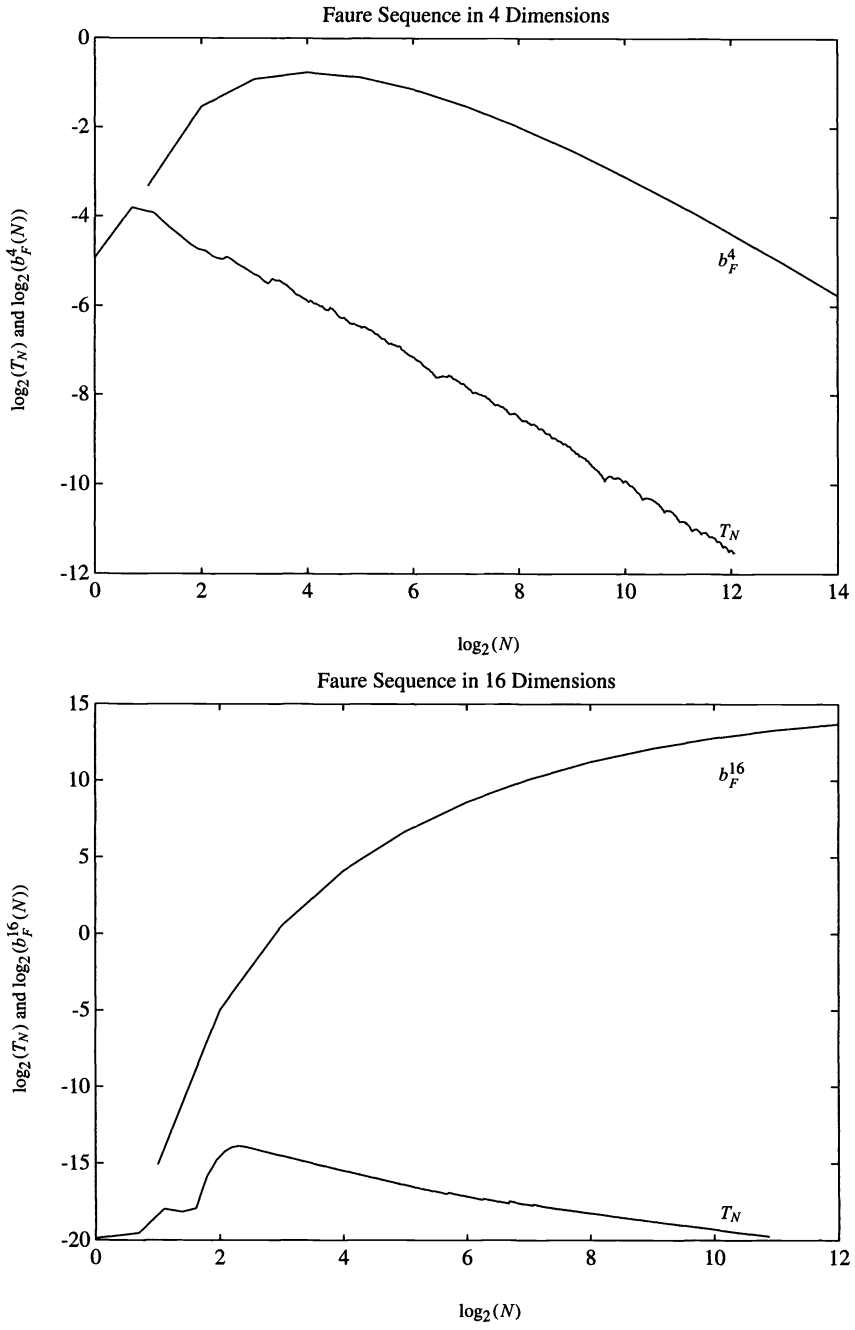


FIG. 2. *L_2 discrepancy T_N and leading term of bound for Faure in* 4 *and* 16 *dimensions.*

6. Calculation of L_2 discrepancy. In his review article on quasi-Monte Carlo methods and sequences [14], Niederreiter discusses only the L_2 star discrepancy T_N^* based on rectangles, which have one corner at the point **0**. An explicit formula for this quantity was first derived by Warnock [25] and subsequently used by Braaten and Weller [1] and Sarkar and Prasad [22].

The result obtained for a given sequence $\{x_n\}$ of N terms is

$$(T_N^*)^2 = \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\prod_{i=1}^{s}\left(1-\max(x_{n,i},x_{m,i})\right) - \frac{2^{-s+1}}{N}\sum_{n=1}^{N}\prod_{i=1}^{s}(1-x_{n,i}^2) + 3^{-s}$$

If the sequence is random, such that each coordinate of each term is an independent random number, then by integrating over the space I^{sN}, the expected value of $(T_N^*)^2$ for a random sequence can be found to be

$$E\left((T_N^*)^2\right) = \frac{2^{-s}-3^{-s}}{N}.$$

While useful in theoretical discussions due to its relationship with D_N^*, T_N^* suffers as a means of comparing sequences and predicting performance because of the strong emphasis it puts on points near $\mathbf{0}$. If $\boldsymbol{x}_j = (0,\ldots,0)$ is a point of the N term sequence, then the dominant term in the calculation of $(T_N^*)^2$ comes from the double sum when $n = m = j$. This term contributes $1/N^2$ to the sum, which tends to dominate all other terms in the sum. Thus $T_N^* \approx \frac{1}{N}$. A similar result is obtained if the sequence contains a point with all coordinates near zero. If this point is excluded from the sequence, however, there is no longer a dominant term, and T_N^* appears rather different. This can be seen by comparing the plots in Fig. 3. Of course this is a transient effect with diminishing influence as N increases. However, as dimension increases, so does the length of the transient region.

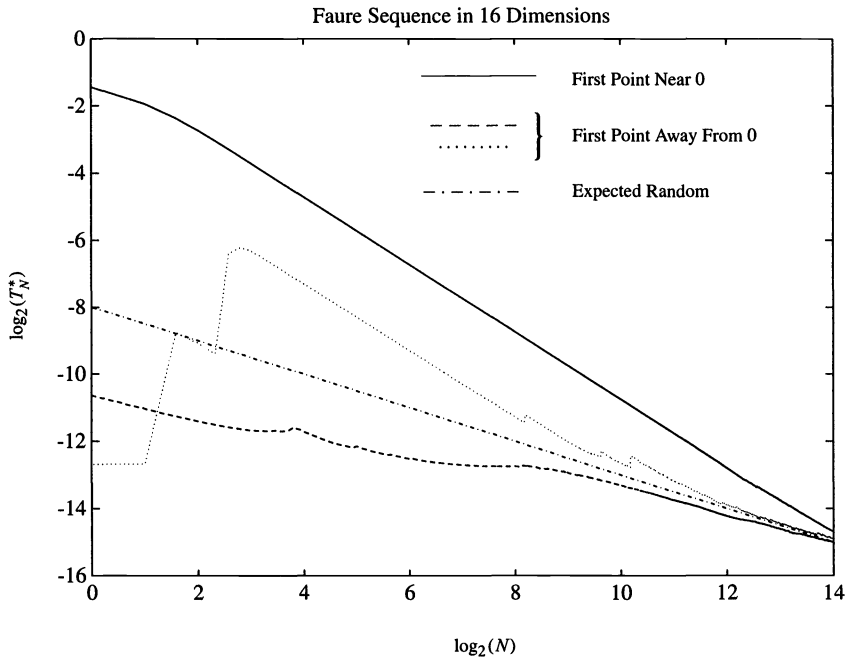


FIG. 3. *L_2^* discrepancy of Faure with and without point near zero.*

As an alternative to T_N^*, the modified L_2 discrepancy T_N was defined in §2. As with the L_2 star discrepancy, it is possible to derive an exact formula for T_N for any given sequence $\{\boldsymbol{a}_n\}$ of N terms (the notation for a sequence is changed here from $\boldsymbol{x}$ to $\boldsymbol{a}$ to help distinguish

between the terms of the sequence and the points defining the rectangles). Using the Heaviside function

$$\theta(y) = \begin{cases} 1, & y > 0, \\ 0, & y \leq 0, \end{cases}$$

it is possible to rewrite R_N as

$$R_N(J(\mathbf{y}, \mathbf{z})) = \frac{1}{N} \sum_{n=1}^{N} \prod_{i=1}^{s} \theta(z_i - a_{n,i}) \cdot \theta(a_{n,i} - y_i) - \prod_{i=1}^{s} (z_i - y_i) .$$

Squaring this quantity and integrating over the domain described above leads to T_N^2, which can be expressed as $T_N^2 = A + B + C$, where

$$A = \frac{1}{N^2} \sum_{n=1}^{N} \sum_{m=1}^{N} \prod_{i=1}^{s} \int_{(y_i, z_i) \in I^2, y_i < z_i} \theta(z_i - a_{n,i}) \theta(a_{n,i} - y_i) \theta(z_i - a_{m,i}) \theta(a_{m,i} - y_i) dy_i dz_i,$$

$$B = \frac{-2}{N} \sum_{n=1}^{N} \prod_{i=1}^{s} \int_{(y_i, z_i) \in I^2, y_i < z_i} (z_i - y_i) \theta(z_i - a_{n,i}) \theta(a_{n,i} - y_i) dy_i dz_i,$$

$$C = \prod_{i=1}^{s} \int_{(y_i, z_i) \in I^2, y_i < z_i} (z_i - y_i)^2 dy_i dz_i .$$

These quantities can be evaluated as follows.

For A.

$$\begin{aligned} &\int_{(y_i, z_i) \in I^2, y_i < z_i} \theta(z_i - a_{n,i}) \theta(a_{n,i} - y_i) \theta(z_i - a_{m,i}) \theta(a_{m,i} - y_i) dy_i dz_i \\ &\quad = \int_0^1 \left[\int_{y_i}^1 \theta(z_i - a_{n,i}) \theta(z_i - a_{m,i}) dz_i \right] \theta(a_{n,i} - y_i) \theta(a_{m,i} - y_i) dy_i \\ &\quad = \int_0^1 [1 - \max(y_i, a_{n,i}, a_{m,i})] \theta(a_{n,i} - y_i) \theta(a_{m,i} - y_i) dy_i \\ &\quad = [1 - \max(a_{n,i}, a_{m,i})] \cdot \min(a_{n,i}, a_{m,i}). \end{aligned}$$

Thus

$$A = \frac{1}{N^2} \sum_{n=1}^{N} \sum_{m=1}^{N} \prod_{i=1}^{s} [1 - \max(a_{n,i}, a_{m,i})] \cdot \min(a_{n,i}, a_{m,i}).$$

For B.

$$\begin{aligned} &\int_{(y_i, z_i) \in I^2, y_i < z_i} (z_i - y_i) \theta(z_i - a_{n,i}) \theta(a_{n,i} - y_i) dy_i dz_i \\ &\quad = \int_0^1 \left[\int_{y_i}^1 (z_i - y_i) \theta(z_i - a_{n,i}) dz_i \right] \theta(a_{n,i} - y_i) dy_i \\ &\quad = \int_0^{a_{n,i}} \left[\int_{y_i}^1 (z_i - y_i) \theta(z_i - a_{n,i}) dz_i \right] dy_i. \end{aligned}$$

The inner integral is the area of a trapezoid with corners $(a_{n,i}, 0)$, $(1, 0)$, $(1, 1 - y_i)$, and $(a_{n,i}, a_{n,i} - y_i)$. Thus we have

$$\int_{y_i}^{1} (z_i - y_i)\theta(z_i - a_{n,i})dz_i = \frac{1}{2}(1 - a_{n,i})(1 + a_{n,i} - 2y_i) .$$

Substituting this in the previous equation, it follows that

$$\begin{aligned}\int_0^{a_{n,i}} \frac{1}{2}(1 - a_{n,i})(1 + a_{n,i} - 2y_i)dy_i &= \frac{1}{2}(1 - a_{n,i})\left[(1 + a_{n,i})y_i - y_i^2\right]_0^{a_{n,i}} \\ &= \frac{1}{2}a_{n,i}(1 - a_{n,i}).\end{aligned}$$

Thus

$$B = -\frac{2^{-s+1}}{N}\sum_{n=1}^{N}\prod_{i=1}^{s} a_{n,i}(1 - a_{n,i}) .$$

For C.

$$\begin{aligned}\int_{(y_i,z_i)\in I^2, y_i<z_i} (z_i - y_i)^2 dz_i dy_i &= \int_0^1 \left[\frac{1}{3}(z_i - y_i)^3\right]_{y_i}^1 dy_i \\ &= \int_0^1 \frac{1}{3}(1 - y_i)^3 dy_i \\ &= \frac{1}{12}.\end{aligned}$$

Thus

$$C = 12^{-s} .$$

Combining these elements leads to the formula

$$\begin{aligned}(T_N)^2 = {} & \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\prod_{i=1}^{s}[1 - \max(a_{n,i}, a_{m,i})] \cdot \min(a_{n,i}, a_{m,i}) \\ & -\frac{2^{-s+1}}{N}\sum_{n=1}^{N}\prod_{i=1}^{s} a_{n,i}(1 - a_{n,i}) + 12^{-s} .\end{aligned}$$

As with the star discrepancy, it is possible to compute the expected value of this quantity for a random sequence. This root mean square (rms) expectation of T_N is given by

$$\begin{aligned}E(T_N^2) &= \int_{I^{sN}} T_N^2 \prod_{n=1}^{N}\prod_{i=1}^{s} da_{n,i} \\ &= \frac{1}{N}6^{-s}(1 - 2^{-s}).\end{aligned}$$

Thus again the average L_2 discrepancy of a random sequence decreases like $N^{-1/2}$, corresponding nicely with the random Monte Carlo bound.

By comparing the formulas for T_N^* and T_N, it appears likely that $T_N^* > T_N$ for all sequences and all N, although this has not been proved. It is certainly true for the expected value of a random sequence, and it has been borne out in all computations. Figure 4 compares the two discrepancies for a couple of versions of the Faure sequence (created by starting at different places in the sequence). The qualitative behavior of the two discrepancies is similar for large N, but T_N is smoother and has a shorter, less extreme, transient region. This becomes even more important in higher dimensions, where the transient region is considerably longer.

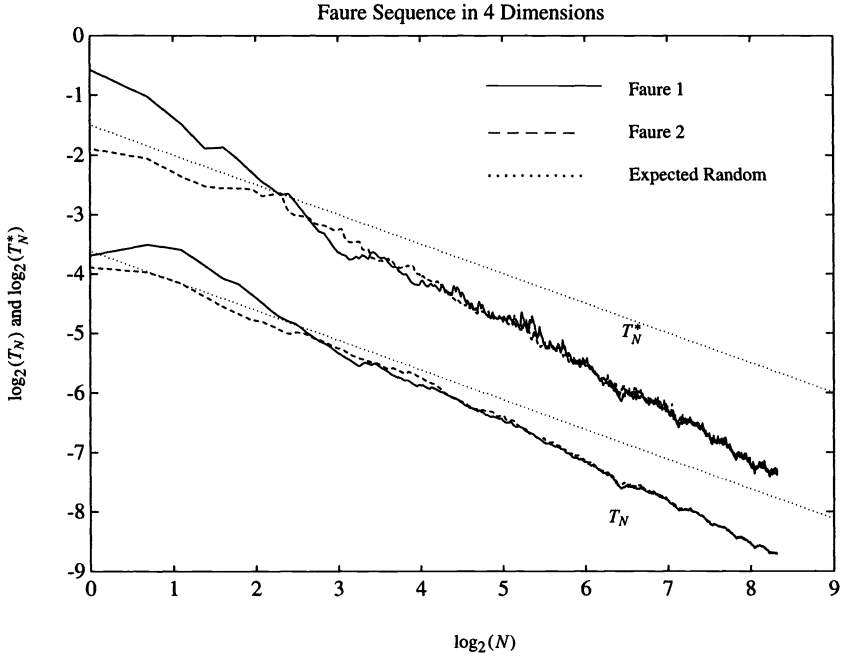


FIG. 4. *Comparison of T_N and T_N^*.*

A disadvantage of using T_N to measure discrepancy is that no direct connection has been established between it and integration error. Whereas the relationship $D_N^* < D_N$ allows the Koksma–Hlawka inequality to be modified to include D_N, the sup discrepancy taken over all rectangles, it is not possible to change the L_2 version from T_N^* to T_N. Nevertheless, actual computation of T_N indicates that it is a useful gauge of integration error convergence as a function of N. Figures 5–8 show plots of T_N (solid line) on a log log base 2 scale. The rms expectation of T_N for a random sequence (dashed line) is also plotted, along with the function $\frac{1}{N}$ (dotted line) for reference.

There are several interesting features of these plots, which should be noted. Sobol' [23] predicts that when N equals large enough powers of two, the value of discrepancy should have a local minimum. The plot for the Sobol' sequence in three dimensions shows this kind of behavior. After $N = 2^{10}$ there appears to be a cusp at the powers of 2. Closer examination of this phenomenon shows that a minimum actually occurs at a few points short of the power of two. Sobol' also predicts how large N must be before this occurs, with the cut-off value increasing with dimension. For three and four dimensions, he shows greater uniformity for $N = 2^6$ and above. However, the plots do not reveal any particularly noteworthy behavior until, as mentioned above, $N = 2^{10}$ for three dimensions and $N = 2^{13}$ for four dimensions. For $s = 8$ Sobol's formula for the cut off value predicts improved discrepancy for powers of

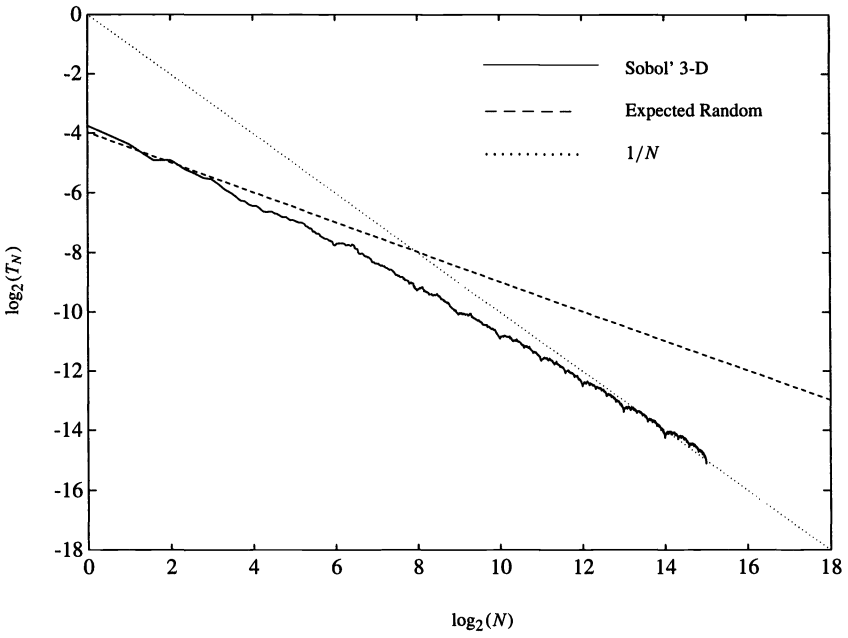


FIG. 5. *T_N for three-dimensional Sobol' sequence.*

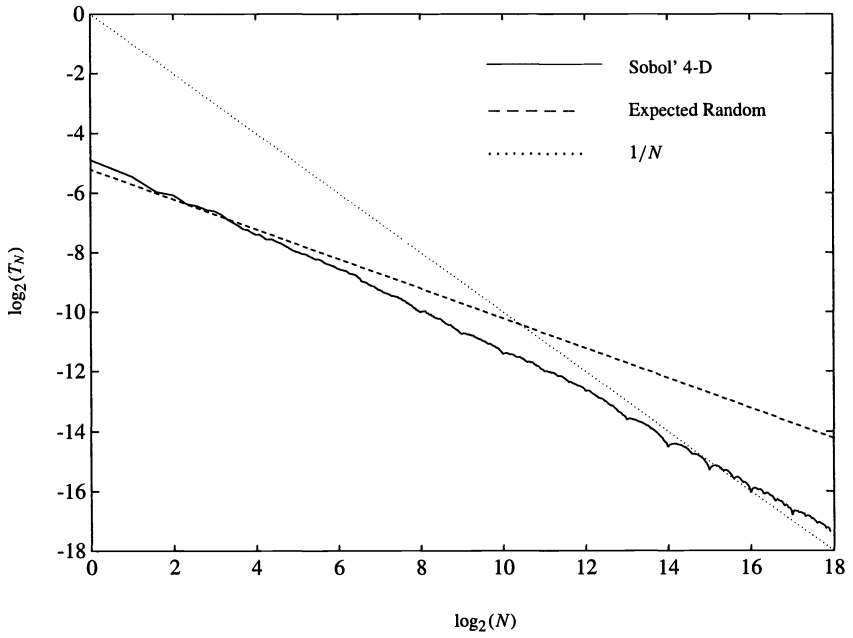


FIG. 6. *T_N for four-dimensional Sobol' sequence.*

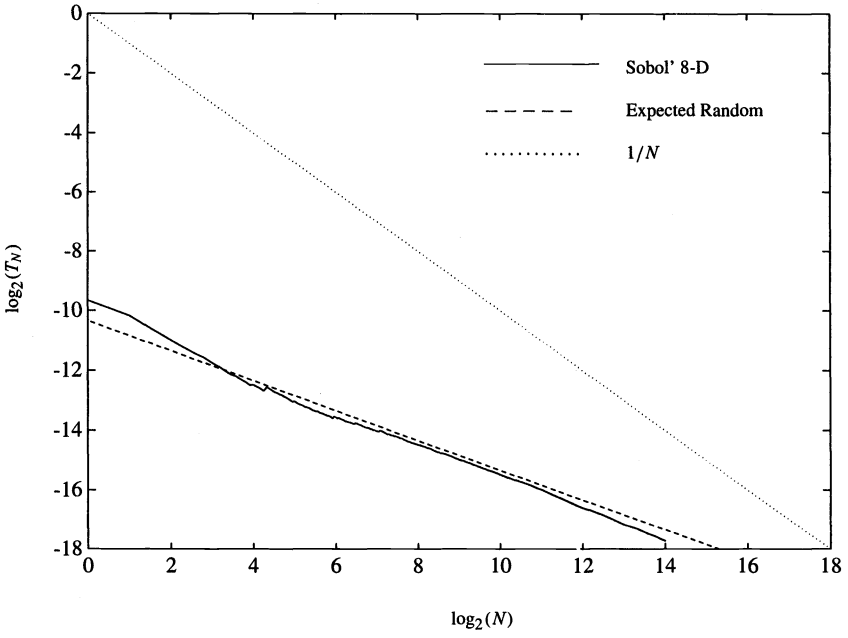


FIG. 7. *T_N for eight-dimensional Sobol' sequence.*

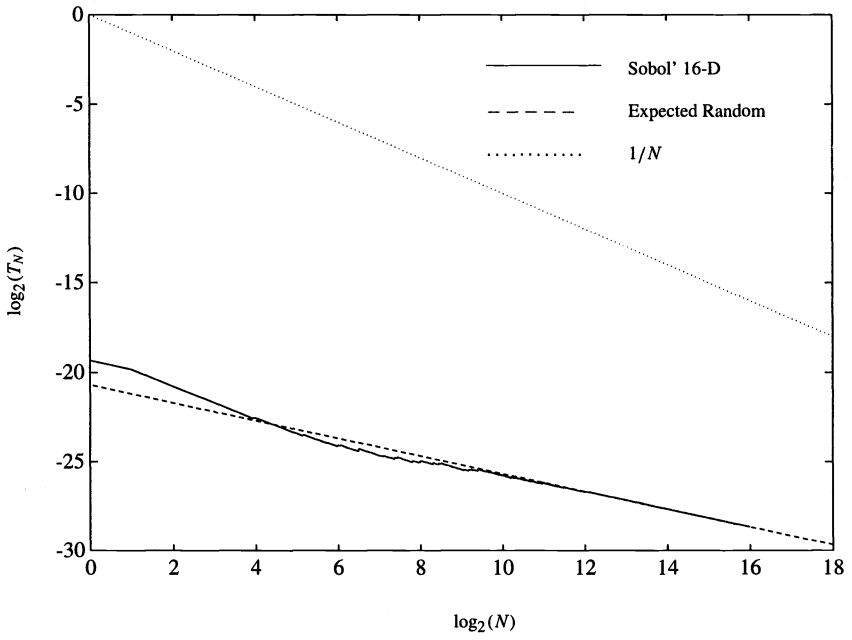


FIG. 8. *T_N for sixteen-dimensional Sobol' sequence.*

two greater than or equal to $N = 2^{22}$. Since this number is larger than four million, it is not surprising that nothing special is seen on the plot for eight dimensions, which only goes out to about 16,000.

A more important observation to be made from these plots is the transition of T_N from random-like behavior for low values of N, to perhaps eventual $\frac{1}{N}$-type convergence. For dimensions three and four, T_N starts near the rms expectation curve of T_N for a random sequence, but fairly quickly starts to decay at a faster rate than $N^{-1/2}$. The transition seems to occur around the point where the rms expectation curve and the $\frac{1}{N}$ curve intersect. This is easily shown to occur at around $N = 6^s$. This is a purely heuristic estimate, since the curve $\frac{1}{N}$ is used only as an approximation to the asymptotic behavior of T_N and does not mean much for smaller values of N. However, it does provide a rough estimate of the nature of T_N. In eight dimensions this predicts a transition at around $N = 2^{20}$. Figure 7 shows that T_N is just beginning to break away from the rms expected curve around $N = 2^{14}$. In 16 dimensions, after an initial transient region which is near the rms expectation of T_N for a random sequence, the value of T_N for the Sobol' sequence lies almost exactly on the rms expectation curve out to $N = 2^{16}$ and probably considerably farther. For this dimension the heuristic estimate predicts the transition at 2^{41}. It might be hoped that this is an overestimate; however, this kind of exponential growth of the transition point is similar to that of the maximum point for the theoretical bound on discrepancy. Figure 9 compares the L_2 discrepancies of various sequences in 16 dimensions. Except for within the initial transient, all of the sequences behave almost identically; that is, as if they were random. This indicates that in high dimensions, unless one uses a very large number of points, quasi-random sequences are no more uniform than random sequences.

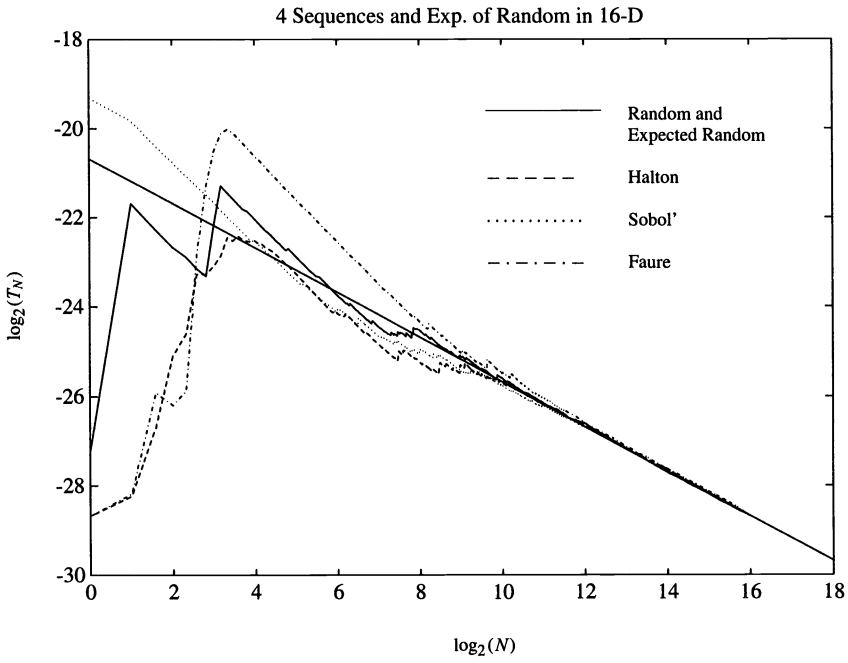


FIG. 9. *L_2 discrepancy in 16 dimensions for various sequences.*

It should also be noted that the value of T_N is insensitive to where the sequence begins. All the sequences considered are produced by mapping the sequence of integers $\{n = 1, 2, 3, \ldots\}$

to points in I^s; however, it is not mandatory to start with $n = 1$. Any number of the initial terms can be discarded without affecting T_N much, except in the transient region. The same is true for T_N^*, but the change in the transient region may be much more extreme if a point very close to zero is included. It is also possible that, when these initial terms are discarded, Sobol's improved bounds on discrepancy are no longer valid. For Sobol' these occur at powers of 2; for Faure they should occur at powers of $p(s)$. However, as pointed out above, this improvement is only of practical value for low dimensions; moreover, from the discrepancy plots, it is clear that the value of discrepancy at the special values of N is not all that much lower than otherwise. Orthogonal projections (discussed next), which are calculations of discrepancy and computations of integrals, show that there is not much difference between any two subsequences of equal length. Thus it does not matter what value of n corresponds to the beginning of the sequence.

For the T_N^* discrepancy, similar dependence on dimension s is observed. To optimize the sequence for a given dimension s, we consider "Hammersley"-type sequences in which the components in the first dimension are lattice points $\frac{n}{N}$, while the other components come from an $(s-1)$-dimensional quasi-random (or random) sequence. Figures 10 and 11 show the T_N^* discrepancy for a variety of sequences, a pure random sequence, a Hammersley random sequence, a Hammersley–Halton sequence, and a normal Halton sequence, in dimensions 2, 10, and 15. The random sequences have discrepancy of size $N^{-1/2}$ in all dimensions. In dimension 2 the quasi-random sequences have discrepancy of size N^{-1}, but in dimension 10 it is of size $N^{-1/2}$ for small N before beginning to drop off faster for larger N. In dimension 15 the quasi-random sequences have discrepancy almost exactly that of a random sequence (i.e., of size $N^{-1/2}$) for the values of N computed here, although it must eventually approach size N^{-1} for extremely large N.

7. Orthogonal projections. Another approach to understanding quasi-random sequences is to look at two-dimensional orthogonal projections of the points in I^s. The assumption made here is that if a sequence is uniformly distributed in I^s, then the two-dimensional sequences formed by pairing coordinates (i.e., the two-dimensional orthogonal projections) should also be uniformly distributed. Moreover the discrepancy of projections of a sequence occur explicitly in the average error identity (6). The appearance of nonuniformity in these projections is an indication of potential problems in using a quasi-random sequence for integration. However, a sequence with very nonuniform behavior in some projection may in fact be reasonably uniform in I^s. Of course, attempting to integrate a function that has strong dimensional dependence on just the two dimensions in question will lead to poor results, but for many functions a bad pairing of dimensions may not have much influence.

Here a catalog of potentially bad behavior is given for all the sequences under consideration, along with some insight into the source of these problems. First it is worthwhile to consider a pseudo-random sequence. Figure 12 shows the projection of 4096 points on the first and 16th dimensions of the sequence generated in Matlab (using seed zero). The points appear to be randomly distributed and fairly uniform. Any decent pseudo-random number generator should be able to produce this effect for orthogonal projections. Nothing particularly different was seen from examining other projections of this sequence.

Figure 13 shows the projection of 4096 points of the Halton sequence onto the first and second dimensions and the 28th and 29th dimensions. Compared with the random sequence, this low-dimensional projection appears to be considerably more uniform, and thus a better sequence. However, difficulties with high dimensions occur, as observed in [1]. If approximately 5900 points are used, then the projection onto the 28th and 29th dimensions would be almost perfectly uniform. However, this would not be true for any other dimensional pairings, and as more points are added the uniformity would disappear. The problem here arises from

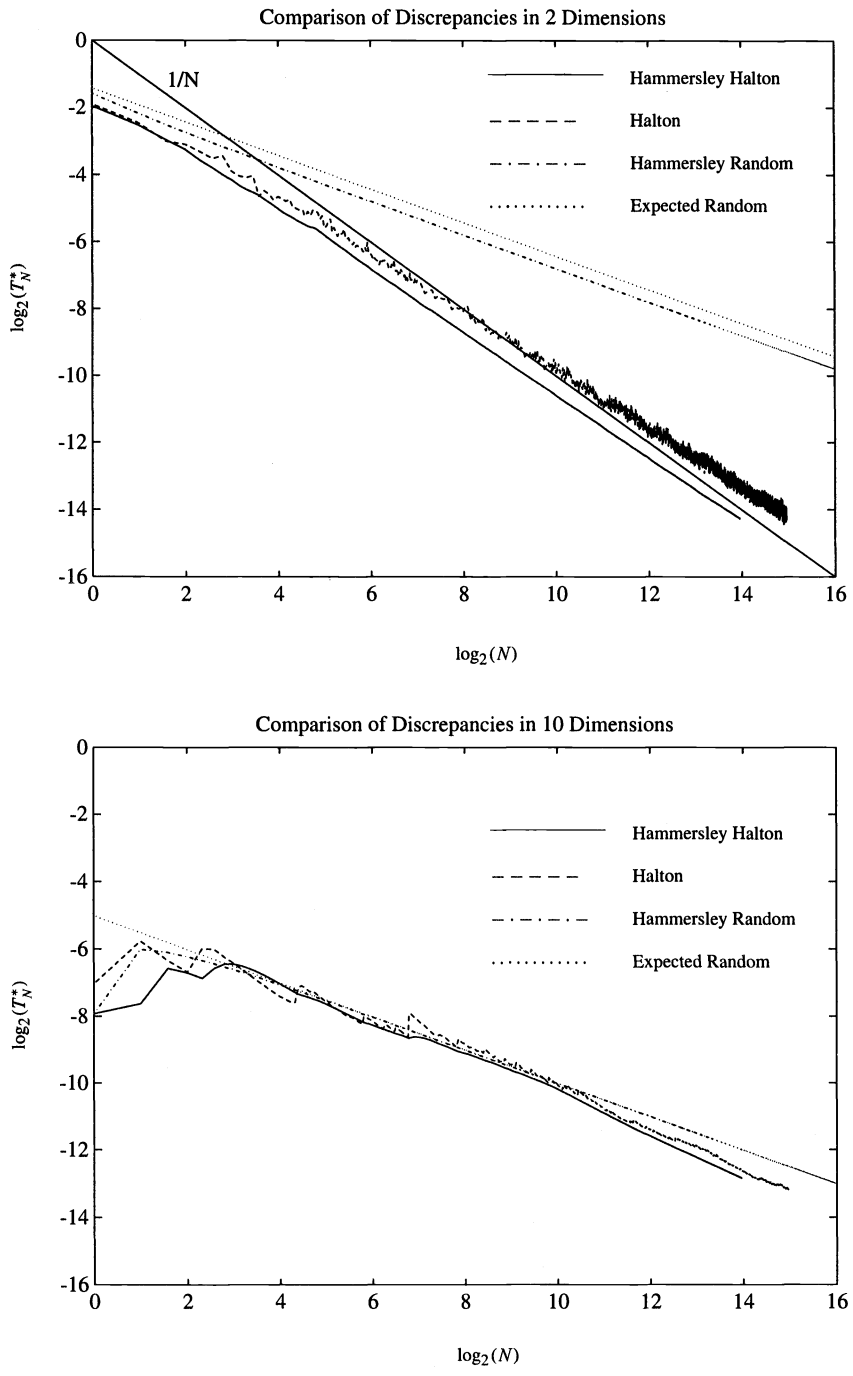


FIG. 10. L_2^* *discrepancy in* 2 *and* 10 *dimensions for various sequences.*

the use of large primes, in this case 107 and 109 for the 28th and 29th dimensions respectively. The 28th dimension of the Halton sequence consists of monotone increasing subsequences of length 107 terms. When this is paired with the monotone subsequences of length 109 for the 29th dimension, the lines seen in the plot occur.

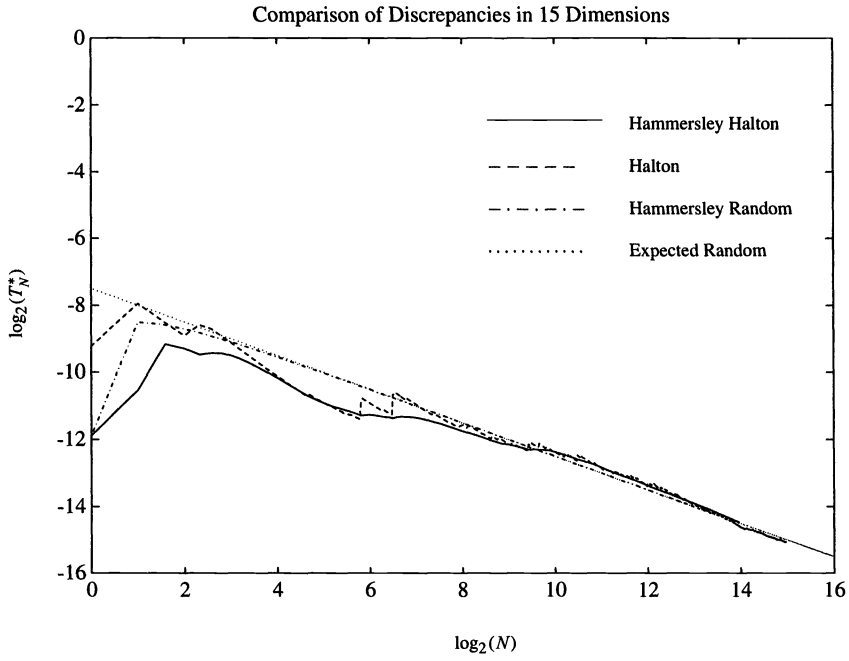


FIG. 11. *L_2^* discrepancy in 15 dimensions for various sequences.*

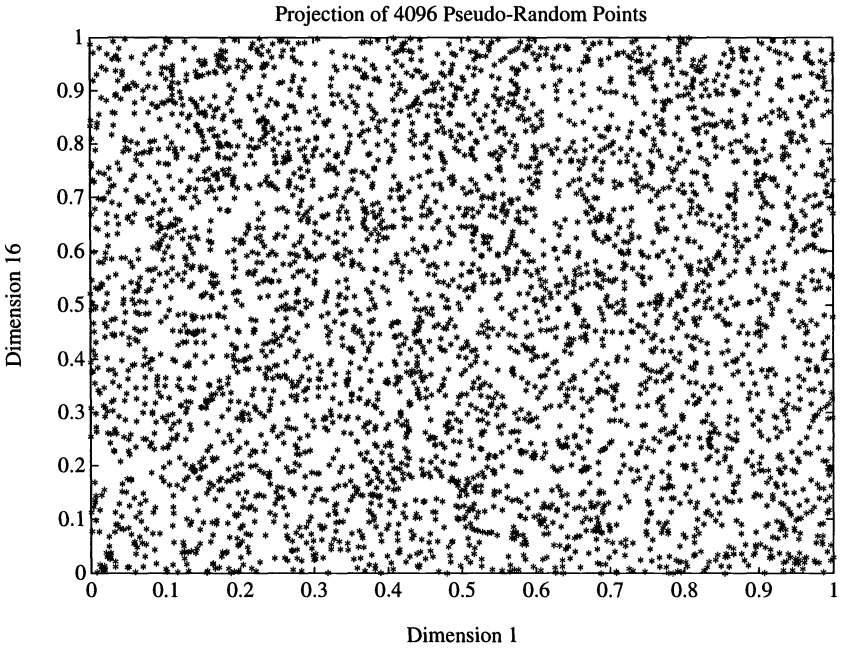


FIG. 12. *Two-dimensional projection of random sequence.*

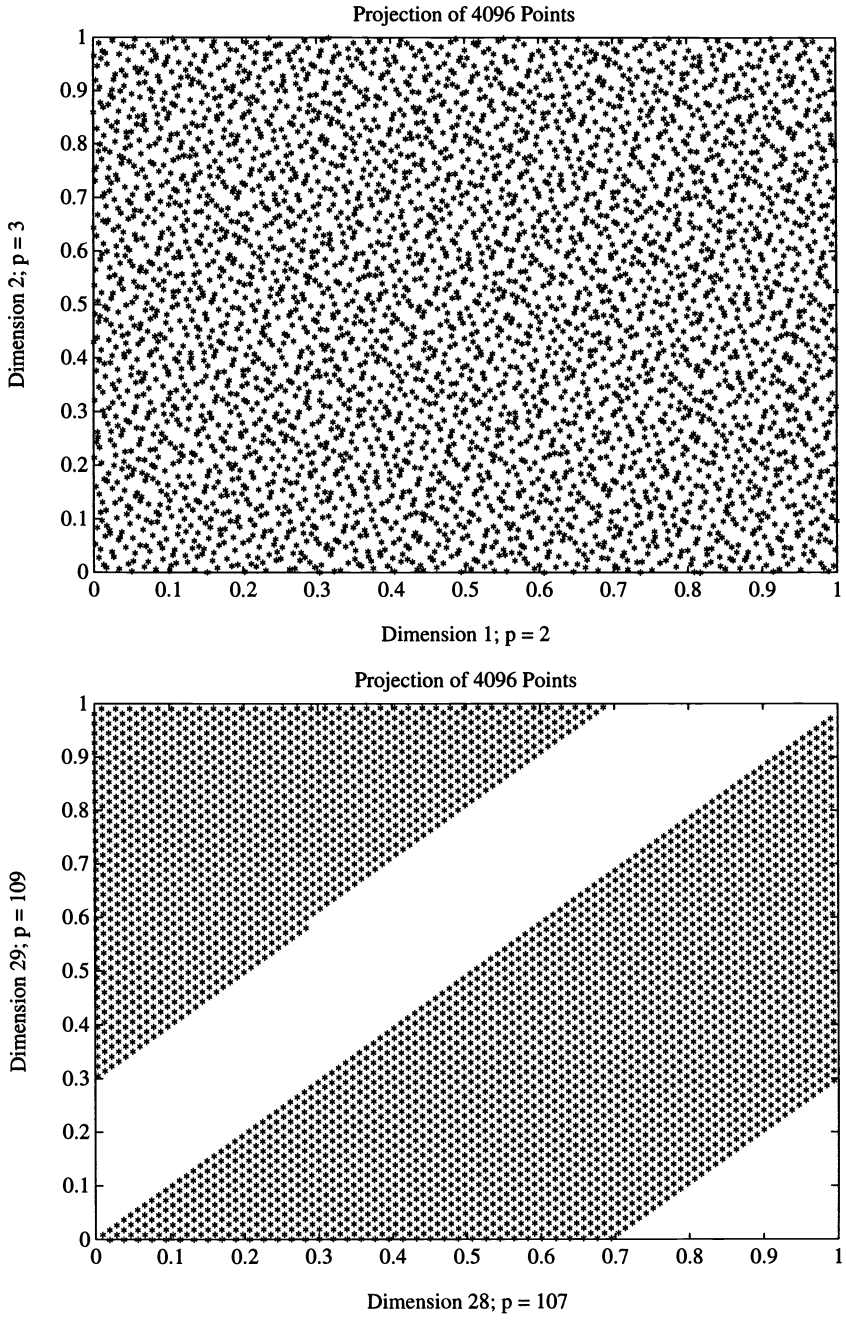


FIG. 13. *Two-dimensional projection of Halton sequence.*

To improve this situation Braaten and Weller [1] suggest a scrambling or permutation procedure, which preserves the traditional $(\log N)^s/N$-type bound for the discrepancy. A less elaborate, but easier to implement, scrambling technique was used in the current work. Here the sequence was simply (pseudo)randomly scrambled independently in each dimension.

For example, if N points in I^s were required, then s sequences of N random numbers were generated and sorted from smallest to largest. This mapping of original position in the sequence to final position was then used to permute the Halton sequence. Figure 14 shows a two-dimensional projection of the 29-dimensional randomly scrambled sequence. At least in terms of projections, scrambling seems to greatly improve the Halton sequence in high dimensions. In each dimension, this procedure does not change the one-dimensional discrepancy of the N points. As the pairings across dimensions are (pseudo-)random, this may lead to slower, more random-like, convergence, although the actual value of discrepancy for a given N will hopefully be smaller over a reasonable range of N.

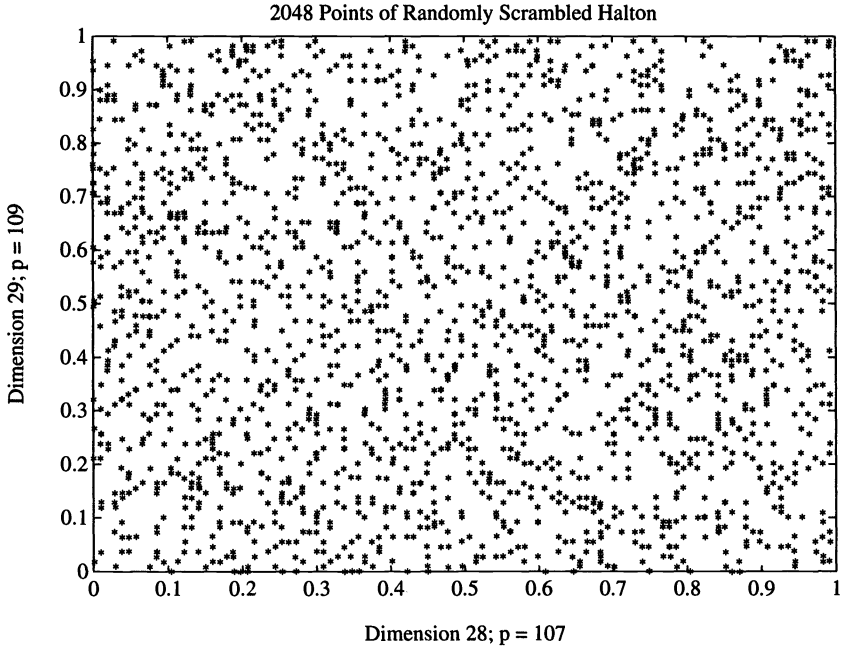


FIG. 14. *28th versus 29th dimensions of scrambled Halton sequence.*

To compare the standard Halton sequence to the scrambled version, Braaten and Weller [1] compute the discrepancy T_N^* of the first 1000 points of each sequence in 8, 12, and 16 dimensions. This result is somewhat misleading, though, because of the use of the T_N^*. This measure of uniformity weights the point zero and points near zero, such as the first terms of the Halton sequence in high dimensions, much greater than other points in the unit cube. The calculated value of T_N^* for the Halton sequence is almost entirely determined by the first point. If the sequences formed by deleting the first ten points of Halton and scrambled Halton are compared, the values of T_N^* are almost identical for this range of number of points and dimension. This is not to say that scrambling does not improve the sequence, but just that the improvement cannot be seen through calculation of T_N^*.

Press and Teukolsky [21] give some examples of the projections of the Sobol' sequence to illustrate how it fills out the unit square. They show how the first 256 points lay down a fairly uniform, but distinct, pattern in the square, and how the next 256 points fill in the gaps left by the first group. The points are put down to be uniform, and additional points "know" about the spaces left by the original points, so they are put down to make the whole sequence even more uniform.

To understand what can potentially go wrong here, it is necessary to have a feel for how the Sobol' sequence is generated. Each dimension of this sequence is just a permutation of the Halton sequence with prime base 2 (this is also known as the van der Corput sequence) whenever $N = 2^m$ for $m = 0, 1, 2, 3, \ldots$. These permutations are generated from irreducible polynomials over the field {0,1}. Ideally, polynomials of the lowest degree possible are used; however, as dimension increases, it is necessary to use polynomials of higher and higher degree. To generate a one-dimensional sequence from a polynomial of degree d, $d-1$ odd integers $j_1, \ldots, j_{d-1}$ must be chosen with the restriction that $j_i < 2^i$. Thus there are $2^d - 1$ possible ways of picking the starting values. Sobol' has given a list of good starting values for dimension up to 16 [24]. These are said to be better because they produce sequences that satisfy an additional uniformity property.

What can go wrong with the Sobol' sequence involves the pairing of dimensions. The fact that each dimension is a permutation of the same sequence allows for certain correlations to develop. In some cases this is good, because it allows for the phenomenon described above where points fill in the gaps left by previous points. However, these correlations can also produce regions in the unit square where no points fall until N becomes extremely large. Figure 15 shows a "good" pairing of dimensions using Sobol's second and third dimensions with his recommended starting values. A "bad" pairing of dimensions is also shown, representing what would be the 27th and 28th dimensions following Sobol's convention for associating dimension with generating polynomial. The polynomials used here are $x^7 + x^5 + x^4 + x^2 + x + 1$ and $x^7 + x^5 + x^4 + x^3 + x^2 + x + 1$ and the starting values are (1,3,5,11,3,3,35) for the 27th dimension and (1,1,7,5,11,59,113) for the 28th dimension. If one or two of these starting values were changed, then the problem illustrated in the graph would disappear. However, it does not seem possible to tell a priori that this is a bad pairing. Moreover, neither set of starting values is particularly at fault, because when they are paired with other dimensions there is no such pathological behavior.

For 29 dimensions, there are 406 pairings of dimensions that could be checked for such correlations; this probably should be done if the Sobol' sequence in this high a dimension is to be used. Sobol' may have checked this for the recommended first 16 dimensions; however, the uniformity property that these sequences satisfy does not exempt them from such bad behavior. There may also be higher-dimensional correlations, which would be difficult to detect.

The bad behavior seen in the second plot of Figure 15 can be explained in terms of the filling-in-holes idea. If 8192 (2^{13}) points are used, the plot looks almost identical to what is shown for 4096. However, the next 8192 points fall only where the gaps appear. Thus by $N = 16,384$, the projection plot is almost perfectly uniform. The problem is that the cycle for filling in holes is 2^{13}, which is too long.

The idea behind the Faure sequence is an extension of the theory of the Sobol' sequence. This theory, which has been somewhat extended by Niederreiter [15], is based on the idea of the elementary rectangle base p in I^s This is a rectangle that is a product of s intervals of the form $[ap^{-d}, (a+1)p^{-d})$, where a is an integer less than p^d and d is a nonnegative integer. For arbitrary integer m, the goal is to construct a sequence such that every subsequence of length p^m of the form $(k-1)p^m < n \leq kp^m$ (n is the index of the sequence) has the property that each elementary rectangle base p of volume p^{-m} contains exactly one point of the subsequence. Faure constructs such a sequence by taking p to be the smallest prime greater than or equal to s. Figure 16 shows the projections onto the first and second dimensions of 3125 points of the five-dimensional sequence ($p = 5$), and 2197 points of the 13-dimensional sequence ($p = 13$). The second plot of this figure shows some considerable difficulties, which may initially seem surprising given that the sequence was constructed so that every elementary rectangle base 13 with volume $\frac{1}{N}$ contains exactly one point if N is a power of 13. However, Figure 17 illustrates

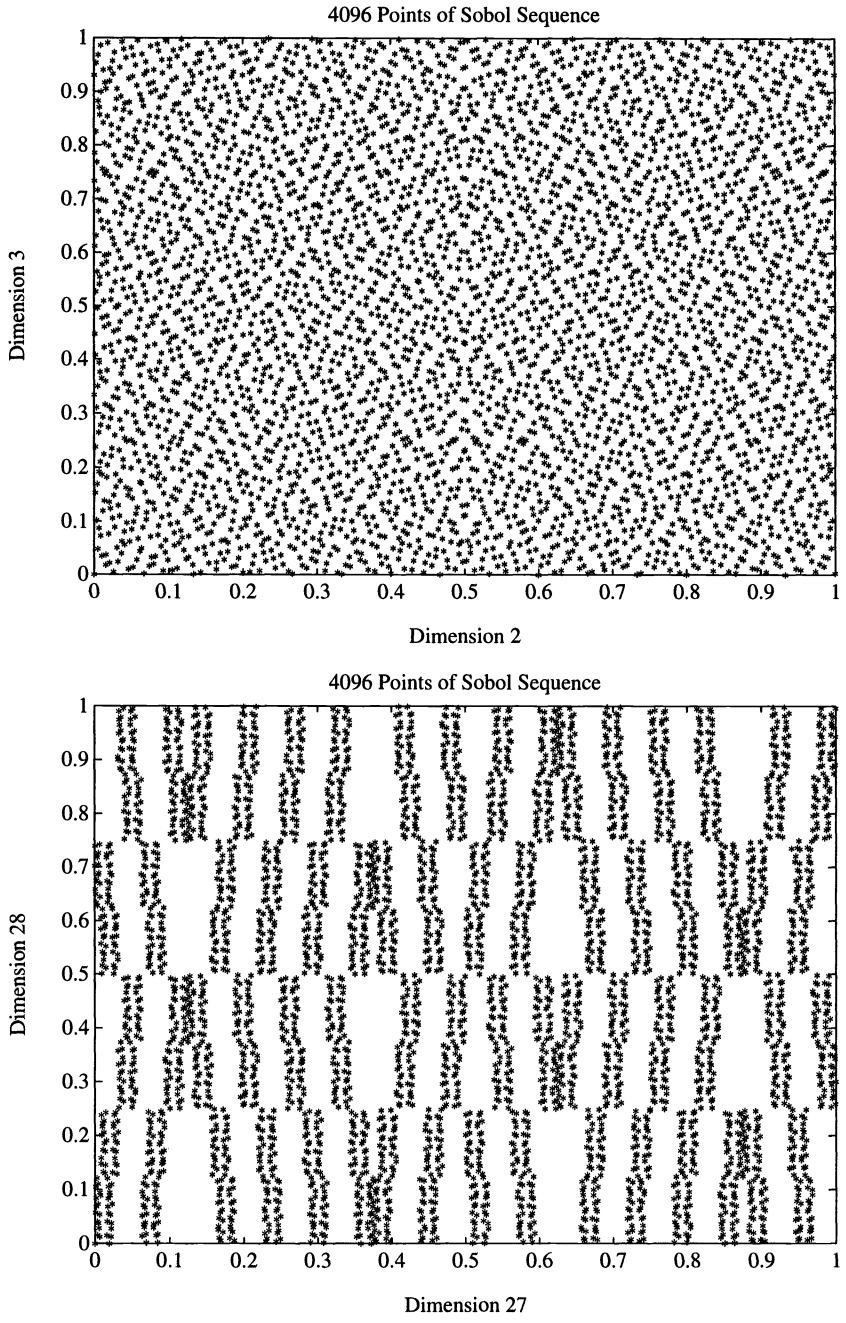


FIG. 15. *Two-dimensional projection of Sobol' sequence.*

how this can happen. The grid shown divides the unit square into elementary rectangles of volume 13^{-3}. It is clear that each rectangle does have exactly one point of the sequence in it; unfortunately, the distribution of the point inside the rectangle is not uniform. It will take approximately $13^4 = 28,561$ points before the square is more satisfactorily filled.

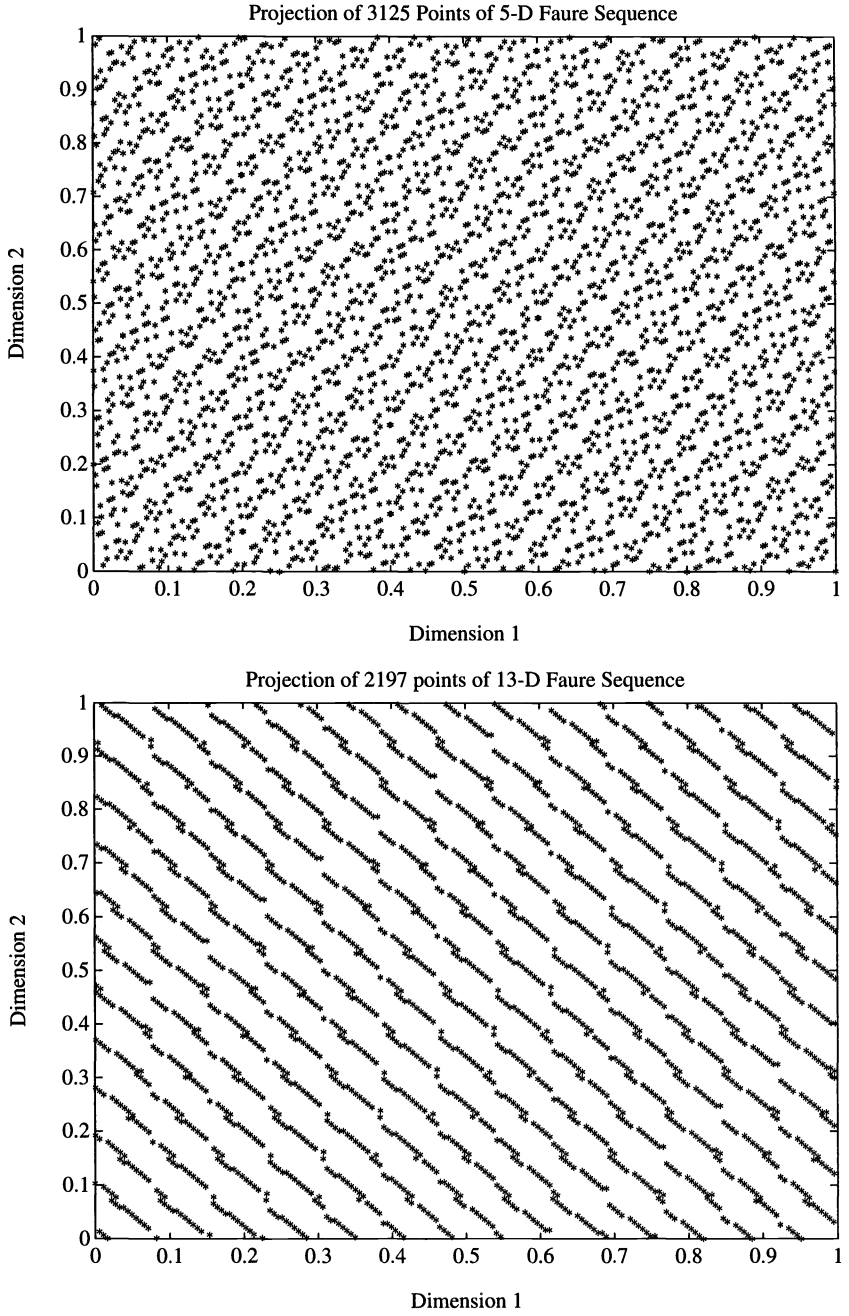


FIG. 16. *Two-dimensional projection of Faure sequence.*

As noted above, even if a sequence has poor two-dimensional projections, it may still be fairly uniform in I^s, and there are many functions which it may integrate quite well. However, it is important to be aware of the potential problems these sequences may have, and the orthogonal projections are a good means of identifying and assessing the difficulties.

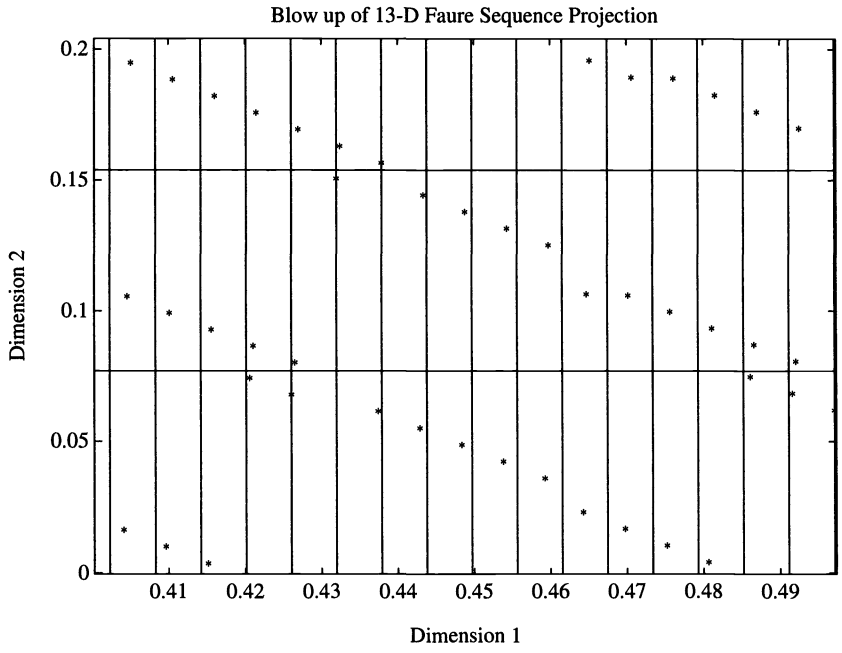


FIG. 17. *Two-dimensional projection of Faure sequence.*

8. Computational speed. Another aspect of quasi-random sequences worth considering is the computer time required to generate them. Fox [5] and Bratley and Fox [2] present such results for various values of N and dimensions and for a pseudo-random number generator along with the three quasi-random sequences under consideration here. Their calculations were done on a Cyber 855 computer and include calls to the initialization routine and a routine to evaluate a simple integral, as well as to the sequence generator. They conclude that the time spent on the initialization routine is negligible. In comparing the sequences, they find that Sobol' is 1 to 3 times faster than their random number generator and 3 to 5 times faster than Halton. They find Halton to be approximately 4 times faster than Faure. They also state that when run on a different computer, the ordering remained the same, but the ratios for computing times for the various sequences were much different. The results show that computation time is approximately proportional to dimension and to number of points used.

Similar timing experiments were run for this work on an Alliant FX/80. The pseudo-random number generator used was the routine lib_vdran supplied by Alliant and found in the common library. The results of these experiments are given in Table 1. These results are not definitive, since we have not made much effort to optimize our code. Nevertheless we expect that they will be of interest to the potential user. Here, the only thing timed was the sequence generating subroutine; the initialization routine was not included, nor was any integral evaluated. Here, the random number generator is the fastest, and its times are proportional to the number of points and the dimension. Sobol' again is faster than Halton, but only slightly so, around 1.2 times faster. Both Sobol' and Halton have timings that are proportional to N, but that grow slower than linearly with dimension. This is probably related to the vector and parallel aspects of the Alliant. Sobol' and Halton are approximately 4 to 5 times faster than Faure.

TABLE 1
Timings for QMC *sequences (in seconds).*

Generator	N	$s = 5$	$s = 10$	$s = 20$	$s = 40$
Random	1000	0.00363	0.00702	0.0138	0.0284
Halton	1000	0.0263	0.0335	0.0424	0.0568
Sobol'	1000	0.0225	0.0257	0.0321	0.0467
Faure	1000	0.102	0.0976	0.123	0.189
Random	10,000	0.0353	0.0707	0.142	0.285
Halton	10,000	0.265	0.342	0.434	0.535
Sobol'	10,000	0.231	0.269	0.367	0.517
Faure	10,000	1.14	1.08	1.28	1.99
Random	100,000	0.361	0.720	1.45	2.88
Halton	100,000	2.69	3.47	4.31	5.29
Sobol'	100,000	2.35	2.73	3.52	5.12
Faure	100,000	13.1	11.8	13.7	20.5

As Fox points out, sequence generation time is frequently only a fraction of what is required to evaluate a complicated integrand. Thus for many realistic problems, the question of which generator is fastest is not all that important.

9. Conclusions. The computations described above show strong dependence of the discrepancy on dimension s. While the theoretical bound N^{-1} is observed in any dimension for sufficiently large N, it appears that there is a transition value of N, below which the discrepancy is of size $N^{-1/2}$. For such values of N, random-like behavior of the sequence can be expected. This transition point grows exponentially with dimension.

Comparison between different quasi-random sequences have also been presented. While the discrepancy bound suggests that Faure is a superior sequence, the actual calculation of the discrepancy indicates that all the sequences are about the same. The orthogonal projections show that all of the sequences have potential problems as dimension increases; however, Halton is probably the worst, because all its high-dimensional pairings will be nonuniform for large ranges of N. To a certain extent, Faure has the same problem, but the degree of nonuniformity is not as severe. Sobol' may be able to avoid this problem if the starting values are carefully checked for two-dimensional correlations. Of course, this does not preclude three- (or higher-) dimensional projection problems. Although a direct connection has not been demonstrated, we expect that nonuniformity of projections will lead to poorer performance of Monte Carlo methods for many functions.

Finally, computational timings put Sobol' and Halton on about the same ground, while Faure is considerably slower.

The actual value of these sequences must be judged by their performance in Monte Carlo methods. In the companion papers [12], [13] we present computational experiments with quasi-Monte Carlo methods applied to multidimensional integration and to simulation of the heat equation. Again it is found that the performance of these methods degrades with increasing dimension. Nevertheless, quasi-Monte Carlo methods using quasi-random sequences consistently give significant, but limited, improvement over standard Monte Carlo methods using random or pseudo-random sequences.

REFERENCES

[1] E. BRAATEN AND G. WELLER, *An improved low-discrepancy sequence for multidimensional quasi-Monte Carlo integration*, J. Comput. Phys., 33 (1979), pp. 249–258.

[2] P. BRATLEY AND B. L. FOX, *Implementing Sobol's quasirandom sequence generator*, ACM Trans. Math. Software, 14 (1988), pp. 88–100.
[3] P. BRATLEY, B. L. FOX, AND H. NIEDERREITER, *Implementation and tests of low-discrepancy sequences*, ACM Trans. Modeling and Comp. Sim., 2 (1992), pp. 195–213.
[4] H. FAURE, *Discrépance de suites associées à un système de numération (en dimension s)*, Acta Arith., 41 (1982), pp. 337–351.
[5] B. L. FOX, *Implementation and relative efficiency of quasirandom sequence generators*, ACM Trans. Math. Software, 12 (1986), pp. 362–376.
[6] J. H. HALTON, *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*, Numer. Math., 2 (1960), pp. 84–90.
[7] M. H. KALOS AND P. A. WHITLOCK, *Monte Carlo Methods*, Vol. I, John Wiley and Son, New York, 1986.
[8] L. KUIPERS AND H. NIEDERREITER, *Uniform Distribution of Sequences*, John Wiley and Son, New York, 1974.
[9] Y. L. LEVITAN et al., *Short communications on quasi-random sequences for numerical computations*, U.S.S.R. Comput. Math. and Math. Phys., 28 (1988), pp. 88–92.
[10] H. MCKEAN, *Stochastic Integrals*, Academic Press, New York, 1969.
[11] H. G. MEIJER, *The discrepancy of a g-adic sequence*, Indag. Math. (N.S.), 30 (1968), pp. 54–66.
[12] W. J. MOROKOFF AND R. E. CAFLISCH, *A quasi-Monte Carlo approach to particle simulation of the heat equation*, SIAM J. Numer. Anal., 30 (1993), pp. 1558–1573.
[13] ———, *Quasi-Monte Carlo integration*, J. Comp. Phys., to appear.
[14] H. NIEDERREITER, *Quasi-Monte Carlo methods and pseudo-random numbers*, Bull. Amer. Math. Soc., 84 (1978), pp. 957–1041.
[15] ———, *Quasi-Monte Carlo methods for multidimensional numerical integration*, in Numerical Integration III, International Series of Numerical Math. 85, H. Brass and G. Hämmerlin eds., Birkhäuser Verlag, Basel, 1988.
[16] ———, *Application of diophantine approximations to numerical integration*, in Diophantine Approximation and Its Applications, C. F. Osgood, ed., Academic Press, New York, 1973, pp. 129–199.
[17] ———, *Point sets and sequences with small discrepancy*, Monatsh. Math., 104 (1987), pp. 273–337.
[18] ———, *Random Number Generators and Quasi-Monte Carlo Methods*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.
[19] G. PAGES, *Van der Corput sequences, Kakutani transforms and one-dimensional numerical integration*, 1992, preprint.
[20] G. PAGES AND Y. J. XIAO, *Sequences with low discrepancy and pseudo-random numbers: theoretical results and numerical tests*, 1992, preprint.
[21] W. H. PRESS AND S. A. TEUKOLSKY, *Quasi- (that is, sub-) random numbers*, Comput. Phys., 3 (1989), pp. 76–79.
[22] P. K. SARKAR AND M. A. PRASAD, *A comparative study of pseudo and quasi random sequences for the solution of integral equations*, J. Comput. Phys., 68 (1987), pp. 66–88.
[23] I. M. SOBOL', *The distribution of points in a cube and the approximate evaluation of integrals*, U.S.S.R. Comput. Math. and Math. Phys., 7 (1976), pp. 86–112.
[24] ———, *Uniformly distributed sequences with additional uniformity property*, U.S.S.R. Comput. Math. and Math. Phys., 16, (1976), pp. 1332–1337.
[25] T. T. WARNOCK, *Computational investigations of low discrepancy point sets*, in Applications of Number Theory to Numerical Analysis, S. K. Zaremba, ed., Academic Press, New York, 1972, pp. 319–343.
[26] H. WOŹNIAKOWSKI, *Average case complexity of multivariate integration*, Bull. Amer. Math. Soc., 24 (1991), pp. 185–194.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1280–1293, November 1994

A GRADIENT RANDOM WALK METHOD FOR TWO-DIMENSIONAL REACTION-DIFFUSION EQUATIONS*

ARTHUR SHERMAN† AND MICHAEL MASCAGNI†‡

Abstract. An extension to two space dimensions of the gradient random walk algorithm for reaction-diffusion equations is presented. This family of algorithms is related closely to the random vortex method of computational fluid dynamics. Although the computational cost is high, the method has the desirable features of being grid free and of automatically adapting to the solution by concentrating elements where the gradient is large. In addition, the method can be extended easily to more than two space dimensions. A key feature of the method is discretization in terms of the dependent, rather than independent, variable, giving it features in common with Lagrangian particle methods. The method is derived here and its application to some simple reaction-diffusion wave propagation problems is illustrated.

Key words. Monte Carlo method, reaction-diffusion equation, gradient transport, grid-free, adaptive, N-body problem

AMS subject classifications. 65C05, 35K57, 65M99, 31C20

1. Introduction.

We are interested in numerical methods for solving reaction-diffusion equations:

$$u_t = \Delta u + f(u), \tag{1.1}$$

$$u = u(\mathbf{x}, t), \qquad \mathbf{x} \in \mathbb{R}^d.$$

We will focus mostly on the initial-value problem,

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}),$$

in two space dimensions. Our goal is a method that will work well on problems that are difficult for finite-difference methods, such as cases where the solution has sharp gradients. For this reason, we consider a particle method in which computational elements representing the gradient of the solution move by diffusion and are modeled by random walks. The method is grid free and automatically adapts to the solution. We derive the method as the natural extension of a one-dimensional (1-D) Monte Carlo method [27]. Here we report preliminary computational studies on some simple model problems in order to outline the main features of the method and gain computational experience to guide future work.

An early time-dependent Monte Carlo method was proposed for the heat equation and was based on a stochastic interpretation of the explicit finite-difference equations [8]. This method used random walks on spatial grids, but the extension to grid-free walks was easy because the fundamental solution of the heat equation is Gaussian. These ideas remained mostly of theoretical interest due to the large variance of the methods. A step toward improving the accuracy was to have the density of diffusing elements represent the *gradient* of the solution, rather than the solution itself. Integrating to obtain the solution reduces the variance considerably. This lead to the coining of the term "gradient random walk" (GRW) [11].

The first of the GRW methods was Chorin's random vortex method (RVM) for incompressible two-dimensional (2-D) fluid flow [5]. Subsequently, Chorin proposed the "random

*Received by the editors February 15, 1993; accepted for publication (in revised form) September 14, 1993.

†National Institutes of Health, National Institute of Diabetes and Digestive and Kidney Diseases, Mathematical Research Branch, Bethesda, Maryland 20892 (sherman@helix.nih.gov).

‡Supercomputing Research Center, Institute for Defense Analyses, Bowie, Maryland 20715-4300 (mascagni@super.org, na.mascagni@na-net.ornl.gov).

element method" [7] for 1-D reaction-diffusion equations, which models diffusion via random walk and reaction via deterministic particle growth and decay. This method was studied further by Ghoniem and colleagues [21], [11], both in its own right and as a model for the vortex sheet algorithm [6]. Hald [14], [15] proved convergence for several simplified versions of the Chorin–Ghoniem algorithm, and Puckett [22] proved convergence of the full method for the Fisher/Kolmogorov–Petrovskii–Piskunov equation. Whereas Puckett's estimate of the error due to the Monte Carlo particle-based discretization was $O(N^{-1/4})$, where N is the number of particles, he presented numerical evidence that the error is actually $O(N^{-1/2})$. Sherman and Peskin [27] developed a variant of the Chorin–Ghoniem method, which was purely stochastic, and applied it to a simplification of the Hodgkin–Huxley nerve conduction equations [28]. Chauvin and Rouault [4] have proved the convergence of this variant.

The extension of the GRW method to higher dimensions began when Anderson [1] showed how to recover a scalar field in two dimensions from a δ-function representation of its gradient in a pure convection problem. Anderson's method relied on representing the solution as the Laplacian of the solution convolved with the Green function. Integration by parts gave an algorithm for the recovery of the solution at discrete points from this point-gradient representation. Sherman and Peskin [28] sketched how to apply Anderson's approach to reaction-diffusion equations. Fogelson [10] used ideas similar to Anderson's on a convection-diffusion problem, but instead of point recovery he used a fast Poisson solver to recover the function values.

In related work, Russo [24]–[26], Raviart [23], and others [16], [9] proposed particle methods for collisional equations. Their methods are deterministic, but share the Lagrangian feature of the GRW discretization.

Although the problem of large variance for the Monte Carlo method has not yet been solved, and these methods have not displaced finite-difference methods, they have stimulated much theoretical work.

A major motivation for studying Monte Carlo methods in this context is the deep connection between reaction-diffusion equations and stochastic processes. It is well known that certain functionals of Brownian motion have expected values that solve reaction-diffusion equations [18]. Moreover, the microscopic phenomenology of the chemical processes described by reaction-diffusion equations are based on the Brownian motion of the reacting species. Thus, it is intellectually appealing to search for numerical methods that share features in common with these fundamental viewpoints.

In addition to the connection between probability theory and partial differential equations (PDEs), the possibility of grid-free methods, especially for multidimensional problems, remains alluring. In this spirit we present the extension of the GRW algorithm for reaction-diffusion equations to two (or more) space dimensions and describe some of its computational properties.

In §2 we briefly review the 1-D GRW methods and derive their extension to two dimensions. In §3 we apply the method to track a traveling wavefront generated by Nagumo's equation without recovery. We illustrate two initial-value problems in the plane and an initial-boundary-value problem in a half-space. In §4 we summarize the results and discuss open problems and possible directions for future work.

2. Derivation. We first review the GRW method [7], [11], [27] for (1.1) with $d = 1$:

$$u_t = u_{xx} + f(u), \qquad u(x,0) = u_0(x). \tag{2.1}$$

We assume $u(-\infty, t) = 0, u(+\infty, t) = 1$. The strategy is to represent $v = u_x$ by diffusing particles. Differentiating (2.1) with respect to x,

$$v_t = v_{xx} + f'(u)v, \qquad v(x,0) = u_0'(x), \tag{2.2}$$

and discretizing v as a sum of δ-functions with strength m_j, we have

$$v(x,t) = \sum_{j=1}^{N} m_j \delta\left(x - X_j(t)\right), \tag{2.3}$$

where $X_j(t)$, $j = 1, \ldots, N$ represent the location of N particles. We use capital X to indicate that the positions are random variables. The density of the particles determines the value of v, and heuristically one thinks of m_j as the "mass" of the jth particle.

We recover u from v by $u(x,t) = \int_{-\infty}^{x} v(x',t)\,dx'$:

$$u(x,t) = \sum_{j=1}^{N} m_j H\left(x - X_j(t)\right). \tag{2.4}$$

Thus u is represented as a step function with jumps of size m_j at X_j. If all the particles have the same mass, m, the value of u at x is m times the number of particles that lie to the left of x. The boundary condition at $-\infty$ is automatically satisfied, and the condition at $+\infty$ is satisfied on average with fluctuations [27]. This corresponds to conservation of mass.

As described in [11], there is much less noise in the computed value of u than of v because all the particles contribute to the value at any point x. Moreover, if the particles have equal mass, their density is large precisely where u has large gradients.

Once the initial data is discretized, the GRW method evolves the particle positions and masses such that u satisfies (2.1). This is done by a fractional step iteration in which the diffusion term is modeled by a random walk and the reaction term is modeled as exponential growth or decay of the particle masses. For each timestep the sequence is as follows.

- Gaussian random walk step:

$$X_j(t + \Delta t) = X_j(t) + \sigma_j, \tag{2.5}$$

  where the σ_j are independent $N(0, 2\Delta t)$ random variables.
- Evaluate $u_j = u(X_j(t + \Delta t))$, $j = 1, \ldots, N$ using (2.4).
- Kill or replicate particles with probability $|f'(u_j)|\Delta t$:
  1. Kill particle if $f' < 0$;
  2. Create a new particle at X_j if $f' > 0$.

Note that the only way the particles interact with each other, and hence the only way that the nonlinearity of the equations is manifest in the algorithm, is that the local value of u depends on the positions of all the particles. Not surprisingly, this is the step that is most difficult computationally. Naively, computing (2.4) for all the X_j's requires $O(N^2)$ work, but if the particles are sorted, it requires only $O(N)$ work plus $O(N \log N)$ for the sorting.

The Chorin–Ghoniem method [7], [21], [11] is essentially the same except that instead of killing and replicating particles, the masses are increased or decreased according to the ordinary differential equation

$$\frac{dm_j}{dt} = f'(u_j). \tag{2.6}$$

In the mean and to $O(\Delta t)$ the two procedures are equivalent.

Both methods have been used to solve traveling-wave problems in one dimension. The mechanism of wave propagation is transfer of mass from behind the front to ahead of it. With randomized particle death and replication, particles are killed off behind the wavefront, where f' is mostly negative, and replicated ahead of the front, where f' is mostly positive. With

deterministic particle growth and decay, there is a graded transmission of mass from neighbor to neighbor.

We now generalize the method to two space dimensions. The key issue is how to recover u from its gradient, represented by particles. For this we follow a suggestion of Anderson [1] to employ Poisson's formula [17]:

$$u(\mathbf{x}, t) = \int G(\mathbf{x} - \mathbf{x}')\Delta u(\mathbf{x}', t)\, d\mathbf{x}', \tag{2.7}$$

where G is the fundamental solution of the Laplacian in $\mathbb{R}^2$:

$$G(\mathbf{x}) = \frac{1}{2\pi} \log |\mathbf{x}|. \tag{2.8}$$

Integrating by parts,

$$u(\mathbf{x}, t) = \int \nabla G(\mathbf{x} - \mathbf{x}') \cdot \nabla u(\mathbf{x}', t)\, d\mathbf{x}'. \tag{2.9}$$

As in one dimension, we represent ∇u as a sum of δ-functions:

$$\nabla u(x, y, t) = \sum_{j=1}^{N} m_j \delta\left(x - X_j(t), y - Y_j(t)\right) \mathbf{n}_j. \tag{2.10}$$

Now, in addition to position and mass, each particle has a unit vector, $\mathbf{n}_j = (\xi_j, \eta_j)$, representing the direction of ∇u at (X_j, Y_j).

Substituting (2.10) in (2.9) we obtain

$$u(x, y, t) = \frac{1}{2\pi} \sum_{j=1}^{N} \frac{\mathbf{r}_j \cdot m_j \mathbf{n}_j}{|\mathbf{r}_j|^2}, \tag{2.11}$$

where $\mathbf{r}_j = (x - X_j, y - Y_j)$. The sum (2.11) is very similar to that found in the RVM [5]. As in the RVM, if $(x, y) = (X_k, Y_k)$, the kth term is excluded from the sum, and a smoothing procedure is needed to avoid numerical instability when $(x, y) \approx (X_k, Y_k)$. See §3.

Although this method of recovering u from its gradient appears to be very different from the 1-D method, it is actually a natural generalization. In $\mathbb{R}^1$, with $G_{xx}(x - x') = -\delta(x - x')$, then $G_x(x - x') = H(x' - x)$, so substituting (2.3) into the 1-D equivalent of (2.9) gives exactly (2.4). From this point of view, one can reinterpret the positive and negative masses used to represent nonmonotonic functions in [28] as particles with orientation $+1$ for up-jumps and -1 for down-jumps.

We can extend the analogy a little further. In two dimensions one can represent u as a step function by constructing a contour plot and replacing u by a function that jumps in value whenever a contour is crossed. Thus we need to be able to represent a function that is 0 outside a closed curve Γ and 1 inside. This is accomplished by placing N particles along Γ, oriented along the outward normal with mass

$$m = \frac{\text{length}(\Gamma)}{N}. \tag{2.12}$$

Then (2.11) is an approximation to the line integral

$$\frac{1}{2\pi} \oint_{\Gamma} \nabla G \cdot \mathbf{n}\, ds. \tag{2.13}$$

Applying the divergence theorem, (2.13) evaluates to 0 if (x, y) is outside Γ and 1 if (x, y) is inside. Equation (2.12) says that each particle represents an oriented arc with a given length and implies that the larger the mean radius of Γ, the more particles of a given mass are required to represent a jump of given height across Γ. In $\mathbb{R}^3$, particles would represent oriented patches of surface area, and so on for arbitrary dimension.

Figure 2.1 shows the u-surface recovered from placing 1, 2, 4, and 10, 000 particles equiangularly on the circle of radius 10. These figures illustrate that the particles can be thought of as oriented steps only in a collective sense: Each one individually is a singular dipole, and only by cancellation can a simple step be constructed.

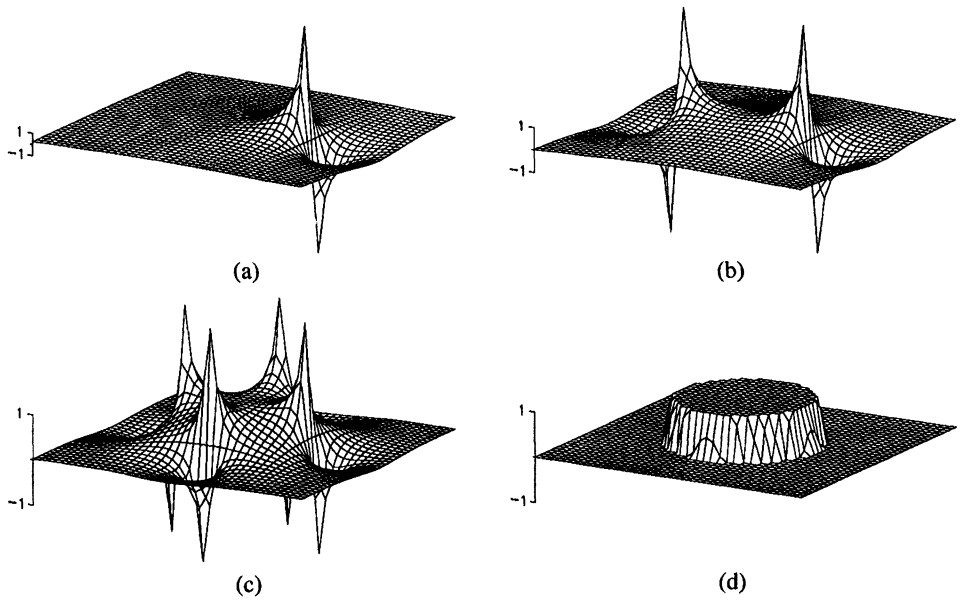


FIG. 2.1. *Representing a step with gradient particles: u-surface recovered using* (2.11) *without smoothing from* (a) 1, (b) 2, (c) 4, *and* (d) 10, 000 *particles equally spaced around a circle of radius* 10. *Note changes of scale: u-axis scales for* (a), (b), (c), *and* (d), *are in the ratio* 1:2:4:4. *The grid covers the square from* $(-20, -20)$ *to* $(20, 20)$.

Moreover, the interpretation of the gradient particles as steps is only valid at $t = 0$; after the first random walk step the particles will not lie on any meaningful curve. The Poisson formula representation, (2.11), is more general than (2.13), however, and continues to apply to any collection of particles in the plane.

After discretizing the initial data, one uses the same fractional step iteration as in $\mathbb{R}^1$, replacing (2.5) with independent Gaussian steps for X_j and Y_j:

- Gaussian random walk step:

$$X_j(t + \Delta t) = X_j(t) + \sigma_j^x,$$

$$Y_j(t + \Delta t) = Y_j(t) + \sigma_j^y,$$

  where σ_j^x and σ_j^y are independent $N(0, 2\Delta t)$ random variables.
- Evaluate $u_j = u(X_j(t + \Delta t), Y_j(t + \Delta t))$, $j = 1, \ldots, N$ using (2.11).
- Kill or replicate particles with probability $|f'(u_j)|\Delta t$:
  1. Kill particle if $f' < 0$;

2. Create a new particle at (X_j, Y_j) if $f' > 0$.

The differences between this 2-D algorithm and the 1-D algorithm given above are that: (1) a 2-D random walk is performed and (2) the Green function of the Laplacian in two dimensions is used to compute u on the particles using (2.11). To extend this to d-dimensions we merely use a d-dimensional random walk and substitute the Green function of the d-dimensional Laplacian in (2.11) to perform the recovery:

$$u(\mathbf{x}, t) = \frac{1}{\omega_d} \sum_{j=1}^{N} \frac{\mathbf{r}_j \cdot m_j \mathbf{n}_j}{|\mathbf{r}_j|^d}, \tag{2.14}$$

where ω_d is the area of the unit ball in $\mathbb{R}^d$.

As in one dimension, the most expensive and algorithmically challenging step is evaluating u at the particles because it is equivalent to calculating the interactions in an N-body problem. Unlike the 1-D case, one cannot circumvent this difficulty by sorting the particles. A partial solution is to apply the fast multipole method of Greengard and Rokhlin [13], [12], which can calculate the $O(N^2)$ interactions in $O(N)$ steps. Specifically, we used their subroutine `rapif2` to compute the product of the complex Hilbert matrix, $H_{jk} = 1/(z_j - z_k)$, $z_j = X_j + iY_j$ with a complex vector, $q_k = \xi_k + i\eta_k$. Notice that q_k is a complex representation of the direction of the kth gradient particle. Then, u is $\frac{m}{2\pi}\Re(Hq)$. The multipole method proved to be much faster than the direct calculation of (2.11), but we had difficulty with smoothing (see §3).

We now give results of applying the GRW method to some simple test problems.

3. Numerical examples.

Example 1. We begin with a pure initial value problem: (1.1) with $d = 2$ and $f(u) = u(1-u)(u-a)$, $a = 0.25$. This is Nagumo's equation without recovery [19]. The initial data is 1 inside a circle of radius $R_0 = 10$ and 0 outside:

$$u(\mathbf{x}, 0) = \begin{cases} 1 & \text{if } |\mathbf{x}| \le R_0, \\ 0 & \text{if } |\mathbf{x}| > R_0. \end{cases} \tag{3.1}$$

With this initial data the solution takes the form of an "excited region" (where $u \approx 1$), which expands radially and asymptotically approaches a traveling wave as the radius $\to \infty$.

To solve this problem, we place $N = 10,000$ particles, each with mass $2\pi R_0/N$, on the circle of radius $R_0 = 10$ at $T = 0$ and advance up to $T = 25$ with a timestep of $\Delta t = 0.1$. The initial data is shown in Fig. 2.1(d), and Figs. 3.1(a), (b) show u as a stacked contour plot at $T = 5$ and 25. Figure 3.1(c) shows the expansion of the $u = 0.5$ contour in time. It retains approximate cylindrical symmetry although this is not imposed in the algorithm. Although we have evaluated the solution on a rectangular grid for display purposes, the particles cluster in a narrow band around the $u = 0.5$ contour (Fig. 3.4(c), (d), Example 3).

As indicated in §2, it was necessary to smooth the singularity in (2.11) when particles approached each other closely, or when evaluating the solution at a grid point near a particle. We thus replaced (2.11) with

$$u(x, y, t) = \frac{m}{2\pi} \sum_{j=1}^{N} \frac{\mathbf{r}_j \cdot \mathbf{n}_j}{\max(|\mathbf{r}_j|^2, \varepsilon^2)}. \tag{3.2}$$

The choice $\varepsilon = 0.3$ gave good results.

For comparison with the exact solution we solved the radial problem,

$$u_t = u_{rr} + \frac{1}{r} u_r + f(u), \tag{3.3}$$

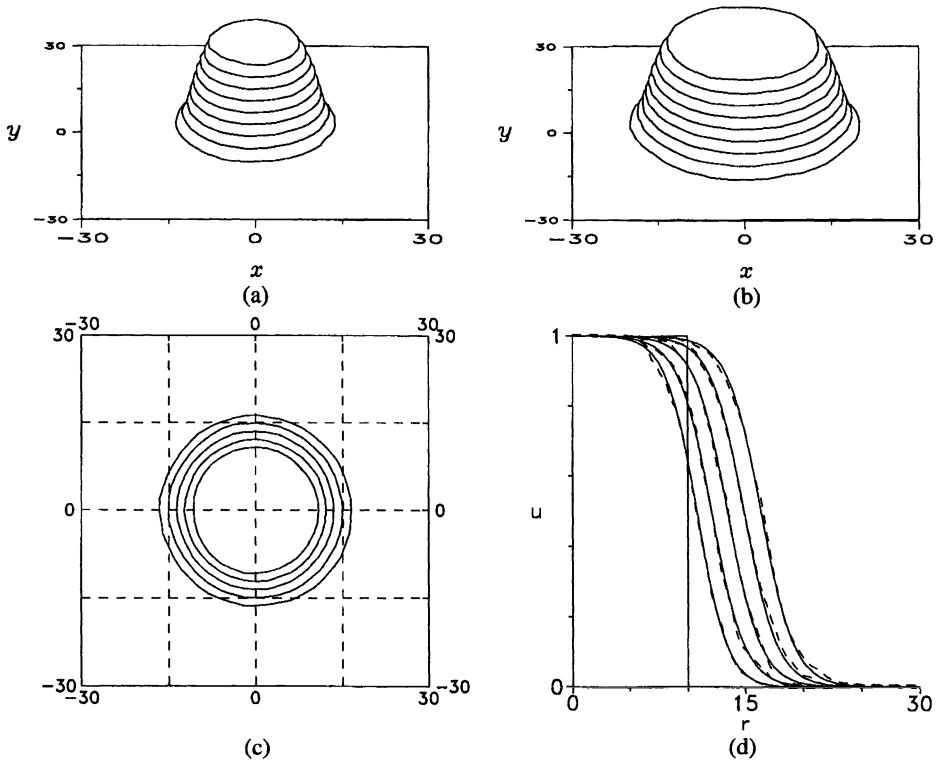


FIG. 3.1. *Example* 1. *Initial value problem with* $u = 1$ *inside a circle of radius* 10, *and* 0 *outside. Initially* 10,000 *particles are placed along the circle, and more particles are automatically added as the front expands (Fig.* 3.2). *Stacked contour plot of* u *at* (a) $T = 5$ *and* (b) $T = 25$. *Contour levels are* 0.1, 0.2, ..., 0.9. (c) *Contour for* $u = 0.5$ *at* $T = 5, 10, \ldots, 25$. (d) *Comparison at* $T = 0, 5, \ldots, 25$ *of* GRW *profiles (dashed) along ray* $\theta = 0$ *with the "exact" solution (solid) computed by a finite difference method for the radial* (3.3).

with a finite-difference method using a small uniform mesh. Figure 3.1(d) shows the wave profile of the GRW solution as compared to the finite-difference solution along a ray. Although the shape of the profile fluctuates, the wave speed is computed accurately.

As in the 1-D algorithm, the propagation of the wave is the result of transfer of mass. In one dimension, total particle mass and particle number, N, are conserved, so that, in effect, particles jump across the front. In two dimensions, total mass is still conserved (u remains near 1 at the origin), but particle number grows with time. Since in a radially symmetric traveling wave, the circumference of the front grows linearly, (2.12) suggests that N must also increase linearly. After a transient, during which the shape of the profile relaxes, linear growth is observed (Fig. 3.2). Note that N increases faster than the average radius, so that the effective arc-length per particle decreases with time. We believe that this is because the particles lose their radial orientation as they diffuse. If at each timestep the particles are reoriented to point radially outward, then N grows at the same rate as the radius (not shown).

The calculation for Fig. 3.1 was programmed in C on a Cray YMP and required about 1.5 CPU hours, virtually all of which was spent computing the readily vectorized N-body sum 3.2. The same calculation took only one-fifth as long using the fast multipole FORTRAN subroutine, `rapif2`, even though this mostly did not vectorize. However, this level of performance was attainable only if no smoothing was done, and the resulting solutions (not shown) suffered from large spike-like errors. Smoothing is implemented by calling a close approach subroutine,

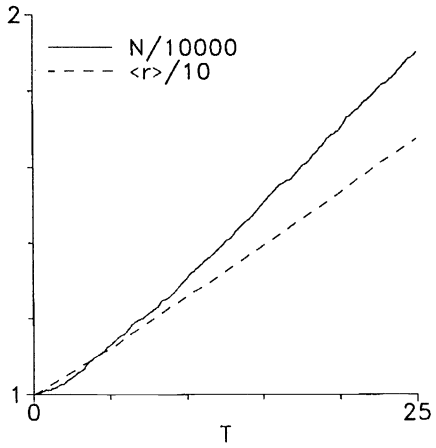


FIG. 3.2. *Growth of particle number with time. Solid curve: The number of particles, normalized by the initial number. Dashed curve: The average radius of the wave (computed as the simple average of the radii of the particles), normalized by the initial radius. Both grow approximately linearly, but N grows faster. Thus, the effective arc-length per particle declines slowly.*

which computes the interactions directly when $|\mathbf{r}| < \varepsilon$. This degrades performance: with $\varepsilon = 0.3$ instability was avoided, but `rapif2` actually took longer than the direct N-body calculation, presumably because the subroutine call inhibited vectorization.

Example 2. The next example addresses the question of what happens when the shape of the excited region changes in time. Fogelson [10] used the same discretization as in Example 1 to represent a circular step of concentration around a localized source, which stretches and elongates into an ellipse due to convection and diffusion. In the case of pure convection he showed that the particles would remain on and normal to the boundary of the region, Γ, because it is the material curve separating regions of high and low concentration. The particles rotate because of the derivatives of the convective term. There are no terms in reaction-diffusion that can turn the normals, which raises the question of how new emerging directions can be represented.

Reconsider Example 1, but with the excited region initially a square. Asymptotically, the square again evolves into a circular traveling wave. Figure 3.3(a) shows the contours for $u = 0.5$ at $T = 5, 10, \ldots, 25$. The shape of the front transforms from a square to a circle. The calculation is in reasonable agreement with results of a 2-D finite-difference method (not shown). The curved front is represented by linear combinations of the four existing directions in the particle population. Diffusion combines with growth and decay of particles to produce, on average, the appropriate mix of particles in the required locations. Note that this depends on the linearity of (2.11), which is inherited from (2.9). Figure 3.3(b) shows the distribution of particle directions (actually the difference between the argument of the particle and the direction of its normal) at $T = 25$, when the front is approximately circular. The distribution closely approximates a cosine, which is sufficient to represent a circular front.

Example 3. In order to illustrate a simple example with boundary conditions we solved (1.1) on the half-space $x > 0$, appending the boundary condition $u_x(0, y) = 0$. This is done by the method of images: a particle at (X_j, Y_j) with orientation (ξ_j, η_j) has an image particle at $(-X_j, Y_j)$ with orientation $(-\xi_j, \eta_j)$. The sum (2.11) is modified to account for the influence of the image particles on the real particles; it is of course not necessary to compute u at the image particles. This calculation is equivalent to two excited regions coalescing, and we have

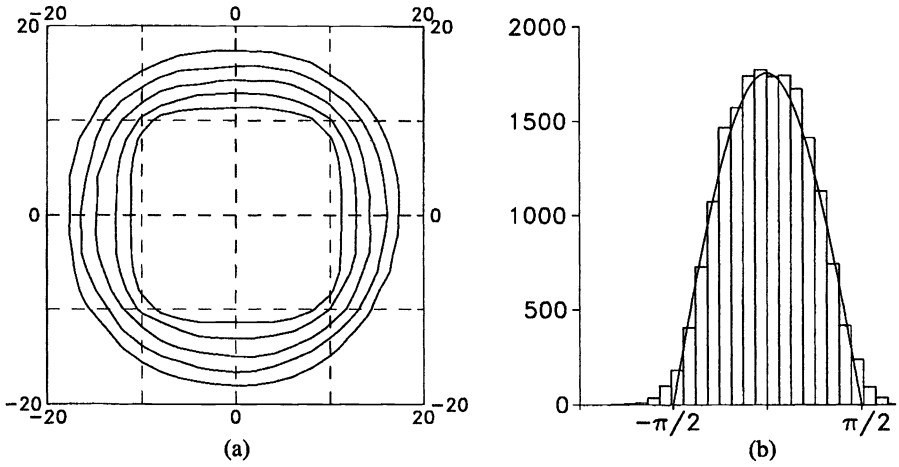


FIG. 3.3. *Example* 2. *Initial value problem with the initial excited region a square circumscribing a circle of radius* 10. (a) *Contours for* $u = 0.5$ *at* $T = 5, 10, \ldots, 25$ *for the* GRW *method.* (b) *Histogram (numbers of particles) versus* $\tan^{-1}\frac{y}{x} - \tan^{-1}\frac{\eta}{\xi}$; *that is, the difference between the argument of the particle and the direction of its normal. The distribution approximates a scaled cosine, shown superimposed.*

displayed it as such in Fig. 3.4. Figure 3.4(a) shows the $u = 0.5$ contours as a function of time, indicating the two regions expanding and merging into one large region. Figure 3.4(b) is a surface plot at $T = 15$. Figures 3.4(c), (d) show the particle positions in the plane to highlight how the method adapts to the solution: The cloud of particles is confined to a narrow band around the front, and, as the regions merge, the particles along the y-axis, which are no longer needed to resolve the solution, are killed off.

Example 4. As a final example, we indicate how to represent more general initial conditions. Consider the heat equation,

$$u_t = \Delta u,$$

with Gaussian initial data and solution

$$u(x, t) = \frac{1}{4\pi(\sigma + t)} e^{-(x^2+y^2)/4(\sigma+t)}.$$

In this example we choose $\sigma = 4$.

We discretize with uniform step size δu along the u direction and construct the corresponding level sets Γ_i (here, circles). By (2.12), each curve Γ_i should contain particles with combined mass

$$M_i = \text{length}(\Gamma_i)\delta u = 2\pi R_i \delta u.$$

The total mass of all the level sets $M = \sum M_i$ approximates the volume under $u(x, 0)$. The particle representation can be uniquely specified by distributing N particles with equal mass, $m = M/N$. Thus, Γ_i gets $N_i = \lfloor M_i/m \rfloor = \lfloor (NM_i/M) \rfloor$ particles. To avoid fractional particles, the mass per particle on Γ_i can be adjusted slightly: $m_i = M_i/N_i$. An alternative is to use random roundoff. The particles are spaced uniformly in arc-length (here, equiangularly) and have the direction of the gradient, which is the outward normal to Γ_i (here, radial).

An alternative method is to distribute particles on a rectangular grid. With uniform spacing h, total mass equal to $|\nabla u(x_i, y_j)|h^2$ is placed at the grid point (x_i, y_j) and subdivided into

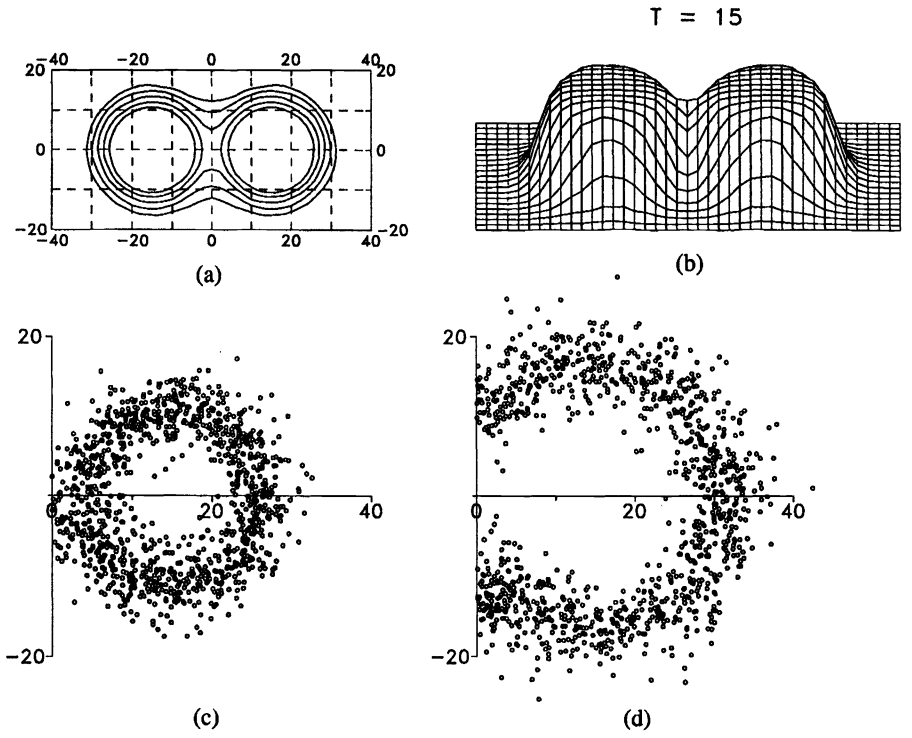


FIG. 3.4. *Example* 3. *Expanding front in the half space* $x > 0$ *with* $u_x(0, y) = 0$, *solved by using image particles.* (a) *Contours for* $u = 0.5$ *at* $T = 5, 10, \ldots, 25$. (b) *Surface plot of* u *at* $T = 15$. (*Plots reflect the influence of both the real particles and their images.*) *Positions of the actual particles* (*only* $\frac{1}{10}$ *of the particles are shown for clarity*) *at* (c) $T = 5$ *and* (d) $T = 25$, *showing that the particles follow the wave front, and that the superfluous particles along* $x = 0$ *are eliminated automatically.*

particles, each with direction $-\nabla u(x_i, y_j)$. This is analogous to initialization in the RVM when an initial vorticity distribution is given. This alternative is easy to implement for our example because ∇u is known analytically, but the level set method is more natural and works better given the symmetry of the problem. One problem with this approach arises when the initial function is given in tabular form. The numerical derivative required to compute the initial gradient reduces the numerical accuracy of the initial conditions passed to the GRW by $O(h^{-1})$.

In the level set approach there are two free numerical parameters, the number of level curves, and the total number of particles; these determine the number of particles per level curve. One must then balance coverage in the radial and azimuthal directions. We have not studied the optimal tradeoff. Figure 3.5(a) shows the representation of the initial data with 100 level curves and initial $N = 10{,}000$ (9948 after adjustments), and Fig. 3.5(b) shows the exact and computed profiles along a ray at $t = 0$ and $t = 4$.

4. Issues for future consideration. We have taken a step toward our goal of a grid-free, adaptive method for reaction-diffusion equations in two or more space dimensions by generalizing a family of 1-D random gradient methods [27], [28], [7], [21], [11]. Example 3 shows spatial adaptivity: The particles are concentrated around the steep gradient of the wavefront (Fig. 3.4(c)) and, when the gradient vanishes due to interaction with a boundary (or another expanding excited region), the particles defining that region die off (Fig. 3.4(d)). Example 2

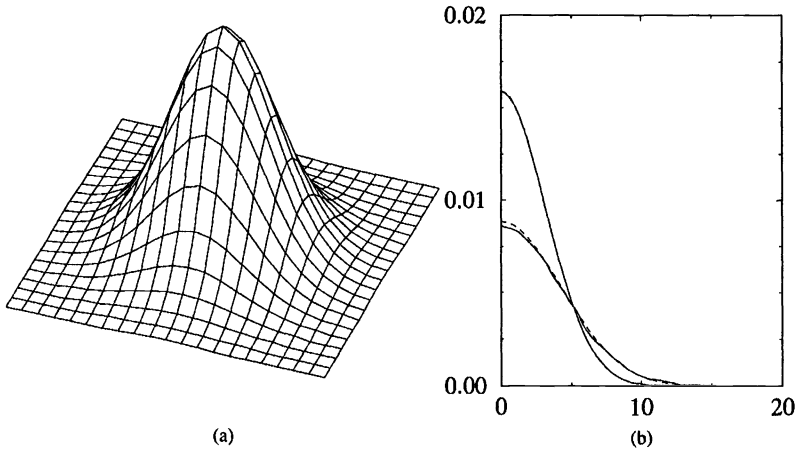


FIG. 3.5. *Example* 4. *Solution of the heat equation with Gaussian initial data.* (a) *Initial surface represented by radially oriented particles along* 100 *level curves uniformly spaced in the u direction. A total of* 9948 *particles were used.* (b) *Comparison of exact (solid) and computed (dashed) profiles along the positive x-axis at time* $t = 0$ *(upper curves) and* $t = 4$.

demonstrates adaptivity with respect to particle directions: Directions not represented in the initial data are represented by linear combinations of existing particles.

We conclude with a discussion of the limitations of the method and suggestions for improvements.

One good feature of the 1-D discretization, which does not generalize to two dimensions, is monotonicity. For equations that have monotonic solutions, such as Nagumo's equation without recovery, the numerical method is guaranteed to have monotonic solutions. This greatly constrains the set of solutions, preventing oscillatory instabilities, for example. In two dimensions there is no such guarantee, and the solutions may go negative as well as exceed 1. Fortunately, in our case $f' < 0$ at $u = 0, 1$, and these errors damp out, but it is evident from Fig. 2.1 that a great deal of cancellation is involved in representing smooth solutions by singular gradient patches.

The main drawback of the GRW method is that large numbers of particles are needed. Although we have not done systematic timing studies, we can estimate how the efficiency of the GRW method with direct calculation of the particle interactions compares to finite-difference methods. Using a 2-D (nonradial) Euler method, an answer more accurate than the GRW solution of Fig. 3.1 can be obtained in 3.6 Cray YMP CPU seconds. The GRW is at least 3 orders of magnitude slower. Accuracy was assessed by comparing along a ray both solutions with that of a second order 1-D (radial) solver using a fine grid. Improvement requires reducing N and speeding up the calculation of the particle interactions.

The GRW may not choose the most economical representation of the solution. In Example 4 the accuracy decreased with time, primarily because the particles lose their radial orientation as they diffuse. In Example 1 we found that efficiency was enhanced by rotating the particles so that they would always point radially. Even then, more particles are needed to resolve the wave as the radius increases.

In order to address this problem, we have experimented with the Chorin–Ghoniem method, which permits masses of individual particles to change with time while GRW conserves particle mass [7], [21]. The result is that in Chorin–Ghoniem while particle number is conserved, individual particles can grow or shrink exponentially in time. This leads to the need to remove

particles that have practically vanished and to split up particles that have grown too massive. In that case, however, the number of particles grows with time just as in the stochastic growth and decay algorithm. We note that Fogelson [10] found that no instability was caused in the convection-diffusion problem by allowing the masses to grow unchecked. This may reflect inherent differences between the convection-dominated and diffusion-dominated cases. We conclude that Chorin–Ghoniem has no practical advantage over GRW in reaction-diffusion problems.

Another modification that could both limit the growth of particles and mitigate loss of optimal particle direction is to recontour the computed solution numerically and reassign particles every so often. That is, N could be reduced by redistributing the mass among a smaller number of particles. This is related to the "rezoning" technique that has been effectively applied to the RVM to reduce the variance of long time solutions [20]. One could apply rezoning by redistributing mass onto regular grids as in the RVM, but this is more complicated in the case of the 2-D GRW because one must compute both the x and y components of the gradient at each point rather than the scalar vorticity.

Another more speculative possibility is to view particle "evolution" in biological terms: particles that help define the solution proliferate while unneeded ones die. Thus, it may be beneficial to introduce "mutations" by randomly or heuristically perturbing the directions. This might also help in problems like Example 2 with direction-deficient initial conditions.

Because of the strong formal similarity between the GRW and RVM [5] methods, much of the machinery that has been developed for the RVM can be applied to the GRW method. For example, a direct implementation of the algorithm requires the computation of $O(N^2)$ interactions between particles at each timestep. We have applied the Greengard–Rokhlin [12], [13] algorithm to accomplish this in $O(N)$ arithmetic operations, increasing the efficiency by almost an order of magnitude. As noted above, however, we had difficulties implementing adequate smoothing to avoid numerical instability without at the same time destroying the computational efficiency. An interesting alternative is to use the Barnes–Hut algorithm to do the N-body calculation [2]. While only $O(N \log N)$ complex, Barnes–Hut permits the use of high-order cut-off functions [3] in a more integral way than Greengard–Rokhlin. Thus Barnes–Hut offers the possibility of incorporating smoothing in a subquadratic N-body algorithm. Further research is required in this area.

An alternative to the multipole method to overcome the quadratic complexity of the GRW would be to do, say, 10 independent runs with 1,000 particles instead of 1 run with 10,000 particles. The solution at a fixed set of grid points would be obtained by averaging. If the variance is $O(N^{-1})$ as in the 1-D GRW methods, then averaging would yield a solution as accurate as that of the single large simulation at 1/10 the cost. Such an approach also would parallelize in a natural way.

A final, and promising, possibility is to model the diffusion deterministically and thereby replace the random walk with a deterministic motion. This approach has already been applied to simple diffusion and convection-dominated diffusion problems in one and two dimensions, [25], [24], [9], and to collisional dynamics [26], [16]. A distinct advantage of such methods is that deterministic movement of particles to simulate diffusion eliminates the statistical fluctuations of a Monte Carlo method. Instead, these methods have a (deterministic) discretization error associated with their approximate diffusion process. In one dimension, the deterministic algorithms share much in common with the GRW methods, as they are grid free and automatically adaptive. In two dimensions, the existing deterministic methods compute the approximate derivatives required by finite differences on the moving grid defined by the particle coordinates. It is hoped that a variant of a deterministic method can be found that does not require the construction of a moving grid to compute the particle trajectories.

Acknowledgments. The bulk of the calculations were performed on a Cray YMP-8/256 computer at the Advanced Scientific Computing Laboratory of the National Cancer Institute's Frederick Cancer Research and Development Facility. Code development and other computing was done using an IBM 3090-VF600 at the National Institutes of Health's Division of Computer Resources and Technology and a TMC CM-2 at the Advanced Computing Laboratory at the Los Alamos National Laboratory.

We gratefully thank Leslie Greengard for making available the program `rapif2` which implements the fast multipole method [13], [12]. We also thank Matt Wilson for the visualization tools `xplot` and `xview`. Finally, we thank Robert Krasny for pointing out long ago that Anderson's method could be adapted to this problem.

REFERENCES

[1] C. R. ANDERSON, *A vortex method for flows with slight density variation*, J. Comput. Phys., 61 (1985), pp. 417–444.

[2] J. E. BARNES AND P. HUT, *A hierarchical $O(N \log N)$ force-calculation algorithm*, Nature, 324 (1986), pp. 446–449.

[3] J. T. BEALE AND A. MAJDA, *High order accurate vortex methods with explicit kernels*, J. Comput. Phys., 58 (1985), pp. 188–208.

[4] B. CHAUVIN AND A. ROUAULT, *A stochastic simulation for solving scalar reaction-diffusion equations*, Adv. in Appl. Probab., 22 (1990), pp. 88–100.

[5] A. J. CHORIN, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.

[6] ———, *Vortex sheet approximation of boundary layers*, J. Comput. Phys., 27 (1978), pp. 428–442.

[7] ———, *Numerical methods for use in combustion modeling*, in Computing Methods in Applied Sciences and Engineering, R. Glowinski and J. L. Lions, eds., North Holland, Amsterdam, 1980, pp. 229–235.

[8] R. COURANT, K. FRIEDRICHS, AND H. LEWY, *Über die partiellen differenzengleichungen der mathematischen physik*, Math. Ann., 100 (1928), pp. 32–74.

[9] P. DEGOND AND F.-J. MUSTIELES, *A deterministic approximation of diffusion equations using particles*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 293–310.

[10] A. L. FOGELSON, *Particle-method solution of two-dimensional convection-diffusion equations*, J. Comput. Phys., 100 (1992), pp. 1–16.

[11] A. F. GHONIEM AND F. SHERMAN, *Grid-free simulation of diffusion using random walk methods*, J. Comput. Phys., 61 (1985), pp. 1–37.

[12] L. GREENGARD, *The Rapid Evaluation of Potential Fields in Particle Systems*, M.I.T. University Press, Cambridge, MA, 1988.

[13] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[14] O. E. HALD, *Convergence of random methods for a reaction-diffusion equation*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 85–94.

[15] ———, *Convergence of a random method with creation of vorticity*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 1373–1386.

[16] F. HERMELINE, *A deterministic particle method for transport diffusion equations: Application to the Fokker-Planck equation*, J. Comput. Phys., 82 (1989), pp. 122–146.

[17] F. JOHN, *Partial Differential Equations*, 4th Edition, Springer-Verlag, New York, 1982, p. 99.

[18] H. P. MCKEAN, *Application of Brownian motion to the equation of Kolmogorov-Petrovskii-Piskunov*, Comm. Pure Appl. Math., 28 (1975), pp. 323–331.

[19] J. NAGUMO, S. ARIMOTO, AND S. YOSHIZAWA, *An active pulse transmission line simulating nerve axon*, Proc. IRE, 50 (1962), pp. 2061–2070.

[20] H. O. NORDMARK, *Rezoning for higer order vortex methods*, J. Comput. Phys., 97 (1991), pp. 366–397.

[21] A. K. OPPENHEIM AND A. GHONIEM, *Application of the random element method to one-dimensional flame propagation problems*, AIAA-83-0600, AIAA 21st Aerospace Sciences Meeting, Reno, NV, 1983.

[22] E. G. PUCKETT, *Convergence of a random particle method to solutions of the Kolmogorov equation* $u_t = \nu u_{xx} + u(1-u)$, Math. Comp., 52 (1989), pp. 615–645.

[23] P. A. RAVIART, *An analysis of particle methods*, in Numerical Methods in Fluid Dynamics, Lecture Notes in Mathematics 1127, Springer-Verlag, New York, 1985, pp. 243–324.

[24] G. RUSSO, *Deterministic diffusion of particles*, Comm. Pure Appl. Math., 43 (1990), pp. 697–733.

[25] G. RUSSO, *A Lagrangian method for collisional kinetic equations*, in Proc. 1988 AMS SIAM Summer Seminar, Colorado State University, Fort Collins, CO, July 18–29, 1988, E. L. Allgower and K. Georg, eds., Lectures in Applied Mathematics 26, American Mathematical Society, Providence, RI, 1990, pp. 519–539.

[26] ———, *A particle method for collisional kinetic equations. I. Basic theory and one-dimensional results*, J. Comput. Phys., 87 (1990), pp. 270–300.

[27] A. S. SHERMAN AND C. S. PESKIN, *A Monte-Carlo method for scalar reaction diffusion equations*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 1360–1372.

[28] ———, *Solving the Hodgkin-Huxley equations by a random walk method*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 170–190.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1294–1310, November 1994

003

COMPUTING GAUSSIAN LIKELIHOODS AND THEIR DERIVATIVES FOR GENERAL LINEAR MIXED MODELS*

RUSS WOLFINGER†, RANDY TOBIAS†, AND JOHN SALL†

Abstract. Algorithms are described for computing the Gaussian likelihood or restricted likelihood corresponding to a general linear mixed model. Included are arbitrary covariance structures for both the random effects and errors. Formulas are also given for the first and second derivatives of the likelihoods, thus enabling a Newton–Raphson implementation. The algorithms make heavy use of the Cholesky decomposition, the sweep operator, and the W-transformation. Also described are the modifications needed for variance profiling, Fisher scoring, and MIVQUE(0), as well as the computational order of the procedures.

Key words. Cholesky decomposition, Fisher scoring, MIVQUE(0), Newton–Raphson, profiling, sweep, W-transformation

AMS subject classifications. 62J10, 62J99

1. Introduction.

1.1. The model. The most common statistical model is the general linear model, which has the following signal-plus-noise form:

$$y = X\beta + \epsilon.$$

Here y represents a known data vector of length n, β is a vector of p unknown parameters with known design matrix X, and ϵ is an unknown error vector. For inference purposes, ϵ is usually assumed to have a Gaussian (normal) distribution with mean 0 and variance matrix $\sigma^2 I$, where I is the $n \times n$ identity matrix. The model is "general" in the sense that the columns of X may consist of either known explanatory variables, as in regression, or dummy variables with 0s and 1s indicating the presence or absence of an effect, as in analysis of variance (ANOVA). The parameters in β are the objects of primary interest, and they are usually estimated using the method of least squares. Classical statistical inference via t- and F-tests are then possible under the above Gaussian assumption.

Probably the most attractive feature of the general linear model is its considerable flexibility in modeling the signal (in the form of the mean) of the data. However, the assumption that the variance matrix of the noise vector equals $\sigma^2 I$ is often too limiting, as under normality this implies that each observation is statistically independent.

The general linear mixed model lifts this restriction by taking the form

$$y = X\beta + Z\upsilon + \epsilon,$$

where y, X, β, and ϵ are as above, and υ is a vector of g unknown parameters with known design matrix Z. The unknown parameters in β and υ correspond to *fixed-effects* and *random-effects*, respectively, and the inclusion of both defines a *mixed model.* The parameters in β are still often the objects of primary interest, with their estimation being an important analysis goal. The parameters in υ are considered to be random variables, and thus account for variability in the data beyond that modeled in ϵ. To be more precise about this variance modeling, assume υ and ϵ have Gaussian distributions with 0 expectations and

*Received by the editors August 10, 1992; accepted for publication (in revised form) September 14, 1993.

†SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513 (sasrdw@unx.sas.com, sasrdt@unx.sas.com, sall@sas.com).

$$Var\begin{bmatrix} \upsilon \\ \epsilon \end{bmatrix} = \begin{bmatrix} G & 0 \\ 0 & R \end{bmatrix},$$

where both G and R are nonsingular. As such, the variance of y is

$$V = ZGZ' + R.$$

The above general definition of the mixed model subsumes that of the classical linear mixed model, in which $R = \sigma^2 I$ and G is a diagonal matrix of variance components [12], [13]. The simplest example of the classical case is the randomized block ANOVA, in which the experimental units occur in clusters, or *blocks*, and each unit is subjected to a level of some experimental factor, the *treatment*. Treatment is the fixed effect, and so X consists of t columns of dummy variables, where t is the number of treatment levels, each column indicating the treatment level of the observations. Similarly, block is the random effect, and Z consists of b dummy columns, where b is the number of blocks, each column indicating block membership. Finally, G equals σ_b^2 times the $b \times b$ identity matrix, σ_b^2 being the variance component for blocks.

Other common statistical analyses falling within this unifying paradigm include the covariance structure approach to repeated measures [11], split-plot and incomplete-block designs [20], MANOVA, seemingly unrelated regressions [23], random coefficients [22], shrinkage estimators, and best linear unbiased predictors [15].

This article presents the details of fitting mixed models with arbitrarily parameterized covariance structures for both G and R. The difficulty is in estimating the parameters of G and R, and many possible estimation methods exist. Several traditional approaches use the method of moments, which solves a system of equations relating expected mean squares to observed ones. However, these methods are often only applicable to classical mixed models for balanced data, that is, data with equal numbers of observations for each random effect. Instead, we make use of the Gaussian assumption and employ the method of maximum likelihood as well as one of its variants, restricted/residual maximum likelihood (REML) [6].

Unfortunately, except in special cases, the maximum likelihood or REML estimates of the parameters in G and R must be computed by numerical methods. The resulting problems can be very computationally intensive, and only in the past decade have sufficient computer resources been routinely available to conquer them. This article shows how to efficiently compute the likelihoods and their first two derivatives, thus enabling a Newton–Raphson optimization scheme. The techniques are implemented in the SAS/STAT MIXED procedure [16].

1.2. Outline. Section 2 presents the details of computing the likelihood and the restricted likelihood of the general linear mixed model. Section 3 contains formulas for the first and second derivatives of these likelihoods, including those needed for variance profiling and Fisher scoring. MIVQUE(0) initial estimates are the topic of §4 and computational order that of §5.

2. Likelihoods. Given particular covariance structures for G and R, let θ denote the vector of unknown parameters in V; assume it is of length q. We adopt a normal-theory maximum-likelihood approach to estimation [6], [2], [10], [19]. This approach is preferred to the traditional ANOVA method [18].

2.1. General form of likelihoods. The negative of twice the Gaussian log likelihood for the general linear mixed model is

$$-2l(\beta, \theta|y) = \log|V(\theta)| + (y - X\beta)'V^{-1}(\theta)(y - X\beta) + n\log 2\pi.$$

Minimizing this expression analytically for β yields

$$b(\theta) = [X'V^{-1}(\theta)X]^{-1}X'V^{-1}(\theta)y$$

and substitution into the original equation produces the negative of twice the profile/concentrated log likelihood for θ:

$$-2l(\theta|y) = \log|V(\theta)| + [y - Xb(\theta)]'V^{-1}(\theta)[y - Xb(\theta)] + n\log 2\pi.$$

Profiling means reducing the dimension of an objective function by analytic substitution. We perform it so that the numerical optimization can be carried out only over the parameters in θ. Assuming $\hat{\theta}$ is the resulting optimum, β is estimated by $b(\hat{\theta})$. The larger the dimension of β, the greater the savings in time profiling offers over the full numerical optimization on (β, θ).

The formula for the negative of twice the restricted/residual log likelihood by definition does not involve β, and is given by

$$\begin{aligned}-2l_R(\theta|y) = {} & \log|V(\theta)| + [y - Xb(\theta)]'V^{-1}(\theta)[y - Xb(\theta)] \\ & + \log|X'V^{-1}(\theta)X| + (n - p)\log 2\pi.\end{aligned}$$

The final two expressions above are the objective functions for maximum likelihood and REML, respectively. Ignoring the constant terms, they can be written as

$$\begin{aligned}-2l(\theta|y) &= l_1(\theta) + l_2(\theta), \\ -2l_R(\theta|y) &= l_1(\theta) + l_2(\theta) + l_3(\theta),\end{aligned}$$

where

$$\begin{aligned}l_1(\theta) &= \log|V(\theta)|, \\ l_2(\theta) &= r(\theta)'V^{-1}(\theta)r(\theta), \\ l_3(\theta) &= \log|X'V^{-1}(\theta)X|,\end{aligned}$$

and

$$r(\theta) = y - Xb(\theta).$$

2.2. Factoring and profiling the residual variance. It is possible to proceed one step further analytically by factoring out one parameter from V; this factorization may or may not be natural and desirable, depending on the structure of R. Assuming that it is desirable, denote the one parameter by σ^2, and let θ^* be the new remaining parameter vector, with $q - 1$ elements. A common example is a diagonal G with variance components as diagonal elements and $R = \sigma^2 I$. In this case θ^* contains each unknown variance component divided by σ^2. The likelihoods become

$$\begin{aligned}-2l(\theta^*, \sigma^2|y) &= n\log\sigma^2 + l_1(\theta^*) + l_2(\theta^*)/\sigma^2 + n\log 2\pi, \\ -2l_R(\theta^*, \sigma^2|y) &= (n - p)\log\sigma^2 + l_1(\theta^*) + l_2(\theta^*)/\sigma^2 + l_3(\theta^*) + (n - p)\log 2\pi.\end{aligned}$$

Minimizing analytically for σ^2 yields

$$\begin{aligned}\hat{\sigma}^2(\theta^*) &= l_2(\theta^*)/n, \\ \hat{\sigma}^2_R(\theta^*) &= l_2(\theta^*)/(n - p).\end{aligned}$$

Back substitution produces the profile likelihoods

$$-2l^*(\theta^*|y) = l_1(\theta^*) + n\log l_2(\theta^*) + n + n\log(2\pi/n),$$
$$-2l_R^*(\theta^*|y) = l_1(\theta^*) + (n-p)\log l_2(\theta^*) + l_3(\theta^*)$$
$$+ (n-p) + (n-p)\log[2\pi/(n-p)].$$

The only substantial difference between these and the previous expressions is that l_2 has been replaced by a multiple of $\log l_2$. The minimization thus proceeds along the same lines as when not profiling the variance.

2.3. Likelihood calculations. Direct construction of V^{-1} in the expressions for the likelihoods can be computationally prohibitive when the number of data points n is large. As a method for reducing the order of the calculations, we employ an extension of the W-transformation [7], [5]. Begin by constructing the following cross-products matrix:

$$W_0 = \begin{bmatrix} X'R^{-1}X & X'R^{-1}Z & X'R^{-1}y \\ Z'R^{-1}X & Z'R^{-1}Z & Z'R^{-1}y \\ y'R^{-1}X & y'R^{-1}Z & y'R^{-1}y \end{bmatrix}.$$

Assuming R is block-diagonal, this construction is best carried out by first premultiplying the corresponding blocks of X, Z, and y by the inverse of the Cholesky root of R, which we denote by $R^{-1/2}$. The Cholesky root is taken blockwise, thereby keeping the dimensions small. Regarding singularity tolerances for the Cholesky root, we recommend using the original diagonal elements of R times $10^4\zeta$, where ζ is a machine-dependent constant defined as the largest floating-point number for which $\zeta + 1 = \zeta$; typically $\zeta = 10^{-16}$. Twice the sum of the logs of the positive diagonal elements of $R^{-1/2}$ equals $-\log|R|$, and should be computed for use below.

The next step involves the sweep operator, which is closely related to Gauss–Jordan elimination and the Forward Doolittle procedure [4]. Sweeping the first partition of a symmetric, nonnegative definite matrix of the form

$$\begin{bmatrix} A & B \\ B' & C \end{bmatrix}$$

produces

$$\begin{bmatrix} A^- & A^-B \\ -B'A^- & C - B'A^-B \end{bmatrix},$$

where $^-$ denotes a g_2-generalized inverse. If A is nonsingular $A^- = A^{-1}$.

To compute likelihoods, sweep the first partition of the following augmentation of W_0:

$$\begin{bmatrix} I + L'Z'R^{-1}ZL & L'W_0(Z,\cdot) \\ W_0(\cdot,Z)L & W_0 \end{bmatrix}.$$

Here L is the lower-triangular Cholesky root of G so that $G = LL'$. This use of L accomodates the case when G is singular [8]. The augmented matrix does not actually have

to be stored in memory when G is diagonal [5]. In this case the sweep can be performed using only a vector of pivots and the elements in W_0. When G is nondiagonal, the lower triangle of the augmented matrix needs to be stored in memory. As with the Cholesky decomposition, we recommend using the original diagonal elements times $10^4\zeta$ as the singularity tolerance for the sweep.

Summing the logs of the positive pivots during the sweep produces

$$\begin{aligned}\log|I + L'Z'R^{-1}ZL| &= \log|I + ZLL'Z'R^{-1}| \\ &= \log|R + ZGZ'| + \log|R^{-1}| \\ &= \log|V| - \log|R|.\end{aligned}$$

Adding this expression to $\log|R|$ computed above yields l_1.

The sweep results in the following matrix:

$$\begin{bmatrix} X'V^{-1}X & X'V^{-1}Z & X'V^{-1}y \\ Z'V^{-1}X & Z'V^{-1}Z & Z'V^{-1}y \\ y'V^{-1}X & y'V^{-1}Z & y'V^{-1}y \end{bmatrix}.$$

Then sweeping the submatrix

$$\begin{bmatrix} X'V^{-1}X & X'V^{-1}y \\ y'V^{-1}X & y'V^{-1}y \end{bmatrix}$$

on all but its last row produces

$$\begin{bmatrix} (X'V^{-1}X)^- & b \\ b' & l_2 \end{bmatrix}.$$

The sum of the log of the pivots from this sweep is l_3, and $(X'V^{-1}X)^-$ is the approximate variance matrix of b.

One further reduction in dimensionality can be accomplished when ZGZ' and R are both block-diagonal, and the blocking form of R is the same or nested within that of ZGZ'. We call such blocks of ZGZ' *subjects*. In this case the data from different subjects are statistically independent, and the entire procedure above can be performed subjectwise on the data, successively accumulating l_1, l_2, and l_3.

3. Derivatives. The maximum likelihood or REML objective functions can be optimized numerically using a Newton–Raphson algorithm, which is preferred to the Expectation-Maximization (EM) algorithm because of its convergence properties and information matrices [13]. One proviso for Newton–Raphson is that the covariance structures of G and R be twice-differentiable. Possible structures for G and R in this class include diagonal, compound symmetry, unstructured, autoregressive, Toeplitz, and several spatial types [11], [24]. A Kalman filter approach can possibly be used to obtain second derivatives of more complicated time-series structures such as autoregressive-moving average and autoregressive conditionally heteroskedastic. Other heteroskedastic structures involving β as a part of the variance function can be difficult to differentiate, although pseudo-likelihoods can be constructed in this case [1].

Assuming the likelihoods can be differentiated, define the gradient vector and Hessian matrix as follows:

$$g_k = \frac{\partial}{\partial\theta} l_k(\theta),$$

$$H_k = \frac{\partial^2}{\partial\theta\partial\theta'} l_k(\theta)$$

for $k = 1, 2, 3$. We subsequently drop the functional reference to θ, and use subscripting outside brackets to denote elements and dots to denote differentiation.

3.1. General form of derivatives. The following expressions are similar to those given in [11], [13]:

$$\begin{aligned}
[g_1]_r &= \operatorname{tr}\left(V^{-1}\dot{V}_r\right), \\
[g_2]_r &= -r'V^{-1}\dot{V}_rV^{-1}r, \\
[g_3]_r &= -\operatorname{tr}\left(X^{*\prime}V^{-1}\dot{V}_rV^{-1}X^*\right)
\end{aligned}$$

for $r = 1, \ldots, q$, and recall q is the number of elements in θ. Here $X^* = XC$ for a matrix C satisfying $CC' = (X'V^{-1}X)^-$. We have

$$\begin{aligned}
[H_1]_{rs} &= -\operatorname{tr}\left(V^{-1}\dot{V}_rV^{-1}\dot{V}_s\right) + \operatorname{tr}\left(V^{-1}\ddot{V}_{rs}\right), \\
[H_2]_{rs} &= 2r'V^{-1}\dot{V}_rV^{-1}\dot{V}_sV^{-1}r \\
&\quad -2r'V^{-1}\dot{V}_rV^{-1}X^*X^{*\prime}V^{-1}\dot{V}_sV^{-1}r \\
&\quad -r'V^{-1}\ddot{V}_{rs}V^{-1}r, \\
[H_3]_{rs} &= 2\operatorname{tr}\left(X^{*\prime}V^{-1}\dot{V}_rV^{-1}\dot{V}_sV^{-1}X^*\right) \\
&\quad -\operatorname{tr}\left(X^{*\prime}V^{-1}\dot{V}_rV^{-1}X^*X^{*\prime}V^{-1}\dot{V}_sV^{-1}X^*\right) \\
&\quad -\operatorname{tr}\left(X^{*\prime}V^{-1}\ddot{V}_{rs}V^{-1}X^*\right)
\end{aligned}$$

for $r, s = 1, \ldots, q$. The second term in $[H_2]_{rs}$ is not found in the above references; it results from profiling on β. Note that the final term in each of the H_k expressions vanishes if $\ddot{V}_{rs} = 0$, which is usually the case, except for most time-series and spatial structures.

The components g_3, H_2, and H_3 can be written more concisely if we define

$$\begin{aligned}
H_2^r &= X^{*\prime}V^{-1}\dot{V}_rV^{-1}r, \\
H_2^{r,s} &= 2r'V^{-1}\dot{V}_rV^{-1}\dot{V}_sV^{-1}r - r'V^{-1}\ddot{V}_{rs}V^{-1}r, \\
H_3^r &= X^{*\prime}V^{-1}\dot{V}_rV^{-1}X^*, \\
H_3^{r,s} &= 2X^{*\prime}V^{-1}\dot{V}_rV^{-1}\dot{V}_sV^{-1}X^* - X^{*\prime}V^{-1}\ddot{V}_{rs}V^{-1}X^*.
\end{aligned}$$

With this notation,

$$\begin{aligned}
[g_3]_r &= -\operatorname{tr}(H_3^r), \\
[H_2]_{rs} &= H_2^{r,s} - 2H_2^{r\prime}H_2^s, \\
[H_3]_{rs} &= \operatorname{tr}(H_3^{r,s} - H_3^rH_3^s).
\end{aligned}$$

3.2. G **derivatives.** In this section we consider an efficient approach to computing derivatives with respect to those elements of θ in G. As with the likelihood calculations, the method is an extension of the W-transformation [7], [4] and avoids direct construction of V^{-1}. All of the computations are performed using the partitions of the following matrix:

$$W = \begin{bmatrix} W(X,X) & W(X,Z) & W(X,r) \\ W(Z,X) & W(Z,Z) & W(Z,r) \\ W(r,X) & W(r,Z) & W(r,r) \end{bmatrix}$$

$$= \begin{bmatrix} X^{*\prime}V^{-1}X^{*} & X^{*\prime}V^{-1}Z & X^{*\prime}V^{-1}r \\ Z'V^{-1}X^{*} & Z'V^{-1}Z & Z'V^{-1}r \\ r'V^{-1}X^{*} & r'V^{-1}Z & r'V^{-1}r \end{bmatrix}.$$

W can be computed by changing y to r and X to X^* in the matrix obtained in the likelihood calculations.

Note that if r corresponds to an element of G, then $\dot{V}_r = Z\dot{G}_r Z'$. Substituting this into the general formulas and performing cyclic permutations inside of the trace operator yield convenient forms for the derivatives. Intuitively, this algorithm turns the inversion problem "inside out." The following are the results:

$$\begin{aligned} [g_1]_r &= \operatorname{tr}\left(W(Z,Z)\dot{G}_r\right), \\ [g_2]_r &= -W(Z,r)'\dot{G}_r W(Z,r), \\ [g_3]_r &= -\operatorname{tr}\left(W(X,Z)\dot{G}_r W(Z,X)\right), \end{aligned}$$

$$\begin{aligned} [H_1]_{rs} &= -\operatorname{tr}\left(W(Z,Z)\dot{G}_r W(Z,Z)\dot{G}_s\right) \\ &\quad +\operatorname{tr}\left(W(Z,Z)\ddot{G}_{rs}\right), \\ [H_2]_{rs} &= 2W(r,Z)\dot{G}_r W(Z,Z)\dot{G}_s W(Z,r) \\ &\quad -2W(r,Z)\dot{G}_r W(Z,X)W(X,Z)\dot{G}_s W(Z,r) \\ &\quad -W(r,Z)\ddot{G}_{rs}W(Z,r), \\ [H_3]_{rs} &= 2\operatorname{tr}\left(W(X,Z)\dot{G}_r W(Z,Z)\dot{G}_s W(Z,X)\right) \\ &\quad -\operatorname{tr}\left(W(X,Z)\dot{G}_r W(Z,X)W(X,Z)\dot{G}_s W(Z,X)\right) \\ &\quad -\operatorname{tr}\left(W(X,Z)\ddot{G}_{rs}W(Z,X)\right). \end{aligned}$$

The simplifying components are

$$\begin{aligned} H_2^r &= W(X,Z)\dot{G}_r W(Z,r), \\ H_2^{r,s} &= 2W(r,Z)\dot{G}_r W(Z,Z)\dot{G}_s W(Z,r) - W(r,Z)\ddot{G}_{rs}W(Z,r), \\ H_3^r &= W(X,Z)\dot{G}_r W(Z,X), \\ H_3^{r,s} &= 2W(X,Z)\dot{G}_r W(Z,Z)\dot{G}_s W(Z,X) - W(X,Z)\ddot{G}_{rs}W(Z,X). \end{aligned}$$

As in the likelihood calculations, the dimensionality of the above calculations can be reduced when ZGZ' is block-diagonal and its blocks contain those of R. Then the entire

procedure above can be performed blockwise, successively accumulating the derivatives on a subject-by-subject basis.

3.3. R **derivatives.** For derivatives with respect to those elements of θ in R, we make heavy use of the identity

$$V^{-1} = R^{-1} - R^{-1}ZMZ'R^{-1},$$

where

$$M = (G^{-1} + Z'R^{-1}Z)^{-1}.$$

For a derivation, see [9] or [17, §10.8].

When R is block-diagonal, this expression shows that operations involving V^{-1} can be computed without inverting large matrices. Our strategy is to substitute this expression for V^{-1} into the general formulas and then to cyclically permute matrices inside the trace operator. We also use the fact that if r corresponds to an element of R, then $\dot{V}_r = \dot{R}_r$.

The resulting expansions are lengthy, but many of the terms equal 0 for some important special cases. For example, if $M = 0$, as is the case when there are no random effects or when computing MIVQUE(0) estimates, then only one term in each of the expansions is needed. Also, several of the terms drop out if $\ddot{R}_{rs} = 0$. Finally, none of the equations are needed if R is equal to $\sigma^2 I$. The following are the results:

$$\begin{aligned}
[g_1]_r &= \operatorname{tr}\left(R^{-1}\dot{R}_r\right) \\
&\quad -\operatorname{tr}\left(MZ'R^{-1}\dot{R}_rR^{-1}Z\right), \\
[g_2]_r &= -r'R^{-1}\dot{R}_rR^{-1}r \\
&\quad +2r'R^{-1}ZMZ'R^{-1}\dot{R}_rR^{-1}r \\
&\quad -r'R^{-1}ZMZ'R^{-1}\dot{R}_rR^{-1}ZMZ'R^{-1}r, \\
[g_3]_r &= -\operatorname{tr}\left(H_3^r\right),
\end{aligned}$$

$$\begin{aligned}
[H_1]_{rs} &= -\operatorname{tr}\left(R^{-1}\dot{R}_rR^{-1}\dot{R}_s\right) \\
&\quad +\operatorname{tr}\left(MZ'R^{-1}\dot{R}_rR^{-1}\dot{R}_sR^{-1}Z\right) \\
&\quad +\operatorname{tr}\left(MZ'R^{-1}\dot{R}_sR^{-1}\dot{R}_rR^{-1}Z\right) \\
&\quad -\operatorname{tr}\left(MZ'R^{-1}\dot{R}_rR^{-1}ZMZ'R^{-1}\dot{R}_sR^{-1}Z\right) \\
&\quad +\operatorname{tr}\left(R^{-1}\ddot{R}_{rs}\right) \\
&\quad -\operatorname{tr}\left(MZ'R^{-1}\ddot{R}_{rs}R^{-1}Z\right), \\
[H_2]_{rs} &= H_2^{r,s} - 2H_2^{r\prime}H_2^s, \\
[H_3]_{rs} &= \operatorname{tr}\left(H_3^{r,s} - H_3^rH_3^s\right),
\end{aligned}$$

where

$$\begin{aligned}
H_2^r &= X^{*\prime}R^{-1}\dot{R}_rR^{-1}r \\
&\quad -X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_rR^{-1}r \\
&\quad -X^{*\prime}R^{-1}\dot{R}_rR^{-1}ZMZ'R^{-1}r \\
&\quad +X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_rR^{-1}ZMZ'R^{-1}r,
\end{aligned}$$

$$
\begin{aligned}
H_2^{r,s} = {} & 2r'R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}r \\
& -2r'R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}r \\
& -2r'R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}r \\
& +2r'R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}r \\
& -2r'R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}r \\
& +2r'R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}r \\
& +2r'R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}r \\
& -2r'R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}r \\
& -r'R^{-1}\ddot{R}_{rs}R^{-1}r \\
& +2r'R^{-1}\ddot{R}_{rs}R^{-1}ZMZ'R^{-1}r \\
& -r'R^{-1}ZMZ'R^{-1}\ddot{R}_{rs}R^{-1}ZMZ'R^{-1}r,
\end{aligned}
$$

$$
\begin{aligned}
H_3^r = {} & X^{*\prime}R^{-1}\dot{R}_r R^{-1}X^* \\
& -X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}X^* \\
& -X^{*\prime}R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}X^* \\
& +X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}X^*,
\end{aligned}
$$

$$
\begin{aligned}
H_3^{r,s} = {} & 2X^{*\prime}R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}X^* \\
& -2X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}X^* \\
& -2X^{*\prime}R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}X^* \\
& +2X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}X^* \\
& -2X^{*\prime}R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}X^* \\
& +2X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}X^* \\
& +2X^{*\prime}R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}X^* \\
& -2X^{*\prime}R^{-1}ZMZ'R^{-1}\dot{R}_r R^{-1}ZMZ'R^{-1}\dot{R}_s R^{-1}ZMZ'R^{-1}X^* \\
& -X^{*\prime}R^{-1}\ddot{R}_{rs}R^{-1}X^* \\
& +2X^{*\prime}R^{-1}\ddot{R}_{rs}R^{-1}ZMZ'R^{-1}X^* \\
& -X^{*\prime}R^{-1}ZMZ'R^{-1}\ddot{R}_{rs}R^{-1}ZMZ'R^{-1}X^*.
\end{aligned}
$$

Each of the terms above can be computed piecewise if R is block-diagonal. This is carried out by expanding the $\dot{R}_r$ factor into a sum of the blocks. The individual blocks can be computed by looping through the random effects, while the other factors in the term need to be computed previously. The use of $R^{-1/2}$ throughout is recommended.

3.4. Cross derivatives. Here we derive expressions for $[H_k]_{rs}$ when r corresponds to a parameter from R and s corresponds to a parameter from G. This effectively means repeating some of the formulas from the previous section with $\dot{R}_s$ replaced by $Z\dot{G}_s Z'$. We assume that G and R share none of the same parameters, so all of the 2-dot terms are zero. Also note that H_2^r, H_2^s, H_3^r, and H_3^s will already be computed by one of the previous methods.

$$
\begin{aligned}
[H_1]_{rs} &= -\operatorname{tr}\left(Z'R^{-1}\dot{R}_r R^{-1} Z\dot{G}_s\right) \\
&\quad +2\operatorname{tr}\left(Z'R^{-1}\dot{R}_r R^{-1} Z\dot{G}_s Z'R^{-1} ZM\right) \\
&\quad -\operatorname{tr}\left(Z'R^{-1}\dot{R}_r R^{-1} ZMZ'R^{-1} Z\dot{G}_s Z'R^{-1} ZM\right), \\
[H_2]_{rs} &= H_2^{r,s} - 2H_2^{r\prime} H_2^s, \\
[H_3]_{rs} &= \operatorname{tr}\left(H_3^{r,s} - H_3^r H_3^s\right),
\end{aligned}
$$

where

$$
\begin{aligned}
H_2^{r,s} &= 2A_1^r B\dot{G}_s B' Z'R^{-1} r, \\
H_3^{r,s} &= 2A_2^r B\dot{G}_s B' Z'R^{-1} X^*
\end{aligned}
$$

and

$$
\begin{aligned}
A_1^r &= r'R^{-1}\dot{R}_r R^{-1} Z \\
&\quad -r'R^{-1} ZMZ'R^{-1}\dot{R}_r R^{-1} Z, \\
A_2^r &= X^{*\prime} R^{-1}\dot{R}_r R^{-1} Z, \\
&\quad -X^{*\prime} R^{-1} ZMZ'R^{-1}\dot{R}_r R^{-1} Z \\
B &= MZ'R^{-1} Z - I.
\end{aligned}
$$

The matrix $W_0(Z, \cdot)$ obtained in the likelihood calculations can be used in computing the above expressions.

3.5. Derivatives when factoring or profiling the residual variance. We now describe derivatives appropriate for the objective functions described in §2.2. Two different forms are presented: the first uses the factored objective functions and considers σ^2 as a parameter to be differentiated; the second uses the profiled objective functions and eliminates σ^2 from the optimization problem. The second method is more efficient for optimization purposes because it has one less parameter than the first. For the first case, the formulas are as follows:

$$
\frac{\partial}{\partial\theta^*}[-2l(\theta^*, \sigma^2)] = \begin{bmatrix} g_1 + g_2/\sigma^2, \\ n/\sigma^2 - l_2/\sigma^4 \end{bmatrix},
$$

$$
\frac{\partial}{\partial\theta^*}[-2l_R(\theta^*, \sigma^2)] = \begin{bmatrix} g_1 + g_2/\sigma^2 + g_3 \\ (n-p)/\sigma^2 - l_2/\sigma^4 \end{bmatrix},
$$

$$
\frac{\partial^2}{\partial(\theta^*, \sigma^2)\partial(\theta^*, \sigma^2)}[-2l(\theta^*, \sigma^2)] = \begin{bmatrix} H_1 + H_2/\sigma^2 & -g_2/\sigma^4 \\ -g_2'/\sigma^4 & -n/\sigma^4 + 2l_2/\sigma^6 \end{bmatrix},
$$

$$
\frac{\partial^2}{\partial(\theta^*, \sigma^2)\partial(\theta^*, \sigma^2)}[-2l_R(\theta^*, \sigma^2)] = \begin{bmatrix} H_1 + H_2/\sigma^2 + H_3 & -g_2/\sigma^4 \\ -g_2'/\sigma^4 & -(n-p)/\sigma^4 + 2l_2/\sigma^6 \end{bmatrix}.
$$

For the second case, let g_2^* and H_2^* denote g_2 and H_2 divided by the appropriate $\hat{\sigma}^2$. The profiling formulas are then as follows:

$$\frac{\partial}{\partial \theta^*}[-2l^*(\theta^*)] = g_1 + g_2^*,$$

$$\frac{\partial}{\partial \theta^*}[-2l_R^*(\theta^*)] = g_1 + g_2^* + g_3,$$

$$\frac{\partial^2}{\partial(\theta^*, \sigma^2)\partial(\theta^*, \sigma^2)}[-2l^*(\theta^*, \sigma^2)] = H_1 + H_2^* - g_2^* g_2^{*\prime}/n,$$

$$\frac{\partial^2}{\partial(\theta^*, \sigma^2)\partial(\theta^*, \sigma^2)}[-2l_R^*(\theta^*, \sigma^2)] = H_1 + H_2^* - g_2^* g_2^{*\prime}/(n-p) + H_3.$$

These profiling formulas result from

$$\frac{\partial}{\partial \theta} \log l_2 = \frac{\frac{\partial}{\partial \theta} l_2}{l_2},$$

$$\frac{\partial^2}{\partial \theta \partial \theta'} \log l_2 = \frac{\frac{\partial^2}{\partial \theta \partial \theta'} l_2}{l_2} - \left[\frac{\partial}{\partial \theta} \log l_2\right]\left[\frac{\partial}{\partial \theta} \log l_2\right]'.$$

3.6. Overall and scoring derivatives. So far we have seen how to calculate $g_k = \partial l_k/(\partial\theta)$ and $H_k = \partial^2 l_k/(\partial\theta\partial\theta')$, $k = 1, 2, 3$. In this section we show how to put these together to form overall first and second derivatives. Once formed, the Newton–Raphson step is obtained by computing the inverse of the second derivative matrix, the Hessian, times the first derivative vector, the gradient.

Tables 1 and 2 present the overall derivatives for maximum likelihood and REML estimation, respectively. The maximum likelihood formulas are profiled with respect to the fixed effects. The rows labeled "Factoring" and "Profiling" correspond to the results from the previous section concerning the residual variance parameter, σ^2. For the factoring and profiling formulas, all of the derivatives are evaluated at θ^*. Also, g_2^* and H_2^* denote g_2 and H_2 divided by the appropriate $\hat{\sigma}^2$.

TABLE 1
Overall derivatives for maximum likelihood estimation.

Technique	Gradient	Hessian
Newton	$g_1 + g_2$	$H_1 + H_2$
Scoring	$g_1 + g_2$	$-H_1$
Factoring-Newton	$\begin{bmatrix} g_1 + g_2/\sigma^2 \\ n/\sigma^2 - l_2/\sigma^4 \end{bmatrix}$	$\begin{bmatrix} H_1 + H_2/\sigma^2 & -g_2/\sigma^4 \\ -g_2'/\sigma^4 & -n/\sigma^4 + 2l_2/\sigma^6 \end{bmatrix}$
Factoring-Scoring	$\begin{bmatrix} g_1 + g_2/\sigma^2 \\ n/\sigma^2 - l_2/\sigma^4 \end{bmatrix}$	$\begin{bmatrix} -H_1 & g_1/\sigma^2 \\ g_1'/\sigma^2 & n/\sigma^4 \end{bmatrix}$
Profiling-Newton	$g_1 + g_2^*$	$H_1 + H_2^* - g_2^* g_2^{*\prime}/n$
Profiling-Scoring	$g_1 + g_2^*$	$-H_1 - g_2^* g_2^{*\prime}/n$

Also in Tables 1 and 2 are formulas for a modification of the Newton–Raphson algorithm known as Fisher scoring. A scoring algorithm works by replacing the second derivative matrix

TABLE 2
Overall derivatives for REML estimation.

Technique	Gradient	Hessian
Newton	$g_1 + g_2 + g_3$	$H_1 + H_2 + H_3$
Scoring	$g_1 + g_2 + g_3$	$-H_1 + H_3$
Factoring-Newton	$\begin{bmatrix} g_1 + g_2/\sigma^2 + g_3 \\ (n-p)/\sigma^2 - l_2/\sigma^4 \end{bmatrix}$	$\begin{bmatrix} H_1 + H_2/\sigma^2 + H_3 & -g_2/\sigma^4 \\ -g_2'/\sigma^4 & -(n-p)/\sigma^4 + 2l_2/\sigma^6 \end{bmatrix}$
Factoring-scoring	$\begin{bmatrix} g_1 + g_2/\sigma^2/ + g_3 \\ (n-p)/\sigma^2 - l_2/\sigma^4 \end{bmatrix}$	$\begin{bmatrix} -H_1 + H_3 & g_1/\sigma^2 \\ g_1'/\sigma^2 & (n+p)/\sigma^4 \end{bmatrix}$
Profiling-Newton	$g_1 + g_2^* + g_3$	$H_1 + H_2^* + H_3 - g_2^* g_2^{*\prime}/(n-p)$
Profiling-scoring	$g_1 + g_2^* + g_3$	$-H_1 + H_3 - g_2^* g_2^{*\prime}/(n-p)$

with its expectation at the true values of the parameters. This entails setting $b(\theta)$ equal to β before taking expectations, thereby changing r to $Z\upsilon + e$. The only random components are those with a subscript of 2, and the resulting expressions are as follows:

$$E_{b=\beta}[l_2(\theta)] = n,$$

$$E_{b=\beta}[l_2(\theta^*)] = \sigma^2 n,$$

$$E_{b=\beta}[g_2(\theta)] = -g_1(\theta),$$

$$E_{b=\beta}[g_2(\theta^*)] = -\sigma^2 g_1(\theta^*),$$

$$E_{b=\beta}[H_2(\theta)] = -2H_1(\theta) + \operatorname{tr}\left(V^{-1}\ddot{V}\right),$$

$$E_{b=\beta}[H_2(\theta^*)] = -2\sigma^2 H_1(\theta^*) + \sigma^2 \operatorname{tr}\left(V^{-1}\ddot{V}\right).$$

The scoring formulas in Tables 1 and 2 omit the 2-dot terms above and are less computationally demanding than the full Newton–Raphson approach. Also, the profiling-scoring formulas are only a partial implementation, as it is cumbersome to take the expectation of $g_2^* g_2^{*\prime}$. Another partial scoring implementation involves using

$$Err' = V - X^* X^{*\prime}$$

directly, although this technique has nearly the same computational order as full Newton–Raphson.

One may wish to transform the Hessian for the (θ^*, σ^2) parameterization into that for the θ parameterization. This is accomplished by

$$H(\theta) = BH(\theta^*, \sigma^2)B' + A.$$

For example, if $\theta^* = (\theta_1^*, \theta_2^*)$

$$A = \begin{bmatrix} 0 & 0 & -g(\theta_1^*)/\sigma^4 \\ 0 & 0 & -g(\theta_2^*)/\sigma^4 \\ -g(\theta_1^*)/\sigma^4 & -g(\theta_2^*)/\sigma^4 & 2/\sigma^4[\theta_1^* g(\theta_1^*) + \theta_2^* g(\theta_2^*)] \end{bmatrix},$$

$$B = \begin{bmatrix} 1/\sigma^2 & 0 & 0 \\ 0 & 1/\sigma^2 & 0 \\ -\theta_1^*/\sigma^2 & -\theta_2^*/\sigma^2 & 1 \end{bmatrix}.$$

Note that A vanishes both at the optimum and when scoring.

4. Initial estimates. We recommend the MIVQUE(0) method [14] for computing initial estimates. One advantage of this approach is that in certain balanced cases it produces the REML estimates, thus eliminating the need for Newton–Raphson iterations. Disadvantages include the fact that the resulting V may not be positive definite and that MIVQUE(0) may not be appropriate for many time-series and spatial structures. In these cases we suggest setting V to some easily obtained and sensible value, or using user-supplied initial values. A brief explanation of MIVQUE(0) estimates now follows, and then their computing formulas are given.

As before, let y be data with variance matrix V, with V containing q unknown parameters denoted by the vector θ. An assumption for the MIVQUE(0) approach is that we can write

$$V = \sum_{r=1}^{q} \theta_r \dot{V}_r.$$

This essentially requires V to have a general linear structure with $\ddot{V}_{rs} = 0$, an assumption not satisfied for many time-series and spatial structures. Define

$$P = V^{-1} - V^{-1}X(X'V^{-1}X)^{-1}X'V^{-1}.$$

The MIVQUE(0) estimates of θ solve the equations

$$\left[\operatorname{tr}\left(P\dot{V}_r P\dot{V}_s\right)\right]_{r,s}\theta = \left[y'P\dot{V}_r Py\right]_r,$$

where P is computed at $V = I$. It is a special case of the MIVQUE(V_0) method that consists of setting the quadratic forms $y'P\dot{V}_r Py$ equal to their expected values given $V = V_0$ [18]. By examining §3.1, these equations are equivalent to

$$-(H_1 + H_3)\theta = -g_2.$$

We solve this system for θ when using REML.

For maximum likelihood, we solve the modified MIVQUE(0) equations

$$\left[\operatorname{tr}\left(V^{-1}\dot{V}_r V^{-1}\dot{V}_s\right)\right]_{r,s}\theta = \left[y'P\dot{V}_r Py\right]_r,$$

which consist of setting the quadratic forms $y'P\dot{V}_r Py$ equal to the expected values of $r'V^{-1}\dot{V}_r V^{-1}r$ given $V = I$. These equations reduce to

$$-H_1\theta = -g_2.$$

When profiling σ^2 out of V, the following formulas are used. For REML,

$$\begin{bmatrix} \left[\operatorname{tr}\left(P\dot{V}_r P\dot{V}_s\right)\right]_{r,s=1}^{q-1} & \left[\operatorname{tr}\left(P\dot{V}_r P\right)\right]_{r=1}^{q-1} \\ \left(\left[\operatorname{tr}\left(P\dot{V}_s P\right)\right]_{s=1}^{q-1}\right)' & \operatorname{tr}\left(P^2\right) \end{bmatrix} \begin{bmatrix} \sigma^2\theta^* \\ \sigma^2 \end{bmatrix} = \begin{bmatrix} \left[y'P\dot{V}_r Py\right]_{r=1}^{q-1} \\ y'P^2y \end{bmatrix},$$

which, under the idempotency of P, becomes

$$\begin{bmatrix} -(H_1 + H_3) & g_1 + g_3 \\ g_1' + g_3' & n - p \end{bmatrix} \begin{bmatrix} \sigma^2\theta^* \\ \sigma^2 \end{bmatrix} = \begin{bmatrix} -g_2 \\ l_2 \end{bmatrix}.$$

For maximum likelihood, we have

$$\begin{bmatrix} \left[\operatorname{tr}\left(V^{-1}\dot{V}_r V^{-1}\dot{V}_s\right)\right]_{r,s=1}^{q-1} & \left[\operatorname{tr}\left(V^{-1}\dot{V}_r V^{-1}\right)\right]_{r=1}^{q-1} \\ \left(\left[\operatorname{tr}\left(V^{-1}\dot{V}_s V^{-1}\right)\right]_{s=1}^{q-1}\right)' & \operatorname{tr}\left(V^{-2}\right) \end{bmatrix} \begin{bmatrix} \sigma^2\theta^* \\ \sigma^2 \end{bmatrix}$$

$$= \begin{bmatrix} \left[y'P\dot{V}_r Py\right]_{r=1}^{q-1} \\ y'Py \end{bmatrix},$$

which, when $V = I$, becomes

$$\begin{bmatrix} -H_1 & g_1 \\ g_1' & n \end{bmatrix} \begin{bmatrix} \sigma^2\theta^* \\ \sigma^2 \end{bmatrix} = \begin{bmatrix} -g_2 \\ l_2 \end{bmatrix}.$$

5. Computational order. Using the notation from Table 3, the following are estimates of the computational speed of the algorithms described above. For likelihood calculations, the cross- products matrix construction is of order $n(p+g)^2$ and the sweep operations are of order $(p+g)^3$. The first derivative calculations for parameters in H are of order qg^3 for maximum likelihood and $q(g^3 + pg^2 + p^2g)$ for REML. If ZGZ' is block-diagonal with blocks corresponding to subjects, then replace g by g/s and q by qs in these calculations. The first derivative calculations for parameters in R are of order $qs(t^3 + gt^2 + g^2t)$ for maximum likelihood and $qs(t^3 + (p+g)t^2 + (p^2+g^2)t)$ for REML. For the second derivatives, replace q by $q(q+1)/2$ in the first derivative expressions. When there are both G and R parameters, then additional calculations are required of order equal to the sum of the orders for G and R; this is the most computationally intensive case. The approximate memory requirement in bytes is $40(p^2 + g^2) + 32(p+g)^2$.

TABLE 3
Notation for order expressions.

Symbol	Number
p	columns of X
g	columns of Z
n	observations
q	covariance parameters
t	maximum observations per subject
s	subjects

Note that the second derivatives cost approximately $q/2$ times as much as the first derivatives, which in turn cost approximately q times as much as likelihood evaluations. This leads one to suspect that first-derivative (quasi-Newton) or derivative-free methods may outperform standard Newton–Raphson for large problems. Sparse matrix techniques are also a possibility, and progress is being made primarily by animal breeders, who often have millions of records [21].

To provide a brief illustration of the efficiency of the algorithms described in this paper, Tables 4 and 5 describe several small examples and their run times on different computing systems. All calculations are performed with Release 6.08.01 of the SAS/STAT MIXED procedure [16], and Table 5 displays time to completion of PROC MIXED with its default options. Approximately 5–10% of the time is spent in initial data set-up and post-convergence calculations, while the remainder is consumed by a ridge-stabilized Newton–Raphson algorithm with REML MIVQUE(0) starting values (§4). Convergence is assumed when the relative orthogonality criterion [13] is less than 10^{-8}.

Table 4 lists the dimensions of each example using the notation from Table 3. In addition, the number of covariance parameters, q, is divided into those corresponding to G (q_G) and R (q_R), indicating the appropriate formulas from §3. Table 4 also includes the number of likelihood evaluations and Newton-Raphson iterations required for convergence.

Examples 1, 3, 6, and 7 are all simple random effects models with diagonal G and R. Examples 6 and 7 employ the same data and model; however, Example 6 breaks Z into 30 subjects during the computations as described at the end of §2. In contrast, Example 7 operates on the entire 630 columns of Z at once. Examples 2 and 5 have no G matrix; Example 2 has a large, unstructured-block-diagonal R matrix, and Example 5 has a 306×306 dense R matrix. Finally, Example 4 has a diagonal G matrix and a Toeplitz-block-diagonal R matrix.

Table 5 compares the computing speed of these examples for several common hardware configurations. The times increase roughly as we move down and left on the table, and all of them are less than an hour. As expected, the PCs ran the slowest, with OS/2 outperforming Windows by a factor ranging from 2 to 6. The HP Unix workstation ran approximately twice as fast as OS/2 and 1.5 times slower than a standard IBM mainframe system. The addition of a vector facility on the IBM mainframe produces savings ranging from 5-30% over the standard configuration. The Convex system was the fastest of the six.

One anomaly in Table 5 is the slowness of the mainframes for Example 6 as compared with Example 7. Because of its block-diagonal nature, we would expect Example 6 to run faster than Example 7, as it did on most of the other systems. Further investigation showed that the elapsed times of the mainframes for Example 6 were more than 3 times the CPU times, whereas in the other six examples they were less than 1.5 times the CPU times. The extra time resulted from I/O inefficiency, and a re-run under virtual I/0 produced an Example 6 time of 02:18 for the 3090 with the vector facility.

TABLE 4
Dimensions of run-time examples, using notation from Table 3.

Example	p	g	n	q	q_G	q_R	t	s	Number of evaluations	Number of iterations
1	3	33	308	7	6	1	31	11	4	2
2	5	0	2002	10	0	10	4	930	4	2
3	108	285	240	8	7	1	240	1	9	3
4	84	21	108	6	2	4	36	3	11	6
5	4	0	306	2	0	2	306	1	4	2
6	231	630	3008	3	2	1	21	30	6	3
7	231	630	3008	3	2	1	3008	1	6	3

For a rough comparison with previously published results, Lindstrom and Bates [13] report a run time of 6.32 seconds for Example 1 on a Vax 11/750 running 4.3 BSC Unix. They utilize Newton–Raphson formulas based on the QR decomposition. The QR method has higher numerical stability than the sweep-based methods described in this paper because

TABLE 5
Elapsed times in minutes:seconds for the examples in Table 4 running on various systems.

Example	486 PC Windows 3.1 16 mb	486 PC OS/2 2.1 16 mb	HP 9000/720 HP-UX 9.01 32 mb	IBM 3090 MVS 32 mb	IBM 3090 MVS-vector 32 mb	Convex 3800 ConvexOS 512 mb
1	00:05	00:03	00:02	00:01	00:01	00:01
2	04:40	01:12	00:42	00:28	00:25	00:33
3	03:23	02:20	00:54	00:48	00:35	00:10
4	09:53	01:51	01:05	00:46	00:44	00:44
5	52:57	07:16	04:07	02:47	02:36	02:41
6	53:49	14:42	06:05	09:26	09:02	01:56
7	52:57	21:11	08:43	05:25	03:42	02:08

it operates on X and not $X'X$. However, the QR method is approximately two times slower than the sweep [3, §5.3.9], and Example 1 provides a confirmation.

In summary, the algorithms appear to be able to handle moderately sized problems quite well. More rigorous investigation is called for to precisely determine their practicality.

Acknowledgments. We thank several referees for helpful comments, Tim Gregoire at VPI for pointing out [9], and Leigh Ihnen and Connie Dunbar at SAS for aid with the run-time examples.

REFERENCES

[1] R. J. CARROLL AND D. RUPPERT, *Transformation and Weighting in Regression*, Chapman and Hall, London, 1988.

[2] A. P. DEMPSTER, M. R. SELWYN, C. M. PATEL, AND A. J. ROTH, *Statistical and computational aspects of mixed model analysis*, Appl. Statist., 33 (1984), pp. 203–214.

[3] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[4] J. H. GOODNIGHT, *A tutorial on the sweep operator*, Amer. Statist., 33 (1979), pp. 149–158.

[5] J. H. GOODNIGHT AND W. J. HEMMERLE, *A simplified algorithm for the W-transformation in variance component estimation*, Technom., 21 (1979), pp. 265–268.

[6] D. A. HARVILLE, *Maximum likelihood approaches to variance component estimation and to related problems*, J. Amer. Statist. Assoc., 72 (1977), pp. 320–338.

[7] W. J. HEMMERLE AND H. O. HARTLEY, *Computing maximum likelihood estimates for the mixed* AOV *model using the W-transformation*, Technom., 15 (1973), pp. 819–831.

[8] C. R. HENDERSON, *Applications of Linear Models in Animal Breeding*, University of Guelph, Canada, 1984.

[9] C. R. HENDERSON, O. KEMPTHORNE, S. R. SEARLE, AND C. M. VON KROSIGK, *The estimation of environmental and genetic traits from records subject to culling*, Biometrics, 15 (1959), pp. 192–218.

[10] R. R. HOCKING, *The Analysis of Linear Models*, Brooks/Cole, Monterey, CA, 1985.

[11] R. I. JENNRICH AND M. D. SCHLUCHTER, *Unbalanced repeated-measures models with structured covariance matrices*, Biometrics, 42 (1986), pp. 805–820.

[12] N. M. LAIRD AND J. H. WARE, *Random-effects models for longitudinal data*, Biometrics, 38 (1982), pp. 963–974.

[13] M. J. LINDSTROM AND D. M. BATES, *Newton-Raphson and* EM *algorithms for linear mixed-effects models for repeated-measures data*, J. Amer. Statist. Assoc., 83 (1988), pp. 1014–1022.

[14] C. R. RAO, *Estimation of variance and covariance components in linear models*, J. Amer. Statist. Assoc., 67 (1972), pp. 112–115.

[15] G. K. ROBINSON, *That* BLUP *is a good thing: the estimation of random effects* (with discussion), Statist. Sci., 6 (1991), pp. 15–51.

[16] SAS INSTITUTE INC., *Chapter 16: The* MIXED *Procedure*, in SAS Technical Report P-229, SAS/STAT Software: Changes and Enhancements, Release 6.07, SAS Institute Inc., Cary, NC, 1992.

[17] S. R. SEARLE, *Linear Models*, John Wiley, New York, 1971.

[18] S. R. SEARLE, *Mixed models and unbalanced data: wherefrom, whereat, and whereto?*, Comm. in Statist. A–Theory Methods, 17 (1988), pp. 935–968.

[19] S. R. SEARLE, G. CASELLA, AND C. E. MCCULLOCH, *Variance Components*, John Wiley, New York, 1992.

[20] W. W. STROUP, *Predictable functions and prediction space in the mixed model procedure*, in Applications of Mixed Models in Agriculture and Related Disciplines, Southern Cooperative Series Bulletin No. 343, Louisiana Agricultural Experiment Station, Baton Rouge, LA, 1989, pp. 39–48.

[21] L. D. VAN VLECK, *The revolution in statistical computing: from least squares to* DFREML, in Proceedings 41st National Breeders Roundtable, Poultry Breeders of America and Southeastern Poultry and Egg Assoc., May 7–8, St. Louis, MO, 1992.

[22] E. F. VONESH AND R. L. CARTER, *Efficient inference for random-coefficient growth curve models with unbalanced data*, Biometrics, 43 (1987), pp. 617–628.

[23] A. ZELLNER, *An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias*, J. Amer. Statist. Assoc., 57 (1962), pp. 348–368.

[24] D. L. ZIMMERMAN AND D. A. HARVILLE, *A random field approach to the analysis of field-plot experiments and other spatial experiments*, Biometrics, 47 (1991), pp. 223–239.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1311–1330, November 1994

004

THIN PLATE SPLINES WITH DISCONTINUITIES AND FAST ALGORITHMS FOR THEIR COMPUTATION*

DAVID LEE† AND JYH-JEN HORNG SHIAU‡

Abstract. This paper discusses the problem of fitting noisy data in high dimensions. Using a reproducing kernel Hilbert space approach, the authors study thin plate splines, which preserve and incorporate discontinuities explicitly. Both smoothing and interpolating splines are computed. To cope with the formidable cost of computation, the authors propose four different fast algorithms, which are an order of magnitude faster than conventional methods. To construct the splines the authors assume that the location and type of discontinuities are known. However, this information is usually not available in practice. A discontinuity detector in high dimensions based on a residual analysis is investigated.

Key words. discontinuity, thin plate spline, smoothing spline, interpolating spline, reproducing kernel Hilbert space, recursive block Toeplitz matrix, fast algorithm, inversion, *LU* decomposition, *QR* decomposition, numerical stability, preconditioning

AMS subject classifications. 26A15, 41A15, 41A29, 41A63, 41A65, 65D07, 65D10, 65F10, 68F30, 68E99

1. Introduction.

The problem of data fitting in high dimensions arises in many applications in science and technology. A number of approaches have been proposed, including thin plate smoothing splines [51], thin plate interpolating splines [33], projection pursuit regression (PPR) [18], generalized additive models [5], sliced inverse regression (SIR) [31], principal Hessian directions (pHd) [30], discretized Laplacian smoothing by Fourier methods [34], multivariate adaptive regression splines (MARS) [17], and interaction splines [50]. Among these methods, spline estimates provide satisfactory solutions for regression problems of recovering unknown functions from noisy data. They are often used as a useful descriptive tool for statistical data analysis. In addition to the pleasing graphical display of fitted splines, which allows analysts to gain more insight from the data, splines provide good estimates of some interesting quantities of unknown functions, such as derivatives and integrals. They also have many interesting theoretical properties, such as optimality in terms of minimizing approximation errors [45]. There are many techniques for constructing splines. An elegant method is to derive splines by a reproducing kernel Hilbert space approach [15], [51], [33]. The problem is then reduced to solving a system of linear equations.

A serious problem encountered in practice is that the underlying structure of the functions to be recovered may have discontinuities. This is typical in applications such as seismology, tomography, image processing and analysis, computer graphics, economic or environmental systems, and meteorology. Splines tend to smooth out the discontinuities and hence do not provide satisfactory results. To cope with such difficulties, different approaches have been proposed, such as splines with multiple knots [14], splines in tension [40], splines with discontinuities [41]–[43], [26], [20], [47], [35], [28], and many other methods. Most of these techniques are for curve fitting in one dimension. However, practical applications often require data fitting in two dimensions or higher. We also observe that discontinuities are not incorporated in the construction of splines in an explicit way in many of these techniques.

A more series problem is the formidable computation cost. For example, to compute a spline in a reproducing kernel Hilbert space, we solve a system of linear equations with a dense coefficient matrix. Given N data points, the computation cost of a direct method such as Cholesky factorization is $O(N^3)$ [33] and is not practical when N is large.

*Received by the editors November 11, 1991; accepted for publication (in revised form) September 20, 1993.

†AT&T Bell Laboratories, Murray Hill, New Jersey 07974-0636 (lee@research.att.com).

‡AT&T Bell Laboratories, Princeton, New Jersey 08540. Current address: Institute of Statistics. National Chiao Tung University, Hsinchu, Taiwan, Republic of China (jyhjen@jenny.nctu.edu.tw).

In this work, we study an extension of smoothing and interpolating thin plate splines in high dimensions such that the spline estimates of the unknown function preserve discontinuities. We use a reproducing kernel Hilbert space approach to construct the splines and incorporate discontinuities in an explicit and quantitative way. The problem is then reduced to solving a system of linear equations. We develop fast algorithms for the computation with a cost $O(N^2 \log N)$. We remark that these algorithms can also be applied to thin plate splines without discontinuities with the same cost.

The remaining sections are organized as follows. We review thin plate splines (without discontinuities) in §2 and discuss discontinuities briefly in §3. In §§4 and 5, respectively, we study smoothing and interpolating splines with discontinuities, assuming that we know where and of which type the discontinuities are. We present four different fast algorithms for computing the splines in §6. Finally, we discuss a residual discontinuity detector in high dimensions in §7.

2. Thin plate splines (without discontinuities). We first review briefly thin plate splines without discontinuities. For details, see [47], [33], [41]. Assume that data $\vec{z} = [z_1, \ldots, z_N]'$ are obtained from the statistical model

$$z_i = f(t_i) + \varepsilon_i, \qquad i = 1, 2, \ldots, N, \tag{2.1}$$

where $t_i = (x_{1i}, \ldots, x_{di}) \in R^d$, R is the set of real numbers, ε_i's are uncorrelated zero mean noise with variance σ^2, and $f : R^d \to R$ is a smooth function. A smoothing spline estimate f_λ^* of f is the minimizer of the variational problem

$$\min_{g \in W_2^m(R^d)} \left\{ \frac{1}{N} \sum_{i=1}^{N} [g(t_i) - z_i]^2 + \lambda J(g) \right\}, \tag{2.2}$$

where

$$J(g) = \sum_{\alpha_1 + \cdots + \alpha_d = m} \frac{m}{\alpha_1! \cdots \alpha_d!} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left(\frac{\partial^m g}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}} \right)^2 dx_1 \cdots dx_d, \tag{2.3}$$

where $W_2^m(R^d)$ is the Sobolev space[1] on R^d, and $\lambda > 0$ is the smoothing parameter controlling the tradeoff between fidelity to data and roughness of the solution. The null space of $J(\cdot)$ in $W_2^m(R^d)$ is $H_0 = P_{m-1}$, which is the span of the $M = \binom{d+m-1}{d}$ monomials of total degree less then m. For example, for $m = 2$, $P_{m-1} = \langle 1, x_1, \ldots, x_d \rangle$. It is necessary that $2m - d > 0$ so that $W_2^m(R^d)$ is a reproducing kernel Hilbert space [33]. Also note that $N >> M$. For more details about the reproducing kernel Hilbert space, see [4].

Let $\{s_1, \ldots, s_M\}$ be a fixed set of M points in R^d such that $\sum_{\nu=1}^{M} a_\nu \phi_\nu(s_\mu) = 0$ for $\mu = 1, \ldots, M$ implies $a_\nu = 0$ for all $\nu = 1, \ldots, M$, where $\{\phi_\nu\}$ is a basis of P_{m-1}. Such points $\{s_\mu\}$ can be chosen from $\{t_i\}$ if and only if the following $N \times M$ matrix is of full rank:

$$T = \begin{pmatrix} \phi_1(t_1) & \cdots & \phi_M(t_1) \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \phi_1(t_N) & \cdots & \phi_M(t_N) \end{pmatrix}. \tag{2.4}$$

[1]Functions on R^d with absolutely continuous $(m-1)$st derivatives and square integrable mth derivatives [1].

This assumption usually holds since $N >> M$. From now on, we assume that T is of full rank. We choose a basis $\{\Psi_\nu\}$ of P_{m-1} such that

$$\Psi_\nu(s_\mu) = \delta_{\mu,\nu} = \begin{cases} 0 & \text{for } \mu \neq \nu, \\ 1 & \text{for } \mu = \nu. \end{cases} \tag{2.5}$$

An inner product in $H_0 = P_{m-1}$ can be defined as

$$\langle f, g\rangle_{H_0} := \sum_{\mu=1}^{M} f(s_\mu) \cdot g(s_\mu) \quad \text{for } f, g \in H_0. \tag{2.6}$$

It can be easily shown that H_0, with respect to the inner product defined in (2.6), is a reproducing kernel Hilbert space with the reproducing kernel

$$Q_0(s, t) = \sum_{\nu=1}^{M} \Psi_\nu(s) \cdot \Psi_\nu(t). \tag{2.7}$$

Note that the corresponding norm in H_0 is a seminorm on $W_2^m(R^d)$ with the null space

$$H_1 = W_{2,0}^m(R^d) := \{h | h(s_\mu) = 0, \mu = 1, \ldots, M\}.$$

On the other hand, we can define an inner product in H_1 according to (2.3), i.e.,

$$\langle f, g\rangle_{H_1} \tag{2.8}$$

$$:= \sum_{\alpha_1+\cdots+\alpha_d=m} \frac{m!}{\alpha_1! \cdots \alpha_d!} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \left(\frac{\partial^m f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}\right) \left(\frac{\partial^m g}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}\right) dx_1 \cdots dx_d.$$

Then the corresponding norm in H_1 is a seminorm on $W_2^m(R^d)$ with the null space H_0. Define

$$E_m(s) = \begin{cases} \theta_{m,d} \|s\|^{2m-d} \ln \|s\| & \text{for } d \text{ even}, \\ \theta_{m,d} \|s\|^{2m-d} & \text{for } d \text{ odd}, \end{cases} \tag{2.9}$$

where

$$\theta_{m,d} = \begin{cases} \dfrac{(-1)^{d/2+1+m}}{2^{2m-1}\pi^{d/2}(m-1)!(m-d/2)!} & \text{for } d \text{ even}, \\ \dfrac{\Gamma(m-d/2)}{2^{2m}\pi^{d/2}(m-1)!} & \text{for } d \text{ odd}, \end{cases}$$

and $\|\cdot\|$ is the Euclidean norm on R^d. Then the reproducing kernel of H_1 is given by [33]

$$\begin{aligned} Q_1(s, t) = {} & E_m(t-s) - \sum_{\mu=1}^{M} \Psi_\mu(t) E_m(s_\mu - s) - \sum_{\nu=1}^{M} \Psi_\nu(s) E_m(t - s_\nu) \\ & + \sum_{\mu=1}^{M}\sum_{\nu=1}^{M} \Psi_\mu(s)\Psi_\nu(t) E_m(s_\mu - s_\nu). \end{aligned} \tag{2.10}$$

It can be shown [51] that the smoothing spline estimate, the minimizer of (2.2), can be expressed as

$$f_\lambda^*(t) = \sum_{i=1}^{N} c_i E_m(t - t_i) + \sum_{\nu=1}^{M} d_\nu \phi_\nu(t), \tag{2.11}$$

and the coefficients $\vec{c} = [c_1, \ldots, c_N]'$ and $\vec{d} = [d_1, \ldots, d_M]'$ can be obtained by solving the quadratic minimization problem

$$\min_{\vec{c},\vec{d}} \left(\frac{1}{N} \|\vec{z} - T\vec{d} - K\vec{c}\|^2 + \lambda \vec{c}' K \vec{c} \right), \tag{2.12}$$

where T is in (2.4) and K is an $N \times N$ matrix with the (i, j) entry $E_m(t_i - t_j)$. Note that the choice of the points $\{s_\nu\}$ and the construction of $\{\Psi_\nu\}$ are avoided in this approach.

When the data are exact (i.e., no noise) or when the noise level is low, i.e., $\varepsilon_i \approx 0$, interpolating spline estimates can be appropriate. An interpolating spline is the minimizer of $J(g)$ in (2.3) for all $g \in W_2^m(R^d)$ with $g(t_i) = z_i, i = 1, 2, \ldots, N$. The unique interpolating spline can be expressed as

$$f_0^*(t) = \sum_{i=1}^{N} c_i E_m(t - t_i) + \sum_{\nu=1}^{M} d_\nu \phi_\nu(t), \tag{2.13}$$

and coefficients $\vec{c}$ and $\vec{d}$ can be obtained by solving the following constrained quadratic minimization problem:

$$\min_{\vec{c}} \vec{c}' K \vec{c},$$

where matrix K is the same as that in (2.12), and the constraints are

$$f_0^*(t_i) = z_i, i = 1, \ldots, N.$$

3. Discontinuities. We now discuss discontinuities, which are to be incorporated in the splines. We do not intend to be formal or complete here. A more systematic study is under way and will be included in a forthcoming paper.

We define a separating hypersurface as a hypersurface that has no intersections with itself, partitions R^d into two simply connected regions, and has Lebesgue measure zero. We assume that discontinuities can only occur on finitely many separating hypersurfaces.

In the presence of function value discontinuities, apparently, the problem could be decomposable. More specifically, the separating hypersurfaces partition R^d into a finite number of simply connected regions; we can estimate the minimizer of (2.2) in each region and then combine the minimizers of all the regions as the minimizer for the whole space. This is a valid approach for fitting data in one dimension [28] and the arguments for decomposition in higher dimensions are the same. However, there are two problems with this approach. If the data are sparse or distributed unevenly, then smoothing splines provide a poor estimate in regions with only a few or no data points. Furthermore, to find a minimizer in a (finite) region is difficult; the approach described in §2 is no longer valid, since the reproducing kernels do not have a finite form [15] and the computation is rather involved. Therefore, unlike in the case of one dimension, the problem is not decomposable in high dimensions, at least in terms of reliability of the estimation and feasibility of computation. On the other hand, in the presence of derivative discontinuities, it is obvious that the problem is not decomposable, since the function may be continuous across a separating hypersurface. Based on the above arguments, we do not decompose the problem even in the presence of function value discontinuities.

When we construct the splines we assume that we know the location and type of discontinuities. However, to determine the location and degree of discontinuities and measure their sizes is an important yet complicated problem [28]. In practice, the available information is

usually function value discontinuities whose size can be well approximated by a low-degree piecewise polynomial. For simplicity, in this paper we only consider the case when the discontinuities are on the function values and their sizes can be well approximated by a low-degree piecewise polynomial. For the most general case, where we have derivative discontinuities with sizes that are of arbitrary functional forms, the situation is more involved, and we shall discuss it in a forthcoming paper.

For simplicity, assume that there is only one separating hypersurface and that there are only function value discontinuities along the hypersurface. The size of discontinuities may vary with different locations on the hypersurface. We model the size of the discontinuities by low-degree piecewise polynomials. Informally, let $\gamma(t)$ be an indicator function, which has value 0 on one side and value 1 on the other side of the separating hypersurface. We approximate the size of discontinuities by piecewise polynomials in P_{m-1}, denoted by $\bar{P}_{m-1}$. Consequently, the discontinuities are represented by a function in the space $\langle \bar{\phi}_1\gamma, \ldots, \bar{\phi}_{\bar{M}}\gamma \rangle$, where $\{\bar{\phi}_i\}$ is a basis of $\bar{P}_{m-1}$ and $\bar{M}$ is its dimension. Note that the linear space $\bar{P}_{m-1}$ depends not only on m but also on where and how the pieces of polynomials in P_{m-1} are joined together, i.e., the smoothness constraints at the junctions. On the other hand, we take the integrand in (2.3) as defined on R^d almost everywhere except on the hypersurface where discontinuity occurs and at the location of the junctions of the pieces of polynomials.[2] The null space of $J(\cdot)$ in (2.3) is $\bar{P}_{m-1}$. However, we choose not to digress to a study of piecewise polynomials in high dimensions, which is a topic by itself. For clarity, from now on we only discuss the case when discontinuity size is approximated by polynomials in P_{m-1}, i.e., $\bar{P}_{m-1} = P_{m-1}$. All the results hold for the general case except those for the bases and consequently the dimensions of the null space.

4. Smoothing splines with discontinuities. We assume that the size of discontinuities can be approximated by a polynomial in $H_0 = P_{m-1}$. Consequently, the discontinuities are represented by a function in the space $H_2 = \langle \phi_1\gamma, \ldots, \phi_M\gamma \rangle$ where $\{\phi_i\}$ is a basis of H_0.

Assume that $H_2 \cap W_2^m(R^d) = \{0\}$. Let $f_1 = \sum_{\nu=1}^{M} \theta_\nu^{(1)}\phi_\nu\gamma$ and $f_2 = \sum_{\nu=1}^{M} \theta_\nu^{(2)}\phi_\nu\gamma$ be any two functions in H_2. We define an inner product on H_2 by

$$\langle f_1, f_2 \rangle_{H_2} = \sum_{\nu=1}^{M} \theta_\nu^{(1)}\theta_\nu^{(2)}.$$

Then it can be shown [41] that H_2 is a reproducing kernel Hilbert space with the reproducing kernel

$$Q_2(s,t) = \gamma(s)\gamma(t)\sum_{\nu=1}^{M} \phi_\nu(s)\cdot\phi_\nu(t). \tag{4.1}$$

To cope with the discontinuities, we expand the function space to include H_2. To make H_0 orthogonal to H_2, we need to redefine the inner product of H_0. Since $\{\phi_i\}$ is a basis for H_0, let $f_1 = \sum_{\nu=1}^{M} a_\nu^{(1)}\phi_\nu$ and $f_2 = \sum_{\nu=1}^{M} a_\nu^{(2)}\phi_\nu$ be any two functions in H_0. We define an inner product on H_0 by

$$\langle f_1, f_2 \rangle_{H_0} = \sum_{\nu=1}^{M} a_\nu^{(1)}a_\nu^{(2)}.$$

[2]Functions g in (2.3) has prespecified derivatives at the location of the junctions that are determined by the smoothness constraints of functions in $\bar{P}_{m-1}$.

Then it can be easily shown that H_0 is a reproducing kernel Hilbert space with the reproducing kernel

$$Q_0(s,t) = \sum_{\nu=1}^{M} \phi_\nu(s) \cdot \phi_\nu(t). \tag{4.2}$$

In the presence of discontinuities, function $J(\cdot)$ in (2.3) is interpreted differently. The integrand is defined on R^d almost everywhere except on the hypersurfaces, where discontinuity occurs and which are of measure zero. By observing that $J(f) = 0$ for $f \in H_0 \oplus H_2$, we define $H = H_0 \oplus H_1 \oplus H_2$. It can be shown [41] that H is a reproducing kernel Hilbert space with the reproducing kernel

$$Q(s,t) = Q_0(s,t) + Q_1(s,t) + Q_2(s,t),$$

where Q_0, Q_1 and Q_2 are in (4.2), (2.10), and (4.1), respectively.

Let L_i be the evaluation functional at t_i, $i = 1, \ldots, N$, i.e., $L_i(g) = g(t_i)$ for $g \in H$. Since H is a reproducing kernel Hilbert space, L_i is a continuous linear functional. By the Riesz Representation Theorem, there exists $\eta_i \in H$, the representer of L_i, and, by the reproducing property, $\eta_i = Q(\cdot, t_i)$. Let $\xi_i = P_1\eta_i$, where P_1 is the projection operator to H_1. Then

$$\xi_i(s) = P_1 Q(s,t_i) = Q_1(s,t_i). \tag{4.3}$$

A function in H can be expressed as

$$g = \sum_{i=1}^{N} a_i\xi_i + \sum_{\nu=1}^{M} d_\nu\phi_\nu + \sum_{\nu=1}^{M} \theta_\nu\phi_\nu\gamma + \rho, \tag{4.4}$$

where a_i's, d_ν's, and θ_ν's are real numbers, and $\rho \perp \langle \xi_1, \ldots, \xi_N, \phi_1, \ldots, \phi_M, \phi_1\gamma, \ldots, \phi_M\gamma \rangle$. The orthogonality of ρ to the finite dimensional space spanned by ξ_i's, ϕ_i's, and $\phi_i\gamma$'s and the reproducing property make $\rho(t_i) = \langle \xi_i, \rho \rangle = 0$ for $i = 1, \ldots, N$.

Substituting (4.4) into (2.2), we have

$$\min_{\vec{\beta},\vec{c},\rho} \left(\frac{1}{N}\|\vec{z} - T\vec{\beta} - W\vec{c}\|^2 + \lambda\vec{c}'W\vec{c} + \lambda\|\rho\|_2^2 \right), \tag{4.5}$$

where $\|\cdot\|_2$ is the L_2-norm, $\vec{z}$ is the data in (2.1), $\vec{\beta} = [d_1, \ldots, d_M, \theta_1, \ldots, \theta_M]'$, $\vec{c} = [c_1, \ldots, c_N]'$, T is now changed to an $N \times 2M$ matrix

$$T = \begin{pmatrix} \phi_1(t_1) & \cdots & \phi_M(t_1) & \phi_1(t_1)\gamma(t_1) & \cdots & \phi_M(t_1)\gamma(t_1) \\ \cdot & \cdots & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot & \cdot & \cdots & \cdot \\ \phi_1(t_N) & \cdots & \phi_M(t_N) & \phi_1(t_N)\gamma(t_N) & \cdots & \phi_M(t_N)\gamma(t_N) \end{pmatrix}, \tag{4.6}$$

and W is an $N \times N$ matrix whose (i, j) entry is $Q_1(t_i, t_j)$, $i, j = 1, \ldots, N$.

Immediately we note that $\rho = 0$ and the coefficients $\vec{c}$ and $\vec{\beta}$ can be easily computed by solving the quadratic minimization

$$\min_{\vec{\beta},\vec{c}} \left(\frac{1}{N}\|\vec{z} - T\vec{\beta} - W\vec{c}\|^2 + \lambda\vec{c}'W\vec{c} \right). \tag{4.7}$$

Note that the computation of the matrix W involves the construction of $\{s_\nu\}$ and $\{\Psi_\nu\}$. Similar to [51], we can bypass W and directly work on an $N \times N$ matrix K whose (i, j) entry is $E_m(t_i - t_j)$, $i, j = 1, \ldots, N$, where E_m is given in (2.9). More specifically,

$$K = \begin{pmatrix} 0 & E_m(t_1 - t_2) & \cdots & E_m(t_1 - t_{N-1}) & E_m(t_1 - t_N) \\ E_m(t_2 - t_1) & 0 & \cdots & E_m(t_2 - t_{N-1}) & E_m(t_2 - t_N) \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ E_m(t_{N-1} - t_1) & E_m(t_{N-1} - t_2) & \cdots & 0 & E_m(t_{N-1} - t_N) \\ E_m(t_N - t_1) & E_m(t_N - t_2) & \cdots & E_m(t_N - t_{N-1}) & 0 \end{pmatrix}. \tag{4.8}$$

From (2.10), (4.4) can be rewritten as

$$g = \sum_{i=1}^{N} c_i E_m(\cdot - t_i) + \sum_{\nu=1}^{M} d_\nu \phi_\nu + \sum_{\nu=1}^{M} \theta_\nu \phi_\nu \gamma + \rho^*, \tag{4.9}$$

where c_i's, d_ν's and θ_ν's are real numbers, and $\rho^* \perp \langle \xi_1^*, \ldots, \xi_N^*, \phi_1, \ldots, \phi_M, \phi_1\gamma, \ldots, \phi_M\gamma \rangle$ with $\xi_i^* = E_m(\cdot - t_i)$, $i = 1, \ldots, N$.

It is known that although $E_m(\cdot - s)$ is not in H_1, it has the desired reproducing property in H_1. That is,

$$\langle E_m(\cdot - s), f(\cdot) \rangle = f(s) \quad \text{for } f \in H_1. \tag{4.10}$$

Furthermore, it can be verified that

$$\langle E_m(\cdot - s), E_m(\cdot - t) \rangle = E_m(s, t) = E_m(s - t). \tag{4.11}$$

By (4.10) and the orthogonality of ρ^* to ξ_i^*'s, we have $\rho^*(t_i) = \langle E_m(\cdot - t_i), \rho^*(\cdot) \rangle = 0$.

Substituting (4.9) into (2.2), from (4.10) and (4.11), we have

$$\min_{\vec{\beta}, \vec{c}, \rho^*} \left(\frac{1}{N} \| \vec{z} - T\vec{\beta} - K\vec{c} \|^2 + \lambda \vec{c}' K \vec{c} + \lambda \| \rho^* \|_2^2 \right),$$

which immediately leads to $\rho^* = 0$, and the smoothing spline estimate is

$$f_\lambda^*(t) = \sum_{i=1}^{N} c_i E_m(t - t_i) + \sum_{\nu=1}^{M} d_\nu \phi_\nu(t) + \sum_{\nu=1}^{M} \theta_\nu \phi_\nu(t) \gamma(t), \tag{4.12}$$

where the coefficients $\vec{c} = [c_1, \ldots, c_N]'$ and $\vec{\beta} = [d_1, \ldots, d_M, \theta_1, \ldots, \theta_M]'$ are computed by solving the quadratic minimization

$$\min_{\vec{\beta}, \vec{c}} \left(\frac{1}{N} \| \vec{z} - T\vec{\beta} - K\vec{c} \|^2 + \lambda \vec{c}' K \vec{c} \right). \tag{4.13}$$

5. Interpolating splines with discontinuities. When the data is exact or when the noise level is low, interpolating spline estimates could be more appropriate. An interpolating spline is defined as the minimizer of $J(g)$ in (2.3) for all $g \in H$ with $g(t_i) = z_i$, $i = 1, 2, \ldots, N$.

Similar to smoothing splines with discontinuities, the unique interpolating spline estimate can be expressed as

$$f_0^*(t) = \sum_{i=1}^{N} c_i E_m(t - t_i) + \sum_{\nu=1}^{M} d_\nu \phi_\nu(t) + \sum_{\nu=1}^{M} \theta_\nu \phi_\nu(t) \gamma(t), \tag{5.1}$$

where the coefficients are computed by a constrained quadratic minimization

$$\min_{\vec{c}} \vec{c}' K \vec{c} \tag{5.2}$$

with constraints

$$f_0^*(t_i) = z_i, i = 1, \ldots, N, \tag{5.3}$$

where K is the matrix in (4.8).

6. Fast algorithms for computing the splines. To compute the smoothing and interpolating splines we solve the quadratic minimization problems in (4.13) and (5.2), respectively. They can be easily reduced to solving systems of linear equations. For splines without discontinuities, direct methods have been proposed with a cost $O(N^3)$ [33], [51], [41]. We now present four different fast algorithms for the splines with or without discontinuities with a cost $O(N^2 \log N)$.

The first approach is based on a fast algorithm for multiplying a recursive block Toeplitz matrix by a vector. The algorithm was reported in [27] and applied to the computation of interpolating thin plate splines in two dimensions (without discontinuities). We now present our results for both smoothing and interpolating thin plate splines with or without discontinuities and in arbitrary dimensions. This algorithm is numerically stable.

The other three approaches are based on fast block Toeplitz matrix inversion, LU decomposition, and QR decomposition, respectively. However, they can only be used for smoothing spline computations, and their numerical stability remains to be further explored.

We first examine the recursive block Toeplitz structure of the $N \times N$ matrix K in (4.8) and describe a fast algorithm for multiplying this matrix by a vector. Based on this result, we obtain the first algorithm. We then discuss the inversion of recursive block Toeplitz matrices and derive the other three algorithms. Finally, we compare and comment on different issues such as space decomposition and numerical computations.

6.1. Recursive block Toeplitz matrices. An $N \times N$ Toeplitz matrix is a matrix $A = (b_{i,j})$, $i, j = 1, \ldots, N$, with the property that $b_{i,j} = b_{i-1,j-1}$, $2 \le i, j \le N$. Let $a_{j-i} = b_{i,j}$; then we have

$$A = \begin{bmatrix} a_0 & a_1 & \cdot & \cdot & a_{N-1} \\ a_{-1} & a_0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & a_1 \\ a_{-(N-1)} & \cdot & \cdot & a_{-1} & a_0 \end{bmatrix}. \tag{6.1}$$

We consider the following $n \times n$ block matrix $A^{(2)} = (B_{i,j}^{(1)})$, which is a block Toeplitz matrix, i.e., $B_{i,j}^{(1)} = B_{i-1,j-1}^{(1)}$, $2 \le i, j \le n$, and each block $B_{i-1,j-1}^{(1)}$ is an $n \times n$ Toeplitz matrix itself. Since there are two levels of blocking, and at each level it is an $n \times n$ (block) Toeplitz matrix, we call it a 2-level recursive block Toeplitz matrix with blocking factor n. Obviously, $A^{(2)}$ is an $N \times N$ matrix, where $N = n^2$.

Let $A^{(1)}_{j-i} = B^{(1)}_{i,j}$; then we have

$$A^{(2)} = \begin{bmatrix} A^{(1)}_0 & A^{(1)}_1 & \cdot & \cdot & A^{(1)}_{n-1} \\ A^{(1)}_{-1} & A^{(1)}_0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & A^{(1)}_1 \\ A^{(1)}_{-(n-1)} & \cdot & \cdot & A^{(1)}_{-1} & A^{(1)}_0 \end{bmatrix}. \tag{6.2}$$

Since each block $A^{(1)}_i$ is a Toeplitz matrix, let

$$A^{(1)}_i = \begin{bmatrix} a_{i,0} & a_{i,1} & \cdot & \cdot & a_{i,n-1} \\ a_{i,-1} & a_{i,0} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & a_{i,1} \\ a_{i,-(n-1)} & \cdot & \cdot & a_{a,-1} & a_{i,0} \end{bmatrix}. \tag{6.3}$$

Here, the first subscript i of each entry $a_{i,j}$ is used to specify its position in $A^{(2)}$ of (6.2), and the second subscript j is used to specify its position in $A^{(1)}_i$ of (6.3), where $i, j = -(n-1), \ldots, (n-1)$.

Generally, a d-level recursive block Toeplitz matrix with blocking factor n is defined recursively as an $n \times n$ block Toeplitz matrix such that each block is a $(d\text{-}1)$-level recursive block Toeplitz matrix with the same blocking factor. Much effort has been devoted to studying Toeplitz and recursive block Toeplitz matrices [48], [6], [27].

We are interested in fast multiplications of a d-level recursive block Toeplitz matrix $A^{(d)}$ by a vector $U^{(d)}$ with a cost $O(N \log N)$, where $N = n^d$, and n is the blocking factor. Let

$$D^{(d)} = A^{(d)} U^{(d)}. \tag{6.4}$$

An entry of $A^{(d)}$ is $a_{i_1,\ldots,i_d}$, where i_r specifies its position in the rth level block matrix, $r = 1, \ldots, d$.

We can impose a recursive block structure on the $N \times 1$ vector $U^{(d)}$ similar to that of $A^{(d)}$; $U^{(d)}$ is a d-level recursive block vector: $U^{(d)} = [U^{(d-1)}_1, \ldots, U^{(d-1)}_n]'$, where each $U^{(d-1)}_i$, $i = 1, \ldots, n$, is a $(d-1)$-level recursive block vector, and the final level of blocks is simply $n \times 1$ vectors. The $(i_1, \ldots, i_d)$th entry of $U^{(d)}$, $u_{i_1,\ldots,i_d}$, is the one belonging to the i_1th block at the first level of blocking, ..., to the i_{d-1}th block at the $(d-1)$th level of blocking, and is the i_dth entry of the final level of blocking. We impose the same structure on $D^{(d)}$, and denote the $(i_1, \ldots, i_d)$th entry by $d_{i_1,\ldots,i_d}$.

6.2. Fast multiplication of a recursive block Toeplitz matrix by a vector. In this subsection we describe a fast algorithm, which multiplies a recursive block Toeplitz matrix by a vector. For details, see [27].

We define two polynomials $A(t)$ and $U(t)$, and we reduce the matrix multiplication problem to a problem of multiplying the two polynomials $A(t)$ and $U(t)$. The degree of the two polynomials is of order $O(N)$, and we can use the fast fourier transform (FFT) for the polynomial multiplication with a cost $O(N \log N)$ [2].

Let

$$A(t) = \sum_{i_1=-(n-1)}^{n-1} \cdots \sum_{i_d=-(n-1)}^{n-1} a_{i_1,\ldots,i_d} t^{\lambda_{i_1,\ldots,i_d}}, \tag{6.5}$$

where $a_{i_1,\ldots,i_d}$ is an entry of matrix $A^{(d)}$, and $\lambda_{i_1,\ldots,i_d} = \sum_{l=1}^{d}(n + i_l - 1)(2n-1)^{d-l}$.

Let

$$U(t) = \sum_{j_1=1}^{n} \cdots \sum_{j_d=1}^{n} u_{j_1,\ldots,j_d} t^{\nu_{j_1,\ldots,j_d}}, \tag{6.6}$$

where $\nu_{j_1,\ldots,j_d} = \frac{1}{2}[(2n-1)^d - 1] - \sum_{l=1}^{d}(j_l - 1)(2n-1)^{d-l}$.

The following algorithm multiplies an $N \times N$ d-level recursive block Toeplitz matrix by a vector in time $O(N \log N)$.

ALGORITHM 6.1. Fast multiplication of a d-level recursive block Toeplitz matrix $A^{(d)}$ by a vector $U^{(d)}$.

1. Construct polynomials $A(t)$ and $U(t)$ from $A^{(d)}$ and $U^{(d)}$, as in (6.5) and (6.6).
2. Compute $P(t) = A(t) \times U(t)$ using the FFT.
3. The $(i_1, \ldots, i_d)$th entry of $D^{(d)} = A^{(d)} \times U^{(d)}$ is $d_{i_1,\ldots,i_d} = p_\xi$, where p_ξ is the coefficient of t^ξ in $P(t)$, and

$$\xi = \frac{(2n-1)[(2n-1)^d - 1]}{2(n-1)} - \sum_{l=1}^{d} i_l (2n-l)^{d-l}.$$

THEOREM 6.1. *Algorithm* 6.1 *multiplies an* $N \times N$ *d-level recursive block Toeplitz matrix by a vector in time* $O(N \log N)$.

6.3. Matrix *K*. Assume that data are given on regular grids in a d-cube in R^d. For simplicity, assume that $N = n^d$ where n is the resolution in each dimension. We now show that matrix K in (4.8) is a symmetric d-level recursive block Toeplitz matrix with blocking factor n.

For clarity, we only prove it for $d = 2$, and the general case can be proved by induction. The N data points t_i, $i = 1, \ldots, N$, are obtained on regular grids in row order, and we relabel them as $t_{i,j}$, $i, j = 1, \ldots, n$. Matrix K in (4.8) is an $n^2 \times n^2$ matrix, or an $n \times n$ block matrix with each block in an $n \times n$ matrix.

One checks that $K_{i,k;j,l}$, the (k, l)th entry of the (i, j)th block of K is on the $((i-1)n+k)$th row and $((j-1)n + l)$th column of K, which is $E_m(t_{i,k} - t_{j,l})$. The $(k+1, l+1)$th entry in the same block is $E_m(t_{i,k+1} - t_{j,l+1})$. Since $t_{i,j}$'s are on regular grids, the Euclidean distance is $\|t_{i,k} - t_{j,l}\| = \|t_{i,k+1} - t_{j,l+1}\|$, and from (2.9) $K_{i,k+1;j,l+1} = K_{i,k;j,l}$, i.e., each block is Toeplitz. Similarly, $K_{i+1,k;j+1,l} = K_{i,k;j,l}$, and K is a block Toeplitz matrix. One easily checks that K is symmetric, and we have the following proposition.

PROPOSITION 6.1. *Given data points on regular grids, the matrix* K *in* (4.8) *is an* $N \times N$ *symmetric d-level recursive block Toeplitz matrix with blocking factor* n *and* $N = n^d$.

6.4. Computing smoothing splines. Setting the partial derivatives of the objective function in (4.13) with respect to each parameter to zero, we have

$$\begin{cases} (K + N\lambda I)\vec{c} + T\vec{\beta} = \vec{z}, \\ T'\vec{c} = \vec{0}, \end{cases} \tag{6.7}$$

where I is an $N \times N$ identity matrix. Denote $S = K + N\lambda I$. Note that since K is symmetric, S is also symmetric. Let $K_{\max}$ be the largest entry in the matrix K. One easily checks that S is diagonally dominant and hence positive definite if $\lambda > |K_{\max}|/\sqrt{N}$.[3] Then we can solve

[3]This assumption is not essential. Without it, an approach similar to that in §6.5 can be used with a cost $O(N^2(M + \log N))$. See §6.7.

the system in (6.7) uniquely, since T is of full rank. We have

$$\vec{\beta} = (T'S^{-1}T)^{-1}T'S^{-1}\vec{z} \quad \text{and} \quad \vec{c} = S^{-1}(\vec{z} - T\vec{\beta}). \tag{6.8}$$

The vectors $\vec{\beta}$ and $\vec{c}$ can be computed in the following six steps:

$$\begin{aligned} &1.\ \vec{y_1} = S^{-1}\vec{z};\\ &2.\ \vec{y_2} = T'\vec{y_1};\\ &3.\ \vec{\beta} = (T'S^{-1}T)^{-1}\vec{y_2};\\ &4.\ \vec{y_3} = T\vec{\beta};\\ &5.\ \vec{y_4} = \vec{z} - \vec{y_3};\\ &6.\ \vec{c} = S^{-1}\vec{y_4}. \end{aligned} \tag{6.9}$$

Steps 2 and 4 cost $O(MN)$, and step 5 costs $O(N)$. Steps 1 and 6 can be done by solving the system of linear equations $S\vec{y} = \vec{b}$. Direct methods such as the Cholesky factorization cost $O(N^3)$ [33]. Instead, we use the conjugate gradient method, which converges in no more than N iterations [23], [46]. At each iteration, we multiply the coefficient matrix S by an N vector. Since $S = K + N\lambda I$ and K is a recursive block Toeplitz matrix, the cost is $O(N \log N)$, using Algorithm 6.1. Therefore, the total cost is $O(N^2 \log N)$. Step 3 is to invert an $M \times M$ matrix $T'S^{-1}T$ with the cost $O(M^3)$ and then multiply by a $2M$ vector with the cost $O(M^2)$. However, to compute $T'S^{-1}T$ could be costly. We compute $S^{-1}T$ first where T is an $N \times 2M$ matrix. To compute $S^{-1}T$, we only have to multiply S^{-1} by each column of T. Using the same approach as for steps 1 and 6, we determine that the cost for each column multiplication is $O(N^2 \log N)$ and the total cost of computing $S^{-1}T$ is $O(MN^2 \log N)$. Finally, the computation of $T'(S^{-1}T)$ costs $O(M^2N)$. Note that $N >> M$ and we have the following theorem.

THEOREM 6.2. *Given data points on regular grids, smoothing splines in* (4.12) *can be computed in time* $O(MN^2 \log N)$.

With a different decomposition of the function space H, smoothing splines can be computed in time $O(N^2(M + \log N))$. See item (i) in §6.7.

6.5. Computing interpolating splines. To solve the minimization problem in §5 for the interpolating spline, we apply Lagrange multipliers and obtain

$$\begin{cases} K\vec{c} + K\vec{\omega} = \vec{0},\\ T'\vec{\omega} = \vec{0},\\ K\vec{c} + T\vec{\beta} - \vec{z} = \vec{0}, \end{cases} \tag{6.10}$$

where $\vec{\omega}$ are the multipliers. It can be easily checked that the coefficient matrix is symmetric but not positive definite since the trace is 0. Therefore, we may have difficulties in solving the system by either direct or iterative methods.

Assume that system (6.10) is $A\vec{x} = \vec{b}$, where the coefficient matrix A is symmetric and of full rank. Let $\vec{x} = A\vec{y}$. We solve $A^2\vec{y} = \vec{b}$ for $\vec{y}$ first and then compute $\vec{x} = A\vec{y}$. To solve for $\vec{y}$, the coefficient matrix A^2 is symmetric and positive definite. However, the condition number of A^2 is the square of that of A, and that can cause problems in the numerical computation.

We now use an approach that leads to a system of linear equations with symmetric and positive definite coefficient matrix without increasing the condition number. This approach is similar to that in [33] for interpolating splines without discontinuities. However, in the presence of discontinuities, it is more involved. We shall decompose H differently than in §4 and use different reproducing kernels.

By the assumption that the matrix T is of full rank, we can find $2M$ data points such that the corresponding $2M$ rows in (4.6) are linearly independent. Denote them by $\{t_{k_\nu}, \nu = 1, \ldots, 2M\}$.

Let $\tilde{H}_0 = \langle \phi_1, \ldots, \phi_M, \phi_1\gamma, \ldots, \phi_M\gamma \rangle$. We then find $\tilde{\Psi}_\nu \in \tilde{H}_0, \nu = 1, \ldots, 2M$ such that

$$\tilde{\Psi}_\nu(t_{k_\mu}) = \delta_{\nu,\mu}, \qquad \nu, \mu = 1, \ldots, 2M, \tag{6.11}$$

where $\delta_{\nu,\mu}$ is the Kronecker delta defined in (2.5).

Let $\tilde{H}_1 = \tilde{W}^m_{2,0}(R^d) := \{h \in H | h(t_{k_\nu}) = 0, \nu = 1, \ldots, 2M\}$. Similar to the arguments in §2 for smoothing splines without discontinuities, we can define corresponding inner products on $\tilde{H}_0$ and $\tilde{H}_1$, respectively, such that $H = \tilde{H}_0 \oplus \tilde{H}_1$. It can be shown that $\tilde{H}_1$ is a reproducing kernel Hilbert space with the reproducing kernel

$$\begin{aligned} \tilde{Q}_1(s,t) = {} & E_m(t-s) - \sum_{\mu=1}^{2M} \tilde{\Psi}_\mu(t) E_m(t_{k_\mu} - s) - \sum_{\nu=1}^{2M} \tilde{\Psi}_\nu(s) E_m(t - t_{k_\nu}) \\ & + \sum_{\mu=1}^{2M} \sum_{\nu=1}^{2M} \tilde{\Psi}_\mu(s) \tilde{\Psi}_\nu(t) E_m(t_{k_\mu} - t_{k_\nu}), \end{aligned} \tag{6.12}$$

where E_m is in (2.9).

By a similar argument as in [33], the interpolating spline can be represented as

$$f_0^*(t) = \sum_{j=2M+1}^{N} c_j \tilde{Q}_1(t, t_{k_j}) + \sum_{\nu=1}^{2M} z_{k_\nu} \tilde{\Psi}(t). \tag{6.13}$$

Given the $N \times N$ matrix

$$Q = \begin{pmatrix} \tilde{Q}_1(t_1, t_1) & \cdots & \tilde{Q}_1(t_1, t_N) \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \tilde{Q}_1(t_N, t_1) & \cdots & \tilde{Q}_1(t_N, t_N) \end{pmatrix}, \tag{6.14}$$

we delete the k_νth rows and columns, $\nu = 1, \ldots, 2M$, and obtain an $(N - 2M) \times (N - 2M)$ matrix $\tilde{Q}$. From the selection of k_ν's, since $\tilde{Q}_1$ is the reproducing kernel of $\tilde{H}_1$, $\tilde{Q}$ is the Gram matrix of $(N - 2M)$ linearly independent elements in space $\tilde{H}_1$ and is symmetric and positive definite.

To obtain c_j's, we substitute the remaining $N - 2M$ data values into (6.13):

$$\sum_{j=2M+1}^{N} \tilde{Q}_1(t_{k_i}, t_{k_j}) c_j = z_{k_i} - \sum_{\nu=1}^{2M} z_{k_\nu} \tilde{\Psi}_\nu(t_{k_i}), \qquad i = 2M+1, \ldots, N. \tag{6.15}$$

Obviously, this is

$$\tilde{Q}\vec{c} = \vec{a}, \tag{6.15'}$$

where the $(N - 2M) \times (N - 2M)$ matrix $\tilde{Q}$ is obtained from (6.14), and the $(N - 2M) \times 1$ vector $\vec{a}$ consists of constants $a_i = z_{k_i} - \sum_{\nu=1}^{2M} z_{k_\nu} \tilde{\Psi}_\nu(t_{k_i})$, $i = 2M + 1, \ldots, N$.

Since $\tilde{Q}$ is symmetric and positive definite, direct methods such as Cholesky factorization can be applied with a cost $O(N^3)$. We can use the conjugate gradient method, which converges in no more than $N - 2M$ iterations. At each iteration, we multiply $\tilde{Q}$ by a vector. However, $\tilde{Q}$ is not a recursive block Toeplitz matrix any more, and additional efforts are needed.

At each iteration, we multiply the $(N-2M) \times (N-2M)$ matrix $\tilde{Q}$ by an $(N-2M) \times 1$ vector $\vec{y}$. We can expand $\vec{y}$ into an $N \times 1$ vector $\vec{x}$ such that the added k_νth entries are 0, $\nu = 1, \ldots, 2M$. We then compute $Q\vec{x}$ and delete the k_νth entry of the product, $\nu = 1, \ldots, 2M$. Therefore, if we can multiply the $N \times N$ matrix Q by a vector in time $O(N \log N)$, we have the desired complexity.

From (6.14), (6.12), and (4.8), the ith entry of $Q\vec{x}$ is, $i = 1, \ldots, N$:

$$(Q\vec{x})_i \tag{6.16}$$

$$= (K\vec{x})_i \tag{6.16a}$$

$$- \sum_{\mu=1}^{2M} \tilde{\Psi}_\mu(t_i) \sum_{j=1}^{N} K_{k_\mu, j} x_j \tag{6.16b}$$

$$- \sum_{\nu=1}^{2M} K_{i,k_\nu} \sum_{j=1}^{N} \tilde{\Psi}_\nu(t_j) x_j \tag{6.16c}$$

$$+ \sum_{\mu=1}^{2M} \sum_{\nu=1}^{2M} \tilde{\Psi}_\mu(t_i) K_{k_\mu, k_\nu} \sum_{j=1}^{N} \tilde{\Psi}_\nu(t_j) x_j. \tag{6.16d}$$

To compute (6.16a), the approach in §6.4 applies and takes time $O(N \log N)$. The second multiplier of (6.16b) is the k_μth entry $(K\vec{x})_{k_\mu}$, which has been obtained from (6.16a). To compute (6.16c), we precompute $\sum_{j=1}^{N} \tilde{\Psi}_\nu(t_j) x_j$ for all ν in time $O(2MN)$, and then compute all the entries in time $O(MN)$. Similarly, we compute (6.16d) in time $O(MN + M^2)$. Since $M << N$, it takes time $O(N(M + \log N))$ to compute $Q\vec{x}$. We summarize this in the following theorem.

THEOREM 6.3. *Given data points on regular grids, interpolating splines in* (5.1) *can be computed in time* $O(N^2(M + \log N))$.

6.6. Three more fast algorithms. We now discuss different fast algorithms for computing the spline estimates. However, they are only applicable for smoothing splines. These approaches use different existing fast algorithms for block Toeplitz matrices and take advantage of the recursive block Toeplitz structure of the matrices involved, yielding a cost $O(N^2(M + \log N))$. We describe the algorithms informally.

To compute the smoothing splines, we compute $\vec{\beta}$ and $\vec{c}$ in (6.8). The dominant cost is to compute $S^{-1}T$ where T is an $N \times 2M$ matrix and $S = K + N\lambda I$. From Proposition 6.1, K is a symmetric d-level recursive block Toeplitz matrix, I is an identity matrix, and λ is a sufficiently large constant so that S is positive definite. Therefore, the $N \times N$ matrix S is a symmetric and positive definite d-level recursive block Toeplitz matrix with blocking factor n and $N = n^d$.

We first discuss inversion of a recursive block Toeplitz matrix. We apply a classical algorithm [3] for block Toeplitz matrix inversion with a cost of n^2 block operations where each block is a $(d-1)$-level recursive block Toeplitz matrix. Among the block operations, inversion and multiplication have a dominant cost. We apply the algorithm recursively.

We now show by an induction on the level of recursion d that inversion takes time $O(n^{2d}\log n^{d-1})$. For an $n \times n$ Toeplitz matrix, inversion takes time $O(n^2)$ and multiplication $O(n^2 \log n)$ [21]. For $d = 2, n^2$ block operations are needed where each block is an $n \times n$ Toeplitz matrix. Since block multiplication cost dominates, the cost to invert a 2-level recursive block Toeplitz matrix is $O(n^4 \log n)$. Suppose that inversion of a $(d-1)$-level recursive block Toeplitz matrix takes time $O(n^{2(d-1)}\log n^{(d-1)-1})$, $d \geq 2$. We now show that inversion of a d-level recursive block Toeplitz matrix costs $O(n^{2d}\log n^{d-1})$. Indeed, it takes n^2 block operations where each block is a $(d$-1)-level recursive block Toeplitz matrix. By the induction hypothesis, inversion of such a block costs $O(n^{2(d-1)}\log n^{d-2})$. It costs $O(n^{d-1}\log n^{d-1})$ to multiply such a block by a vector if we use Algorithm 6.1, and therefore a multiplication of two such blocks costs $O(n^{2(d-1)}\log n^{d-1})$, which dominates the inversion cost. Therefore, the total cost is $O(n^{2d}\log n^{(d-1)})$. Since $N = n^d$, we have the following proposition.

PROPOSITION 6.2. *It takes time $O(n^{2d}\log n^{d-1})$, which is $O(N^2 \log N)$, to compute the inversion of a nonsingular d-level recursive block Toeplitz matrix with blocking factor n.*

The above result is used for the following three approaches to compute $S^{-1}T$. Recall that T is an $N \times 2M$ matrix and that S is a symmetric and positive definite d-level recursive block Toeplitz matrix with blocking factor n and $N = n^d$.

(i) Inversion. The simplest method is to compute S^{-1} in time $O(N^2 \log N)$ and then compute $S^{-1}T$ in time $O(MN^2)$. Therefore, using a fast inversion of recursive block Toeplitz matrices, $S^{-1}T$ and hence the smoothing spline in (4.12) can be computed in time $O(N^2(M + \log N))$.

(ii) LU decomposition. Suppose that the ith column of T is $\vec{t_i}$. Then $S^{-1}T$ can be obtained from $S^{-1}\vec{t_i}$, $i = 1, \ldots, 2M$. Note that $S^{-1}\vec{t_i} = \vec{x_i}$ can be computed by solving the system of linear equations $S\vec{x_i} = \vec{t_i}$. We compute an LU decomposition of $S = LU$ where L and U are lower and upper triangular matrices, respectively. Then we can solve the system $LU\vec{x_i} = \vec{t_i}$ by back substitutions in time $O(N^2)$, $i = 1, \ldots, 2M$.

For a fast LU decomposition of S, we use the algorithm in [39] and the cost is n^2 block operations, of which inversion and multiplication have a dominant cost. Since each block is a $(d$-1)-level recursive block Toeplitz matrix, from Proposition 6.2, each block inversion costs $O(n^{2(d-1)}\log n^{d-2})$ and multiplication $O(n^{2(d-1)}\log n^{d-1})$. Therefore, the total cost of an LU decomposition is $O(n^{2d}\log n^{d-1})$, which is $O(N^2 \log N)$.

In summary, using a fast LU decomposition, $S^{-1}T$ and hence the smoothing spline in (4.12) can be computed in time $O(N^2(M + \log N))$.

(iii) QR decomposition. Instead of LU decomposition, QR decomposition of S can also be used to solve $S\vec{x_i} = \vec{t_i}$. We first decompose $S = \mathcal{T}\mathcal{R}$ where $\mathcal{T}$ is an orthogonal matrix and $\mathcal{R}$ is an upper triangular matrix. To solve the system $S\vec{x_i} = \vec{t_i}$, we have $\mathcal{T}\mathcal{R}\vec{x_i} = \vec{t_i}$, and $\mathcal{R}\vec{x_i} = \mathcal{T}'\vec{t_i}$ where $\mathcal{T}'$ is the transposition of $\mathcal{T}$. We first compute $\mathcal{T}'\vec{t_i}$ in time $O(N^2)$ and then a back substitution gives $\vec{x_i}$.

A fast QR decomposition can be obtained using the method in [44], which also requires n^2 block operations. Similarly, each block operation costs $O(n^{2(d-1)}\log n^{d-1})$ and the total cost of a QR decomposition is $O(n^{2d}\log n^{d-1})$, which is $O(N^2 \log N)$.

In summary, using a fast QR decomposition, $S^{-1}T$ and hence the smoothing spline in (4.12) can be computed in time $O(N^2(M + \log N))$.

For interpolating splines, the coefficient matrix in (6.15′) is obtained by deleting $2M$ rows and columns from matrix Q in (6.14). On the other hand, matrix Q is *not* block Toeplitz either; it is obtained by changing every entry of a block Toeplitz matrix; see (6.12). It seems to be difficult to compute the interpolating splines from (6.15′) by applying known fast algorithms on block Toeplitz matrices.

To conclude this subsection, we comment that the numerical stability of these three fast algorithms is still not well understood. Furthermore, their application relies heavily on the

block Toeplitz structure and cannot be used for solving systems as in (6.15′) where each entry of the block Toeplitz matrix is perturbed. On the other hand, the first approach works since we make an adjustment after each iteration and we can still take advantage of the recursive block Toeplitz structure that is "buried" in the coefficient matrix in (6.15′).

6.7. Comparisons and remarks. We now briefly compare different techniques and comment on numerical computations.

(i) Space decomposition. We use two decompositions of the function space H—for smoothing splines, $H = H_0 \oplus H_1 \oplus H_2$, and for interpolating splines, $H = \tilde{H}_0 \oplus \tilde{H}_1$. As commented earlier, we cannot use the first decomposition to compute interpolating splines, since the resulting coefficient matrix of the linear system is not positive definite. However, it can be easily checked that we *can* use the second decomposition to compute the smoothing splines. As a result, similar to Theorem 6.3, we have the following theorem.

THEOREM 6.2′. *Given data points on regular grids, smoothing splines in* (4.12) *can be computed in time* $O(N^2(M + \log N))$.

Therefore, we can use the second decomposition for both smoothing and interpolating splines. We still present the first decomposition for the smoothing splines because it suggests a different and interesting approach by itself, and, furthermore, there is no need to compute t_{k_μ} and $\tilde{\Psi}_\nu$ as in (6.11). More importantly, the resulting coefficient matrix is a recursive block Toeplitz matrix and the other three fast algorithms can also be applied with the same cost as in Theorem 6.2′.

(ii) Preconditioning and numerical stability. The first approach only uses the FFT, which is numerically stable. The multiplicative constant in the FFT is 2, and therefore the multiplicative constant in the algorithm is small. On the other hand, the numerical stability of the other three fast algorithms on block Toeplitz matrices is still not well understood.

In the first approach, at each iteration we multiply a block Toeplitz matrix by a vector. We can take advantage of the structure of the coefficient matrix and precondition the system [9] to significantly accelerate the convergence [16].

(iii) Splines without discontinuities. As special cases, the approaches used here can also be applied to the computation of splines without discontinuities, and we obtain algorithms with a cost $O(N^2(M + \log N))$.

(iv) More on cost. We approximate the size of discontinuities by a low-degree (piecewise) polynomial, and M is the dimension of the linear space of the (piecewise) polynomials. Typically, M is small and can be treated as a constant. More precisely, if $M = 0(\log N)$ where N is typically large, then the cost of computing the splines is $O(N^2 \log N)$.

Also note that in our analysis we treat the dimension d as a constant.

7. A residual discontinuity detector. We assume that we know the location and type of discontinuities when we compute the splines. Usually this is not the case in practice. Given a set of noisy sampled data, to locate the discontinuities is itself a complicated process. Discontinuity detection is an old but still not well-studied problem. A number of approaches have been proposed in statistics literature, such as statistical estimation methods. The problem is studied mostly in image analysis in terms of edge detection. We do not intend to survey the literature here. Interested readers are referred to [13], [32], [24], [7], [41], [28] and the references therein and recent issues of [25], [36], [37] among others.

We propose a discontinuity detector in high dimensions based on a residual analysis.

Given an input function f, we smooth f by convolving it with a kernel (typically bell-shaped) function G : $F = f^*G$, where * is the convolution operator. The difference of the original function and the smoothed function is called the residual: $R = f - F$. If the input function has a discontinuity at $t_0 \in R^d$, then with some assumptions the residual has a zero-crossing exactly at t_0, i.e., the residual R has a sign change across t_0. We study the

correspondence between the discontinuities of the input function f and the zero-crossings of the residual R. We state the results in Theorem 7.1 and leave the proof for the Appendix.

DEFINITION 7.1. *A function $f : R^d \to R$ has a discontinuity at t_0 if* (i) *there is an open ball B, centered at t_0*; (ii) *there is a separating hypersurface, which passes through t_0 and which partitions B into two connected open regions B_+ and B_-*; (iii) *$f(t)$ is continuous in B_+ and B_-, respectively*; *and* (iv) $f(t_0+) - f(t_0-) \neq 0$. *The difference is the size of the discontinuity.*

We use $f(t_0\pm)$ to denote the limit of function f when t is approaching t_0 in regions B_+ and B_-, respectively. More specifically,

$$f(t_0+) = \lim_{t \to t_0, t \in B_+} f(t) \quad \text{and} \quad f(t_0-) = \lim_{t \to t_0, t \in B_-} f(t).$$

THEOREM 7.1. *Suppose that $f(t)$ has a discontinuity of size δ at t_0*

$$\delta = |f(t_0+) - f(t_0-)| > 0. \tag{7.1}$$

Assume that a kernel G supports on a ball D, which is centered at the origin, and that D is sufficiently small such that $D \subseteq B - t_0$, where $-$ denotes a translation of ball B. Let $D_- = D \cap (B_- - t_0)$ and $D_+ = D \cap (B_+ - t_0)$. Furthermore, assume that the kernel function $G(t)$ is continuous, symmetric, positive definite, and normalized:

$$\int_D G(t)dt = 1. \tag{7.2}$$

Let F be the convolution of the input function with the kernel G :

$$F(t) = f^* G(t). \tag{7.3}$$

Then the residual

$$R(t) = f(t) - F(t) \tag{7.4}$$

has a zero-crossing at t_0, i.e., $R(t_0-) \cdot R(t_0+) < 0$, if

$$\delta > \frac{L(t_0)}{\min\left\{\int_{D_-} G(t)dt, \int_{D_+} G(t)dt\right\}}, \tag{7.5}$$

where

$$L(t_0) = \int_{D_-} |f(t+t_0) - f(t_0-)|G(t)dt + \int_{D_+} |f(t+t_0) - f(t_0+)|G(t)dt. \tag{7.6}$$

The condition in (7.5) shows that if the size of the discontinuity is large enough then there is a zero-crossing of the residual, which corresponds to the discontinuity. From a different perspective, for a discontinuity of a fixed size the right side of (7.5) has to be small enough so that the discontinuity can be represented by a zero-crossing in the residual. This implies that the numerator $L(t_0)$ should be small. From (7.6), $L(t_0)$ is an estimation of the (weighted) sum of the deviations of the input function f on both sides of the discontinuity t_0. Therefore, the input function f cannot vary too drastically on each side of the discontinuity, particularly when it is getting close to the discontinuity. On the other hand, the denominator should not be too small. Consider the two extreme cases.

Case (i). $\int_{D_-} G(t)dt \approx \int_{D_+} G(t)dt \approx \frac{1}{2}$. For example, this case occurs when the separating hypersurface is a hyperplane in D. Then the right side of (7.5) obtains a minimum value.

Case (ii). $\int_{D_-} G(t)dt \approx 0$ or $\int_{D_+} G(t)dt \approx 0$. For example, this case occurs when the separating surface forms an angle of two hyperplanes of degree almost zero. Then the right side of (7.5) approaches infinity. This implies that if discontinuities form a smooth hypersurface of low curvature then they are easier to detect and if they form a sharp corner then detection becomes very difficult. This phenomena have been observed in edge detection implementations as a piece of folklore without adequate explanations.

Note that condition (7.5) depends on t_0, where the discontinuities occur. When function f satisfies a Lipschitz condition on both sides of the discontinuity:

$$|f(t + t_0-) - f(t_0-)| \leq L \quad \text{for } t \in D_-$$

and

$$|f(t + t_0+) - f(t_0+)| \leq L \quad \text{for } t \in D_+,$$

which one can easily check using the following corollary.

COROLLARY 7.1. *When function f satisfies a Lipschitz condition on both sides of a discontinuity with a Lipschitz constant $L > 0$, if*

$$\delta > \frac{L}{\min\{\int_{D_-} G(t)dt, \int_{D_+} G(t)dt\}},$$

then the residual in (7.4) *has a zero-crossing at the discontinuity.*

Based on either Theorem 7.1 or Corollary 7.1, a preliminary scheme for discontinuity detection can be proposed, i.e., find all zero-crossing points for the residual function $R(t)$.

8. Conclusion. We study thin plate splines in high dimensions, which preserve and incorporate discontinuities, using a reproducing kernel Hilbert space approach. To cope with the formidable cost of computation, we propose fast algorithms, which are of an order of magnitude faster than conventional methods. For the detection of discontinuities, we discuss a residual detector in high dimensions. To conclude this paper, we mention briefly several issues that remain to be resolved.

We study splines, which incorporate function value discontinuities. We further assume that discontinuity size is well approximated by low-degree piecewise polynomials. In general, when there are derivative discontinuities and their size can be arbitrary functions, the construction of the splines is more involved. We will discuss this in a forthcoming paper.

We derive fast algorithms based on an assumption that data points are on a regular grid. This is typical in applications such as image processing and analysis. For irregularly distributed data points, the coefficient matrices of the corresponding systems are not well structured, and to obtain fast algorithms is challenging.

We discuss briefly a discontinuity detection method from a residual analysis. Discontinuity detection itself is a complicated process and is still an unsolved problem, especially in high dimensions and for derivative discontinuity detection. Most of the discontinuity detectors are in one dimension and usually do not have any theoretical justification. Our method is in high dimensions with a mathematical justification. However, from practical point of view, much work has to be done to make it useful. For instance, it is essentially a high-pass filter [8], and the filter response (residual) could be noisy. This has been observed in early works on implementations of residual analysis in one dimension [10]. Postprocessing such as thresholding is needed for eliminating spurious responses [7], [29].

Appendix: Proof of Theorem 7.1. Without loss of generality, assume that $f(t_0-) < f(t_0+)$. We show that if inequality (7.5) holds then there is a zero-crossing of the residual R at t_0, or equivalently,

$$f(t_0-) < F(t_0) < f(t_0+). \tag{A.1}$$

Denote $L_-(t) = |f(t) - f(t_0-)|$ for $t \in B_-$ and $L_+(t) = |f(t) - f(t_0+)|$ for $t \in B_+$. For $t \in B_+$,

$$f(t_0+) - L_+(t) \le f(t) \le f(t_0+) + L_+(t).$$

Multiplying both sides by $G(t - t_0)$ and integrating over region B_+, we have

$$\begin{aligned} & f(t_0+)\int_{B_+} G(t-t_0)dt - \int_{B_+} L_+(t)G(t-t_0)dt \\ & \le \int_{B_+} f(t)G(t-t_0)dt \\ & \le f(t_0+)\int_{B_+} G(t-t_0)dt + \int_{B_+} L_+(t)G(t-t_0)dt. \end{aligned}$$

Since G supports on D and $D \subseteq B - t_0$, by changing variables, we get

$$\begin{aligned} & f(t_0+)\int_{D_+} G(t)dt - \int_{D_+} L_+(t+t_0)G(t)dt \\ & \le \int_{D_+} f(t+t_0)G(t)dt \\ & \le f(t_0+)\int_{D_+} G(t)dt + \int_{D_+} L_+(t+t_0)G(t)dt. \end{aligned} \tag{A.2}$$

Similarly,

$$\begin{aligned} & f(t_0-)\int_{D_-} G(t)dt - \int_{D_-} L_-(t+t_0)G(t)dt \\ & \le \int_{D_-} f(t+t_0)G(t)dt \\ & \le f(t_0-)\int_{D_-} G(t)dt + \int_{D_-} L_-(t+t_0)G(t)dt. \end{aligned} \tag{A.3}$$

Adding (A.2) and (A.3),

$$\begin{aligned} & f(t_0-)\int_{D_-} G(t)dt + f(t_0+)\int_{D_+} G(t)dt - L(t_0) \\ & \le \int_{D} f(t+t_0)G(t)dt \\ & \le f(t_0-)\int_{D_-} G(t)dt + f(t_0+)\int_{D_+} G(t)dt + L(t_0), \end{aligned} \tag{A.4}$$

where $L(t_0)$ is given in (7.6).

Subtracting $f(t_0-)$ from the first inequality of (A.4), since $\int_{D_-} G(t)dt + \int_{D_+} G(t)dt = 1, \delta = f(t_0+) - f(t_0-)$, and G is symmetric, we have

$$\delta \int_{D_+} G(t)dt - L(t_0) \leq F(t_0) - f(t_0-).$$

Therefore, $F(t_0) > f(t_0-)$ if

$$\delta > \frac{L(t_0)}{\int_{D_+} G(t)dt}. \tag{A.5}$$

Similarly, subtracting $f(t_0+)$ from the second inequality of (A.4), we have $F(t_0) < f(t_0+)$ if

$$\delta > \frac{L(t_0)}{\int_{D_-} G(t)dt}. \tag{A.6}$$

The theorem follows directly from (A.5) and (A.6).

Acknowledgments. We are grateful to the anonymous referee for the insightful and constructive review. We are indebted to Grace Wahba for valuable comments and for bringing to our attention reference [34]. The first author thanks Roland W. Freund for the stimulating discussions.

REFERENCES

[1] R. A. ADAM, *Sobolev Spaces*, Academic Press, New York, 1975.
[2] A. V. AHO, J. E. HOPCROFT, AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.
[3] H. AKAIKE, *Block Toeplitz matrix inversion*, SIAM J. Appl. Math., 24 (1973), pp. 234–241.
[4] N. ARONSZAJN, *Theory of reproducing kernels*, Trans. Amer. Math. Soc., 68 (1950), pp. 337–404.
[5] A. BUJA, T. HASTIE, AND R. TIBSHIRANI, *Linear smoothers and additive models (with discussions)*, Ann. Statist., 17 (1989), pp. 453–555.
[6] J. R. BUNCH, *Stability of methods for solving Toeplitz systems of equations*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 349–364.
[7] J. CANNY, *A computational approach to edge detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8 (1986), pp. 679–698.
[8] K. R. CASTLEMAN, *Digital Image Processing*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979.
[9] R. H. CHAN AND G. STRANG, *Toeplitz equations by conjugate gradients with circulant preconditioner*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 104–119.
[10] M. CHEN, D. LEE, AND T. PAVLIDIS, *Residual analysis for feature detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 13 (1991), pp. 30–40.
[11] W. S. CLEVELAND AND E. GROSSE, *Computational methods for local regression*, Statistics and Computing, 1 (1991).
[12] P. CRAVEN AND G. WAHBA, *Smoothing noisy data with spline functions; Estimating the correct degree of smoothing by the method of generalized cross-validation*, Numer. Math., 31 (1979), pp. 337–403.
[13] L. S. DAVIS, *A survey of edge detection techniques*, Computer Graphics and Image Processing, 4 (1975), pp. 248–270.
[14] C. DE BOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.
[15] J. DUCHON, *Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces*, RAIRO Analyze Numerique, 10 (1976), pp. 5–12.
[16] R. W. FREUND, G. H. GOLUB, AND N. M. NACHTIGAL, *Iterative solution of linear systems*, Acta Numerica (1991), pp. 57–100.
[17] J. H. FRIEDMAN, *Fitting functions to noisy data in high dimensions*, in Computer Science and Statistics: Proc. of the 20th Symposium on the Interface, Washington, DC, 1988, pp. 13–43.
[18] J. H. FRIEDMAN AND W. STUETZLE, *Projection pursuit regression*, J. Amer. Statist. Assoc., 76 (1981), pp. 817–823.
[19] H. A. GAMBER, *Choice of an optimal shape parameter when smoothing noisy data*, Comm. Statist. Theory Methods, A8 14 (1979), pp. 1425–1435.

[20] D. GIRARD AND P. J. LAURENT, *Splines and estimation of nonlinear parameters*, in Mathematical Methods in Computer Aided Geometric Design, T. Lyche and L. L. Schumaker eds., Academic Press, New York, 1989, pp. 273–298.
[21] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, North Oxford Academic, Oxford, 1985.
[22] E. GROSSE, *A catalog of algorithms for approximation*, in Algorithms for Approximation II, J. Mason ed., Chapman & Hall, London, 1989.
[23] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.
[24] B. K. P. HORN, *Robot Vision*, MIT Press, Cambridge, MA, 1986.
[25] *IEEE Trans. Pattern Analysis and Machine Intelligence*, IEEE Computer Society Press, Washington, D.C.
[26] P. J. LAURENT AND F. I. UTRERAS, *Optimal smoothing of noisy data using spline function*, J. Approx. Theory Appl., 3 (1986).
[27] D. LEE, *Fast multiplication of a recursive block Toeplitz matrix by a vector and its application*, J. Complexity, 2 (1986), pp. 295–305.
[28] ———, *Discontinuity detection, classification, and measurement*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 311–341.
[29] D. LEE AND G. W. WASILKOWSKI, *Discontinuity detection and thresholding - a stochastic approach*, Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, Lahaina, Maui, Hawaii, 1991, pp. 208–214.
[30] K. C. LI, *On principal Hessian directions for data visualization and dimension reduction: Another application of Stein's Lemma*, Unpublished manuscript, 1990.
[31] ———, *Sliced inverse regression for dimension reduction (with discussions)*, J. Amer. Statist. Assoc., 86 (1991), pp. 316–342.
[32] D. MARR AND E. C. HILDRETH, *Theory of edge detection*, Proc. Roy. Soc. London Ser. B, 207 (1980), pp. 187–217.
[33] J. MEINQUET, *Surface spline interpolation: Basic theory and computational aspects*, Seminaire de mathematique (nouvelle serie), Rapport n^0 35, October 1983.
[34] F. O'SULLIVAN, *Discretized Laplacian smoothing by Fourier methods*, J. Amer. Statist. Assoc., 86 (1991), pp. 634–642.
[35] C. POTIER, M. B. GUERMAH, AND C. VERCKEN, *Curves fitting using nurbs*, in Curves and Surfaces, P. J. Laurent, A. Le Mehaute, and L. L. Schumaker, eds., Academic Press, New York, 1990.
[36] *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society Press, Los Alamitos, CA.
[37] *Proc. International Conference on Computer Vision*, IEEE Computer Society Press, Washington, D.C.
[38] C. H. REINSCH, *Smoothing by spline functions*, Numer. Math., (10) 1967, pp. 177–183.
[39] J. RISSANEN, *Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with application to factoring positive matrix polynomials*, Math. Comput., 27 (1973), pp. 147–154.
[40] D. G. SCHWARTZ, *An interpolation curve using a spline in tension*, J. Math. Phys., 45 (1966), pp. 312–317.
[41] J. H. SHIAU, *Smoothing spline estimation of functions with discontinuities*, Tech. report #768, Dept. of Statistics, University of Wisconsin, Madison, 1985.
[42] ———, *A note on MSE coverage interval of a partial spline model*, Communications in Statistics, A16 (1987), pp. 1851–1866.
[43] J. H. SHIAU, G. WAHBA, AND D. R. JOHNSON, *Partial spline models for the inclusion of Tropopause and frontal boundary information in otherwise smooth two and three dimensional objective analysis*, Journal of Atmospheric and Oceanic Technology, 3 (1986), pp. 714–725.
[44] D. R. SWEET, *Fast block Toeplitz orthogonalization*, Numer. Math., 58 (1991), pp. 613–629.
[45] J. F. TRAUB AND H. WOZNIAKOWSKI, *A General Theory of Optimal Algorithms*, Academic Press, New York, 1980.
[46] ———, *On the optimal solution of large linear systems*, J. Assoc. Comput. Mach., 31 (1984), pp. 545–559.
[47] C. VERCKEN AND C. POTIER, *Spline fitting numerous noisy data with discontinuities*, in Curves and Surfaces, P. J. Laurent, A. Le Mehaute, and L. L. Schumaker, eds., Academic Press, New York, 1990.
[48] S. N. VOEVODINA, *The solution of systems of equations with block-Toeplitz matrices*, Computational Methods and Programming, 94 (1975).
[49] G. WAHBA, *Cross-validated spline methods for the estimation of multivariate functions from data on functionals*, in Statistics: An Appraisal, Proc. of the 50th Anniversary Conference, Iowa State Statistical Laboratory, H. A. David and H. T. David, eds., 1984.
[50] ———, *Partial and interaction splines for the semiparametric estimation of functions of several variables*, in Computer Science and Statistics: Proc. of the 18th Symposium on the Interface, 1986, pp. 75–80.
[51] G. WAHBA AND J. WENDELBERGER, *Some new mathematical methods for variational objective analysis using splines and cross-validation*, Monthly Weather Review, 108 (1980), pp. 36–57.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1331–1348, November 1994

005

ON THE PARALLEL IMPLEMENTATION OF JACOBI AND KOGBETLIANTZ ALGORITHMS*

JÜRGEN GÖTZE†

Abstract. Modified Jacobi and Kogbetliantz algorithms are derived by combining methods for modifying the orthogonal rotations. These methods are characterized by the use of approximate orthogonal rotations and the factorization of these rotations. The presented new approximations exhibit better properties and require less computational cost than known approximations. Suitable approximations are applied together with factorized rotation schemes in order to gain square root free or square root and division free algorithms. The resulting approximate and factorized rotation schemes are highly suited for parallel implementations. The convergence of the algorithms is analyzed and an application in signal processing is discussed.

Key words. Jacobi algorithm, Kogbetliantz algorithm, parallel algorithms, approximate and factorized rotations, convergence

AMS subject classification. 65F15

1. Introduction.

For a real $m \times n$ $(m \geq n)$ matrix $\mathbf{A}$ the decomposition

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T \qquad (\mathbf{U}^T\mathbf{U} = \mathbf{I},\ \mathbf{V}^T\mathbf{V} = \mathbf{I},\ \Sigma \text{ diagonal})$$

is called the singular value decomposition (SVD) of $\mathbf{A}$. For a symmetric $\mathbf{A}$ $(\mathbf{A}^T = \mathbf{A})$ the corresponding decomposition

$$\mathbf{A} = \mathbf{W}\Lambda\mathbf{W}^T \qquad (\mathbf{W}^T\mathbf{W} = \mathbf{I},\ \Lambda \text{ diagonal})$$

is the eigenvalue decomposition (EVD) of $\mathbf{A}$. The methods of choice for the fast parallel computation of these decompositions are the Kogbetliantz (SVD) and Jacobi (EVD) algorithms [3], [23], since they exhibit a significantly higher degree of parallelism than the **QR**-algorithm.

It is well known [6], [20], [8], [5], [21] that it is advantageous to apply the Kogbetliantz algorithm (KA) to the triangular matrix $\mathbf{R}$ obtained from $\mathbf{A}$ by a preparatory QR-decomposition $\mathbf{A} = \mathbf{QR}$. This triangular Kogbetliantz algorithm (TKA) as well as the Jacobi algorithm (JA) only have to work with triangular matrices, since the upper triangular structure of the initial matrix (upper triangular part $(\mathbf{D} + \mathbf{T})$ of $\mathbf{A} = \mathbf{T}^T + \mathbf{D} + \mathbf{T}$ for the JA; upper triangular matrix $\mathbf{R}$ of the QR-decomposition $\mathbf{A} = \mathbf{QR}$ for the TKA) can be preserved during the algorithms.

According to this upper triangular structure one sweep of the JA and the TKA can be implemented on an upper triangular array of processors with nearest neighbor interconnections [24], [6]. The different parallel implementations of the TKA (JA) are distinguished by an algorithm for evaluating the rotations and by an ordering scheme for the rotations. Here, we are mainly interested in deriving efficient algorithms for evaluating the rotations. For ordering schemes, which enable an efficient parallel implementation on multiprocessor arrays with nearest neighbor interconnections, we refer to [23], [6].

In this paper new JAs and TKAs are derived by combining methods for modifying the evaluation of the orthogonal rotations. These methods are characterized by the use of approximate (but still orthogonal) rotation schemes and the use of factorized rotation schemes for gaining square root free [11], [12] or square root and division free [14] rotations.

*Received by the editors November 25, 1991; accepted for publication (in revised form) September 20, 1993. This work was performed while the author was with the Department of Computer Science, Yale University, New Haven and was funded by a grant of the German National Science Foundation.

†Institute of Network Theory and Circuit Design, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany (jugo@nws.e-technik.tu-muenchen.de).

The possibility of using approximate rotation schemes has already been investigated for the sequential case in [29], [10], [32]. Since in a parallel environment the arithmetic is much more costly than other components (e.g., storage access), Modi and Pryce [25] and Charlier, Vanbegin, and van Dooren [6] have shown that the use of approximate rotations gives worthwhile speedups on parallel computers.

The second method for modifying rotations is the use of factorized rotations [11], [12], [14]. By applying these factorized rotation schemes together with suitable approximations a further speedup can be achieved and the hardware requirements of the processor arrays can significantly be decreased. For example, it is possible to obtain a square root and division free TKA/JA (this is not possible if the factorized rotation scheme is applied to the exact rotations [27]), which is essential for regular processor arrays (e.g., application specific integrated circuit (ASIC)-based processor arrays).

In §2 we only consider the JA, since the results for the JA can easily be extended to the TKA. A brief review of the JA is given and the possibility of using approximate rotation schemes is described. New approximations are presented, which exhibit better properties and require less computational cost than the known approximations in [32], [25], [6]. The factorized rotation schemes for gaining square root free or square root and division free rotations are also reviewed. Then, the approximate rotation scheme and the factorized rotation scheme are combined for obtaining a procedure for the design of square root free or square root and division free algorithms. This section ends with the comparison of the different Jacobi rotations concerning their required operations and their suitability for a parallel implementation.

In §3 it is shown that all the results for the JA can be extended to the TKA. The same approximations can be used and the factorized rotations can be derived in the same way. It is even possible to show that using the same approximation for the TKA as for the JA results in a better overall performance of the TKA compared with the JA.

In §4 the global and the ultimate quadratic convergence of the TKA and the JA with exact/approximate rotations is considered. In §5 it is shown that the approximate and factorized scheme is particularly advantageous for SVD-based subspace tracking algorithms [9], [26]. Section 6 gives some concluding remarks.

2. The Jacobi algorithm.

2.1. Basic algorithm. The Jacobi algorithm works by applying a sequence of orthogonal similarity transformations to the symmetric matrix $\mathbf{A}$: $\mathbf{A}^{(0)} := \mathbf{A}$.

For $k = 0, 1, 2, \ldots,$

$$\mathbf{A}^{(k+1)} = \mathbf{J}_{pq}\mathbf{A}^{(k)}\mathbf{J}_{pq}. \tag{1}$$

We assume throughout this paper that in virtue of a parallel implementation the index pairs (p, q) are chosen in an ordering scheme equivalent to the cyclic-by-row scheme and the rotation $\mathbf{J}_{pq}$ includes the required row (left-sided rotation) and column (right-sided rotation) exchanges (see §3 of [6] for details), i.e.,

$$\mathbf{J}_{pq} = \mathbf{J}_{pq}^T = \begin{bmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & -s & & c & & \\ & & & \ddots & & & \\ & & c & & s & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{bmatrix}, \qquad \begin{aligned} c &= \cos\phi_k, \\ s &= \sin\phi_k. \end{aligned} \tag{2}$$

Since the $\mathbf{A}^{(k)}$ are symmetric, they are completely determined by their upper triangular part, such that only one triangle of $\mathbf{A}^{(k)}$ must be processed and the off-diagonal quantity of $\mathbf{A}^{(k)}$ can be measured by

$$S^{(k)} = \sqrt{\frac{1}{2}\left[\|\mathbf{A}^{(k)}\|_F^2 - \sum_{i=1}^{n}\left(a_{ii}^{(k)}\right)^2\right]}. \tag{3}$$

Since $\mathbf{J}_{pq}$ is an orthogonal transformation, i.e., $\|\mathbf{A}^{(k+1)}\|_F = \|\mathbf{A}^{(k)}\|_F$, and one similarity transformation (1) affects only rows and columns p and q of $\mathbf{A}^{(k)}$, it is easy to verify [12] that

$$\left[S^{(k+1)}\right]^2 = \left[S^{(k)}\right]^2 - \left[\left(a_{pq}^{(k)}\right)^2 - \left(a_{pq}^{(k+1)}\right)^2\right]. \tag{4}$$

Obviously, the maximal reduction of $S^{(k)}$ is obtained if $a_{pq}^{(k+1)} = 0$ after the similarity transformation (1). This can be achieved by the following cosine-sine pair(c, s), which defines the rotation (2):

$$t_{ex} = \tan\phi_k = \frac{\operatorname{sign}(\tau_J)}{|\tau_J| + \sqrt{1+\tau_J^2}} \quad \text{with } \tau_J = \frac{a_{pp}^{(k)} - a_{qq}^{(k)}}{2a_{pq}^{(k)}}, \tag{5}$$

$$c = \frac{1}{\sqrt{1+t_{ex}^2}}, \qquad s = t_{ex}c. \tag{6}$$

In the subsequent sections a rotation $\mathbf{J}_{pq}$ defined by (c, s) of (6) is called an exact Jacobi rotation.

2.2. Approximate rotation schemes. For the reduction of $S^{(k)}$ it is not necessary to compute the exact Jacobi rotation, but it is sufficient to compute (c, s), such that the orthogonality is preserved and

$$|a_{pq}^{(k+1)}| = |d_J||a_{pq}^{(k)}| \quad \text{with } 0 \le |d_J| < 1 \tag{7}$$

holds. Therefore, the reduction of $\left[S^{(k)}\right]^2$ is a factor $(1 - d_J^2)$ less than the reduction by the exact rotation. A Jacobi rotation with an approximate computation of (c, s) is called an approximate Jacobi rotation and is described by $\tilde{\mathbf{J}}_{pq}$.

Since the orthogonality must be preserved for the approximate rotation, it is convenient to establish the approximations for $t = \tan\phi = s/c$ instead of (c, s). Since (1) yields

$$a_{pq}^{(k+1)} = d_J a_{pq}^{(k)} \quad \text{with } d_J = c^2 - s^2 - 2\tau_J cs,$$

we obtain

$$|d_J(t, \tau_J)| = \left|\frac{1 - 2\tau_J t - t^2}{1 + t^2}\right|. \tag{8}$$

The maximal value of $|d_J|$ is a measure for the badness of the approximation. It is easy to verify that the exact Jacobi rotation yields $d_J(t_{ex}, \tau_J) = 0$.

In the following the known approximations (KA) of [32], [25], [6] are discussed and some new approximations (NA) are derived. The formulae for the tangent of the rotation angle of the approximations are summarized in Table 1.

TABLE 1

Formulae for the tangent of the rotation angle and accuracy of the corresponding approximations.

Approximation	Accuracy
$t_{KA1} = \frac{\sigma_J}{1+\lvert\sigma_J\rvert}$	$\lvert d_J\rvert \le 0.21$
$t_{KA2} = \sigma_J$	$\lvert d_J\rvert \le 1$
$t_{KA3} = \frac{\sigma_J}{1+\sigma_J^2}$	$\lvert d_J\rvert \le 1$
$t_{KA4} = \frac{\sigma_J(1+\alpha\lvert\sigma_J\rvert)}{1+\beta\lvert\sigma_J\rvert+\alpha\sigma_J^2}$ $(\beta = 2\alpha = \sqrt{2}+1)$	$\lvert d_J\rvert < 0.25$
$t_{KA5} = \begin{cases} \operatorname{sign}(\sigma_J) & \text{if } \lvert\sigma_J\rvert \ge \frac{2}{1+\sqrt{2}} \\ \frac{4\sigma_J}{4-\sigma_J^2} & \text{if } \lvert\sigma_J\rvert < \frac{2}{1+\sqrt{2}} \end{cases}$	$\lvert d_J\rvert \le 0.6036$
$t_{NA1} = \begin{cases} \frac{\operatorname{sign}(\tau_J)}{1+\lvert\tau_J\rvert+0.5\tau_J^2} & \text{if } \lvert\tau_J\rvert \le 1 \\ \frac{\sigma_J}{1+\sigma_J^2} & \text{if } \lvert\tau_J\rvert > 1\ (\lvert 2\sigma_J\rvert < 1) \end{cases}$	$\lvert d_J\rvert \le 0.035$
$t_{NA2} = \begin{cases} \operatorname{sign}(\sigma_J) & \text{if } \lvert\sigma_J\rvert \ge 1 \\ \sigma_J & \text{if } \lvert\sigma_J\rvert < 1 \end{cases}$	$\lvert d_J\rvert \le 0.5$
$t_{NA3} = \begin{cases} \operatorname{sign}(\sigma_J) & \text{if } \lvert\sigma_J\rvert \ge 1.3982 \\ \frac{\sigma_J}{1+\sigma_J^2} & \text{if } \lvert\sigma_J\rvert < 1.3982 \end{cases}$	$\lvert d_J\rvert \le 0.3576$
$t_{NA4} = \begin{cases} \operatorname{sign}(\sigma_J) & \text{if } \lvert\sigma_J\rvert \ge 2 \\ r\cdot\sigma_J \quad \text{with} \quad r = 1/2 & \text{if } \lvert\sigma_J\rvert \ge 1 \\ \qquad\qquad\ \ r = 2/3 & \text{if } \lvert\sigma_J\rvert \ge 0.5 \\ \qquad\qquad\ \ r = 1 & \text{if } \lvert\sigma_J\rvert < 0.5 \end{cases}$	$\lvert d_J\rvert \le 0.25$
$t_{NA5} = \begin{cases} \operatorname{sign}(\sigma_J) & \text{if } \lvert\sigma_J\rvert \ge 2 \\ \frac{1}{2}\sigma_J & \text{if } \lvert\sigma_J\rvert \ge 1 \\ \frac{\sigma_J}{1+\sigma_J^2} & \text{if } \lvert\sigma_J\rvert < 1 \end{cases}$	$\lvert d_J\rvert \le 0.25$

With $\sigma_J = \frac{1}{2\tau_J}$ the KAs of [6], [25] are given by t_{KA1} (formula 1 of [25], approximation 2 of [6]), t_{KA2} (formula 3 of [25], approximation 1 of [6]), and t_{KA3} (approximation 3 of [6]). KA1 yields $|d_J| < 0.21$. However, t_{KA1} does not converge to t_{ex} for $|\tau_J| \to \infty$, although $d_J(|\tau_J| \to \infty) \to 0$ holds (i.e., ultimate quadratic convergence). t_{KA2} and t_{KA3}converge to t_{ex} for $|\tau_J| \to \infty$ and therefore they yield a faster convergence than t_{KA1} as $|\tau_J|$ increases during the algorithm. However, only $|d_J| \le 1$ can be achieved, because of the bad approximation for $|\tau_J| \to 0$ ($t_{KA2} \to \infty$, $t_{KA3} \to 0$ while $t_{ex} \to 1$).

A way to circumvent these problems is to distinguish between small and large values of $|\tau_J|$. Approximation 3 of [6] can be obtained by using the approximation $\sqrt{1+x^2} \approx 1 + \frac{1}{2}x^2$ according to the Taylor series $\sqrt{1+x^2} = 1 + \frac{1}{2}x^2 - \frac{1}{6}x^4 + \cdots$. However, this approximation is only accurate if $|x| < 1$. Therefore, the approximation is bad for $|2\sigma_J| > 1$. But, if we apply this approximation to

$$t_{ex} = \frac{\operatorname{sign}(\tau_J)}{|\tau_J| + \sqrt{1+\tau_J^2}} = \frac{2\sigma_J}{1+\sqrt{1+4\sigma_J^2}} \tag{9}$$

for $|\tau_J| < 1$ ($\sqrt{1+\tau_J^2} \approx 1 + \frac{1}{2}\tau_J^2$) and $|2\sigma_J| < 1$ ($\sqrt{1+4\sigma_J^2} \approx 1 + 2\sigma_J^2$), respectively, we obtain t_{NA1} for which (8) yields $|d_J(t_{NA1}, \tau_J)| < 0.035$. This NA1 requires one square root less than the exact formula (as well as KA3) and requires a comparison (KA3 requires no comparison but has a much worse $|d_J|_{\max}$).

Further NAs can be derived in a similar way. Reducing the computational cost means an increase of the maximal value of $|d_J|$ but the following NAs still provide the same advantages as NA1 in comparison to the KAs, i.e., $|d_J|_{\max} \ll 1$ and $t_{NA} \to t_{ex}$ for $|\tau_J| \to 0$ *and* $|\tau_J| \to \infty$. The easiest way to fulfill these conditions are the approximations NA2 and NA3. NA3 is slightly better than NA2, since for small $|\sigma_J|$ KA3 is a better approximation than KA2. Therefore, it can be used earlier as $|\sigma_J|$ decreases during the algorithm. Since $|d_J(\text{sign}(\sigma_J), \sigma_J)| = \frac{1}{2|\sigma_J|} = |\tau_J|$ holds, this is also the reason for $|d_J(t_{NA3}, \sigma_J)| < |d_J(t_{NA2}, \sigma_J)|$. Better approximations (smaller $|d_J|_{\max}$) can easily be obtained by inserting more cases in NA2 and NA3. This results in NA4 and NA5, respectively.

Until now we have omitted KA4 (t_{KA4} is formula 2 of [25]) and KA5 (t_{KA5} is given in [32], p. 276) for the following reasons. KA4 requires greater cost and has a greater $|d_J|_{\max}$ than KA1. Therefore, it is not discussed further in [25] and it has already been omitted in [6]. Wilkinson [32] has already used different cases for his approximation KA5 (as well as all NAs). However, it is easy to see that KA5 is a worse approximation and requires greater computational cost than NA2.

In Table 2 the average number of sweeps required by 10 random matrices is shown for all the above mentioned approximations for $n = 10, 20, 30, 40$. For all our numerical examples the algorithms are terminated if $S^{(k)} < 10^{-12} S^{(0)}$. The better performance of the NAs in comparison with the KAs is shown in Fig. 1 (Fig. 1(a): NA2 and NA4 in comparison with KA2 and the exact rotation; Fig. 1(b): NA3 and NA5 in comparison with KA3 and the exact rotation). The NAs require less sweeps and less computational cost than the corresponding KAs.

TABLE 2
Required average number of sweeps for 10 *random matrices per n.*

n	ex.	KA1	KA2	KA3	KA4	KA5	NA1	NA2	NA3	NA4	NA5	√&÷ free NA4	√&÷ free NA5
10	5.9	6.8	7.1	6.5	7.0	7.0	5.9	6.3	6.0	5.9	5.9	6.0	6.1
20	6.4	7.6	9.4	7.4	7.7	8.6	6.4	7.0	6.8	6.8	6.8	6.9	6.9
30	7.0	8.0	9.9	7.7	8.2	8.8	7.0	7.5	7.0	7.2	7.0	7.1	7.0
40	7.2	8.3	9.7	8.5	8.3	9.5	7.1	8.0	7.3	7.5	7.4	7.4	7.3

Furthermore, in the case of clusters of eigenvalues, for which $|\tau_J|$ remains small during the algorithm, it is essential that not only $|d_J|$ becomes small as $|\tau_J|$ increases during the algorithm but that the approximation is also good for small values of $|\tau_J|$. These requirements are fulfilled by the NAs. Therefore, for clusters of eigenvalues the NAs yield much better results than the KAs. As an example in Table 3 the required number of sweeps is shown for the Hilbert matrices of dimension $n = 10, 20, 30, 40$.

TABLE 3
Required number of sweeps for Hilbert matrix of dimension n.

n	Ex.	KA1	KA2	KA3	KA4	KA5	NA1	NA2	NA3	NA4	NA5	√&÷ free NA4	√&÷ free NA5
10	5	8	8	9	8	8	5	6	7	9	7	7	6
20	5	8	7	10	9	8	6	6	7	7	8	8	6
30	5	9	10	13	8	10	6	7	7	9	6	8	7
40	6	8	8	10	10	12	6	7	7	7	7	8	7

2.3. Factorized rotation schemes. Factorizations of Givens rotations [11] have been used for the parallel implementation of the QR-decomposition (QRD) by several authors [1],

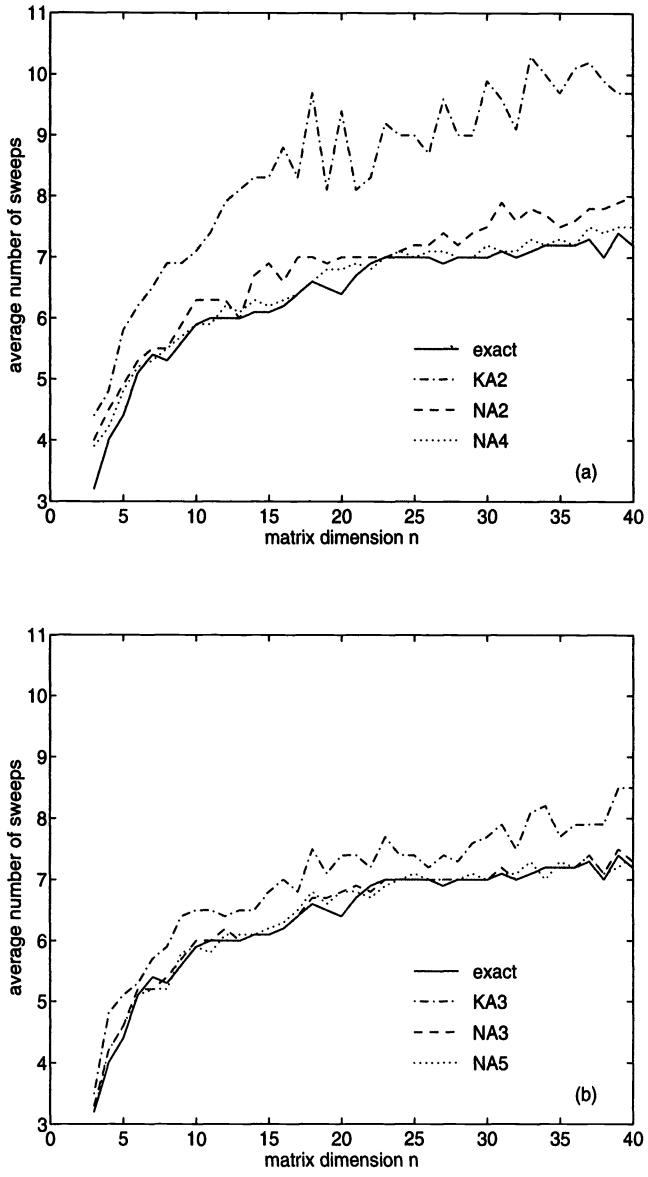


FIG. 1. *Average number of sweeps for* 10 *random matrices per n.*

since the implementation of the square roots can be avoided. Recently, a square root and division free Givens rotation has been presented [14], such that a processor array for the QRD only requires additions and multiplications and a distributed computation of the rotation factors is possible [15]. These factorized rotation schemes can also be applied to orthogonal similarity transformations [27], [16]. For that purpose the matrices $\mathbf{A}^{(k)}$ are factorized as follows:

$$\mathbf{A}^{(k)} = \left[\mathbf{Z}^{(k)}\right]^{-1/2} \mathbf{Y}^{(k)} \left[\mathbf{Z}^{(k)}\right]^{-1/2} \tag{10}$$

with $\mathbf{Z}^{(k)} = \operatorname{diag}(z_i^{(k)})$. With a further diagonal matrix $\mathbf{Z}^{(k+1)} = \operatorname{diag}(z_i^{(k+1)})$ a rotation $\mathbf{K}_{pq}$ (no matter whether it is a Givens or a Jacobi rotation) can also be described in factorized form:

$$\mathbf{K}_{pq} = \left[\mathbf{Z}^{(k)}\right]^{1/2} \mathbf{K}'_{pq} \left[\mathbf{Z}^{(k+1)}\right]^{-1/2}. \tag{11}$$

The application of the factorized rotation scheme to the similarity transformation $\mathbf{A}^{(k+1)} = \mathbf{K}_{pq}^T \cdot \mathbf{A}^{(k)} \cdot \mathbf{K}_{pq}$ leads to the reduction of the square roots by one half if $\mathbf{K}_{pq}$ is an exact Jacobi rotation [27] and to the complete avoidance of the square roots [27] or of the square roots and the divisions [16] if $\mathbf{K}_{pq}$ is a Givens rotation. A Givens rotation is distinguished from a Jacobi rotation, since different formulae are used to compute their rotation parameters. Contrary to Givens rotations, the exact Jacobi rotations cannot be factorized, such that square roots or square roots and divisions can completely be avoided. However, if the factorized rotation schemes are applied to suitable approximations of the exact Jacobi rotation, it is possible to gain square root free or square root and division free Jacobi rotations.

The factorization of $\mathbf{K}_{pq} = \begin{bmatrix} -s & c \\ c & s \end{bmatrix}$ according to (11) yields

(12)

$$\mathbf{K}_{pq} = \begin{bmatrix} z_q^{(k)}\left(1+t^2\right) & \\ & z_p^{(k)}\left(1+t^2\right) \end{bmatrix}^{-1/2} \begin{bmatrix} -t\sqrt{\frac{z_q^{(k)}}{z_p^{(k)}}} & 1 \\ 1 & t\sqrt{\frac{z_p^{(k)}}{z_q^{(k)}}} \end{bmatrix} \begin{bmatrix} z_p^{(k)} & \\ & z_q^{(k)} \end{bmatrix}^{1/2}$$

or

(13)

$$\mathbf{K}_{pq} = \begin{bmatrix} z_q^{(k)}\left(c_t^2+s_t^2\right) & \\ & z_p^{(k)}\left(c_t^2+s_t^2\right) \end{bmatrix}^{-1/2} \begin{bmatrix} -s_t\sqrt{\frac{z_q^{(k)}}{z_p^{(k)}}} & c_t \\ c_t & s_t\sqrt{\frac{z_p^{(k)}}{z_q^{(k)}}} \end{bmatrix} \begin{bmatrix} z_p^{(k)} & \\ & z_q^{(k)} \end{bmatrix}^{1/2}$$

depending on $t = s_t/c_t$ and (c_t, s_t), respectively (note that s_t and c_t are the nominator and the denominator of the formula for the tangent t, i.e., they are different from c and s of the rotation). These factorized rotations result in a square root free or a square root and division free factorized rotation if the matrix elements of the matrices $\mathbf{K}'_{pq}$ and $\mathbf{Z}^{(k+1)}$ can be computed without square roots in (**2.3**) and without square roots and divisions in (**2.3**).

Now, the formula for the tangent of the rotation angle $t = f(a_{ij}^{(k)}) = s_t(a_{ij}^{(k)})/c_t(a_{ij}^{(k)})$ must be adapted to the factorized representation by using $a_{ij}^{(k)} = y_{ij}^{(k)}/\sqrt{z_i^{(k)} z_j^{(k)}}$, which is the number description of the matrix elements $a_{ij}^{(k)}$ according to (10). This yields

$$t = f(y_{ij}^{(k)}, z_i^{(k)}, z_j^{(k)}) = \frac{s_t(y_{ij}^{(k)}, z_i^{(k)}, z_j^{(k)})}{c_t(y_{ij}^{(k)}, z_i^{(k)}, z_j^{(k)})}.$$

If $s_t(y_{ij}^{(k)}, z_i^{(k)}, z_j^{(k)}) = s'_t(y_{ij}^{(k)}, z_i^{(k)}, z_j^{(k)}) \cdot \sqrt{z_i^{(k)} z_j^{(k)}}$ and the computation of $s'_t(y_{ij}^{(k)}, z_i^{(k)}, z_j^{(k)})$ and $c_t(y_{ij}^{(k)}, z_i^{(k)}, z_j^{(k)})$ only requires additions and multiplications, the square root free rotation is defined by

$$\mathbf{K}'_{pq} = \begin{bmatrix} -\frac{s'_t \cdot z_q^{(k)}}{c_t} & 1 \\ 1 & \frac{s'_t \cdot z_p^{(k)}}{c_t} \end{bmatrix}, \qquad \begin{aligned} z_p^{(k+1)} &= z_q^{(k)} \cdot \Delta \\ z_q^{(k+1)} &= z_p^{(k)} \cdot \Delta \end{aligned} \quad \text{with } \Delta = -\det \mathbf{K}'_{pq} \tag{14}$$

and the square root and division free rotation by

$$(15)\qquad \mathbf{K}'_{pq} = \begin{bmatrix} -s'_t \cdot z_q^{(k)} & c_t \\ c_t & s'_t \cdot z_p^{(k)} \end{bmatrix}, \qquad \begin{aligned} z_p^{(k+1)} &= z_q^{(k)} \cdot \Delta \\ z_q^{(k+1)} &= z_p^{(k)} \cdot \Delta \end{aligned} \quad \text{with } \Delta = -\det \mathbf{K}'_{pq}.$$

This provides an easy scheme for deriving square root free or square root and division free rotations from the formula for the tangent of the rotation angle (e.g., for the Givens rotation, where $t = a_{pq}^{(k)}/a_{pp}^{(k)}$, one obtains with $a_{ij}^{(k)} = y_{ij}^{(k)}/\sqrt{z_i^{(k)} z_j^{(k)}}$ that $t = y_{pq}^{(k)}\sqrt{z_p^{(k)} z_q^{(k)}}/y_{pp}^{(k)} z_q^{(k)}$; i.e., $s'_t = y_{pq}^{(k)}$, $c_t = y_{pp}^{(k)} z_q^{(k)}$ for (14) and (15), respectively).

For all factorized schemes there is a problem with the growth of the matrix elements. To overcome this problem the scaling procedure of [14] can be applied to (14) and (15), respectively. This scaling procedure bounds the growth of the elements of the diagonal matrices $\mathbf{Z}^{(k)}$, such that $z_i^{(k)} \in [0.5, 2]$ for all k. It only requires two single shifts and four additions of exponents for scaling $\mathbf{K}'_{pq}$.

2.4. Factorized approximate rotations. Square root free or square root and division free JAs can be derived by combining the approximation and the factorization of the rotations. For the KAs the formula for the tangents of the rotation angles can be described in an appropriate form only for KA2 and KA3. For KA1 a factorization without square roots is not possible, since

$$\begin{aligned} t_{KA1} &= \frac{s_{t_{KA1}}}{c_{t_{KA1}}} = \frac{\operatorname{sign}\left(a_{pp}^{(k)} - a_{qq}^{(k)}\right) \cdot a_{pq}^{(k)}}{\left|a_{pq}^{(k)}\right| + \left|a_{pp}^{(k)} - a_{qq}^{(k)}\right|} \\ &= \frac{\operatorname{sign}\left(y_{pp}^{(k)} z_q^{(k)} - y_{qq}^{(k)} z_q^{(k)}\right) \cdot y_{pq}^{(k)}\sqrt{z_p^{(k)} z_q^{(k)}}}{\left|y_{pq}^{(k)}\right|\sqrt{z_p^{(k)} z_q^{(k)}} + \left|y_{pp}^{(k)} z_q^{(k)} - y_{qq}^{(k)} z_q^{(k)}\right|} = \frac{s'_{t_{KA1}}\sqrt{z_p^{(k)} z_q^{(k)}}}{c_{t_{KA1}}}. \end{aligned}$$

Although $s_{t_{KA1}} = s'_{t_{KA1}}\sqrt{z_p^{(k)} z_q^{(k)}}$ holds, the computation of $c_{t_{KA1}}$ requires a square root operation. For KA2 and KA3 one obtains

$$\begin{aligned} t_{KA2} &= \frac{s_{t_{KA2}}}{c_{t_{KA2}}} = \frac{a_{pq}^{(k)}}{a_{pp}^{(k)} - a_{qq}^{(k)}} \\ &= \frac{y_{pq}^{(k)}\sqrt{z_p^{(k)} z_q^{(k)}}}{y_{pp}^{(k)} z_q^{(k)} - y_{qq}^{(k)} z_q^{(k)}} = \frac{s'_{t_{KA2}}\sqrt{z_p^{(k)} z_q^{(k)}}}{c_{t_{KA2}}}, \end{aligned}$$

$$\begin{aligned} t_{KA3} &= \frac{s_{t_{KA3}}}{c_{t_{KA3}}} = \frac{\left(a_{pp}^{(k)} - a_{qq}^{(k)}\right) \cdot a_{pq}^{(k)}}{\left(a_{pq}^{(k)}\right)^2 + \left(a_{pp}^{(k)} - a_{qq}^{(k)}\right)^2} \\ &= \frac{\left(y_{pp}^{(k)} z_q^{(k)} - y_{qq}^{(k)} z_q^{(k)}\right) \cdot y_{pq}^{(k)}\sqrt{z_p^{(k)} z_q^{(k)}}}{\left(y_{pq}^{(k)}\right)^2 z_p^{(k)} z_q^{(k)} + \left(y_{pp}^{(k)} z_q^{(k)} - y_{qq}^{(k)} z_q^{(k)}\right)^2} = \frac{s'_{t_{KA3}}\sqrt{z_p^{(k)} z_q^{(k)}}}{c_{t_{KA3}}} \end{aligned}$$

such that square root free Jacobi rotations are obtained by (14) and square root and division free Jacobi rotations by (15).

For the NAs different formulae for the tangent are required in dependence of the value of $|\tau_J|$, whereby it must be possible to describe each formula in an appropriate form. The

formula for $|\tau_J| < 1$ of NA1 enables no square root free factorization and therefore NA1 likewise not. All other NAs (NA2–5) are composed of KA2, KA3, and $t_1 = \text{sign}(\sigma_J)$. Therefore, the remaining problem, for deriving square root free or square root and division free Jacobi rotations for the NA2–5 is that $t_1 = \text{sign}(\sigma_J)$ cannot be factorized such that square roots are avoided.

In order to overcome this problem the approximation for the first cases of NA2–5, i.e., $t_1 = \text{sign}(\sigma_J)$, is replaced by the following approximation:

$$t_{f1} = \text{sign}(\sigma_J) \cdot \rho \cdot \sqrt{z_p^{(k)} z_q^{(k)}}. \tag{16}$$

With $s'_{t_{f1}} = \rho \cdot \text{sign}(\sigma_J)$ and $c_{t_{f1}} = 1$ (14) and (15) yield the corresponding factorized rotations. Since the scaling procedure guarantees $z_i^{(k)} \in [0.5, 2]$ for all k, $|t_{f1}| \in [0.5, 2]$ holds. Since this approximation t_{f1} is only used for $|\tau_J| < b$ (e.g., $b = 0.5$ for NA2), the following bounds for the maximal value of $|d_J|$ can be given:

$$\left|d_J\left(t_{f1}, \tau_J\right)\right| \leq 0.6, \qquad (\rho = 1), \tag{17}$$

$$\left|d_J\left(t_{f1}, \tau_J\right)\right| \leq \tfrac{1}{3} \text{ with } \begin{cases} \rho = 0.5 & \text{if } z_p^{(k)} z_q^{(k)} > 2, \\ \rho = \sqrt{2} & \text{if } z_p^{(k)} z_q^{(k)} < 0.5, \\ \rho = 1 & \text{otherwise.} \end{cases} \tag{18}$$

(18) requires additional comparisons for determining ρ. However, these comparisons can be included, if they do not delay the data flow of the processor array (i.e., if the other formulae of the approximation are more complex, e.g., KA3). Furthermore, for NA4 and NA5 the difference between $|d_J|_{\max} = 0.25$ for $|\tau_J| > b$ and $|d_J|_{\max}$ for $|\tau_J| < b$ is reduced ($\frac{1}{3}$ instead of 0.6).

Although $|t_{NAi}| \to 1$ for $|\tau_J| \to 0$ no longer holds, the NA2–5 with t_{f1} instead of t_1 exhibit the same favorable properties as the NA2–5 of §2.2, since $|d_J|_{\max} \ll 1$ still holds and the approximation is still better than the KAs. The last two columns of Tables 2 and 3 show the required number of sweeps of the square root and division free versions of NA4 and NA5. The required number of sweeps is about the same as for the nonfactorized forms of NA4 and NA5.

Finally the comparison for determining the different cases, i.e., $\frac{1}{2|\tau_J|} = |\sigma_J| \geq b$, must also be referred to the factorized scheme

$$|\sigma_J| \geq b \Leftrightarrow \left(y_{pq}^{(k)}\right)^2 z_p^{(k)} z_q^{(k)} \geq b^2 \left(y_{pp}^{(k)} z_q^{(k)} - y_{qq}^{(k)} z_p^{(k)}\right)^2. \tag{19}$$

Thus, the JAs based on the approximate rotation schemes, which are suitable for the factorized rotation schemes (KA2, KA3, NA2–5 with (17) or (18)), can be implemented requiring only $\{+, *, \div\}$ and $\{+, *\}$, respectively. At the end of the JA ($k = k_{\text{end}}$) the results must be refactorized. The eigenvalues are obtained by $\lambda_i = y_{ii}^{(k_{\text{end}})} / z_i^{(k_{\text{end}})}$, which requires n divisions. The eigenvectors require no refactorization. They are merely not normalized but the square of their length is contained in $\mathbf{Z}^{(k_{\text{end}})}$. Since the refactorization is only required at the end of the JA, the corresponding operations can be transferred to the host computer and must not be implemented on the multiprocessor array.

2.5. Comparison. Since the evaluation of the rotations is a more complex task than the pre- and postmultiplication of the rotations, which requires ($8\pm$, $16*$) for the two 2×2 matrix multiplications, the hardware requirements and the data pulse frequency of a regular (e.g.,

systolic) processor array are determined by the computational cost for the evaluation of the rotation.

In Table 4 the required operations for the evaluation of the different Jacobi rotations are shown. For the approximate rotations, which require different cases, the case with the greatest computational cost is indicated and the required operations for this case are specified. Since in general the operations $\{\div, \surd\}$ are (much) more expensive than the operations $\{\pm, *\}$, the number of the former operations is minimized for all rotations. Therefore, instead of computing $t = s_t/c_t$ and then (c, s) with t according to (6), the parameters (c, s) are computed with c_t and s_t as follows:

$$c = \frac{1}{\sqrt{c_t^2 + s_t^2}}, \qquad s = s_t \cdot c \tag{20}$$

(e.g., computing the exact rotation according to (5), (6) requires $(4\pm, 4*, 3\div, 2\surd)$ [25], while using (20) only requires $(4\pm, 5*, 2\div, 2\surd)$). The exponent operations for the scaling of the factorized schemes are not specified. We assume that the processors contain a simple unit for the exponent manipulations (single shifts and additions), which can be executed in parallel to the other operations.

TABLE 4
Required operations of the different Jacobi rotations.

Rotation	Required operations
Exact	$4\pm$ $4*$ $2\div$ $2\surd$
KA1	$3\pm$ $3*$ $2\div$ $1\surd$
KA2	$2\pm$ $3*$ $2\div$ $1\surd$
KA3	$3\pm$ $4*$ $2\div$ $1\surd$
KA4	$5\pm$ $7*$ $2\div$ $1\surd$
KA5	$3\pm$ $3*$ $2\div$
NA1 (case 1)	$4\pm$ $6*$ $2\div$ $1\surd$
NA2 (case 2)	$2\pm$ $3*$ $2\div$ $1\surd$
NA3 (case 2)	$3\pm$ $4*$ $2\div$ $1\surd$
NA4 (case 2)	$2\pm$ $4*$ $2\div$ $1\surd$
NA5 (case 3)	$3\pm$ $4*$ $2\div$ $1\surd$
• $\surd$ free	
KA2,NA2	$2\pm$ $7*$ $1\div$
NA4	$2\pm$ $8*$ $1\div$
KA3,NA3,NA5	$3\pm$ $12*$ $1\div$
• $\surd$ and $\div$ free	
KA2,NA2	$2\pm$ $8*$
NA4	$2\pm$ $9*$
KA3,NA3,NA5	$3\pm$ $12*$

In a parallel implementation the number of sweeps N is predetermined [6], [3], [4]. If the evaluation of an approximate rotation requires t_a time, while the exact evaluation requires t_e time with $t_a = h \cdot t_e$ $(h < 1)$ and if only a few more sweeps are required for the approximate scheme, i.e., $N + N_1$ sweeps instead of N sweeps, then the time consumption of one sweep is reduced from $T_e = (n-1)t_e$ to $T_a = (n-1)t_a$ and the overall time consumption is $T_{AA} = (N + N_1)nt_a = (N + N_1)nht_e$ for the algorithm with approximate rotations instead of $T_{EA} = Nt_e$ for the algorithm with exact rotations. Obviously, $T_{AA} < T_{EA}$ holds if $h(N+N_1) < N$ holds. For random matrices we can set $N_1 = 0$ for NA1, NA4, and NA5, such that $T_{AA} = hT_{EA}$ holds. Even if a few more sweeps are needed with the approximate scheme—e.g., in the case of clusters of eigenvalues ($N_1 = 4$ is the worst case in Table 2 or in the case of simpler approximations ($N_1 \leq 2$ for NA2)—we have $T_{AA} < T_{EA}$ if $h(N + N_1) < N$. For

example, if we set $N = 10$ [3] and $N_1 = 2$ ($N_1 = 4$) $T_{AA} < T_{EA}$ holds if $h < 0.83$ ($h < 0.71$). Therefore, the choice of the computation scheme for the evaluation of the rotation strongly depends on the value of h for the particular computer and the particular implementation (see, e.g., [25]; $h = 0.566$ for KA2 (NA2) on the distributed array processor (DAP)).

The square root and division free schemes only require $\{\pm, *\}$ to be implemented in all processor cells. They enable a very regular (all processors need the same hardware) parallel implementation. Therefore, these algorithms are particularly suited for ASIC-based processor arrays. Furthermore, the operations $\{\pm, *\}$ can be implemented with a time complexity of $O(\log_2 w)$ [30], while the operations $\{\div, \sqrt{}\}$ can only be implemented with a time complexity of $O(w)$, where w is the wordlength of the data. Clearly, NA4 and NA5 are used for a square root and division free implementation, since they exhibit the highest accuracy of the approximation requiring the same amount of $\{\pm, *\}$ as the other worse approximations.

3. Triangular Kogbetliantz algorithm.

3.1. Basic algorithm. The TKA works by applying a sequence of two-sided orthogonal transformations to the upper triangular matrix $\mathbf{R}$ obtained by a preparatory QR-decomposition $\mathbf{A} = \mathbf{QR}$ of the arbitrary $m \times n$ matrix $\mathbf{A}$: $\mathbf{A}^{(0)} := \mathbf{R}$.

For $k = 0, 1, 2, \ldots,$

$$\mathbf{A}^{(k+1)} = \mathbf{Q}_{pq}\mathbf{A}^{(k)}\mathbf{V}_{pq}. \tag{21}$$

Except for the preparatory QRD, the only difference between the TKA and the JA is that the orthogonal $n \times n$ rotation matrices multiplied to the left and right of $\mathbf{A}^{(k)}$ are defined by different rotation parameters. $\mathbf{Q}_{pq}$ is defined by $c_\Phi = \cos\Phi_k$, $s_\Phi = \sin\Phi_k$ and $\mathbf{V}_{pq}$ by $c_\Psi = \cos\Psi_k$, $s_\Psi = \sin\Psi_k$.

In order to preserve the upper triangular structure of $\mathbf{A}^{(k)}$ for all k,

$$a_{qp}^{(k+1)} = -c_\Phi s_\Psi a_{pp}^{(k)} + c_\Phi c_\Psi a_{pq}^{(k)} + s_\Phi c_\Psi a_{qq}^{(k)} := 0 \tag{22}$$

must be met. For a maximal reduction of the off-diagonal quantity

$$a_{pq}^{(k+1)} = -s_\Phi c_\Psi a_{pp}^{(k)} - s_\Phi s_\Psi a_{pq}^{(k)} + c_\Phi s_\Psi a_{qq}^{(k)} = 0 \tag{23}$$

must be fulfilled. From (22) and (23) the exact formulae for computing $\mathbf{Q}_{pq}$, $\mathbf{V}_{pq}$ are obtained in the following.

Case 1 ($t_\Phi = \tan\Phi_k$ is computed first; $t_\Psi = \tan\Psi_k = f(t_\Phi)$).

$$\tau_{K1} = \frac{\left(a_{pp}^{(k)}\right)^2 - \left(a_{qq}^{(k)}\right)^2 + \left(a_{pq}^{(k)}\right)^2}{2a_{qq}^{(k)}a_{pq}^{(k)}}, \tag{24}$$

$$t_\Phi = \frac{\operatorname{sign}(\tau_{K1})}{|\tau_{K1}| + \sqrt{1+\tau_{K1}^2}}, \tag{25}$$

$$t_\Psi = \frac{a_{qq}^{(k)}t_\Phi + a_{pq}^{(k)}}{a_{pp}^{(k)}}. \tag{26}$$

Case 2 (t_Ψ is computed first; $t_\Phi = f(t_\Psi)$).

$$\tau_{K2} = \frac{\left(a_{qq}^{(k)}\right)^2 - \left(a_{pp}^{(k)}\right)^2 + \left(a_{pq}^{(k)}\right)^2}{2a_{pp}^{(k)}a_{pq}^{(k)}}, \tag{27}$$

$$t_\Psi = \frac{-\operatorname{sign}(\tau_{K2})}{|\tau_{K2}| + \sqrt{1+\tau_{K2}^2}}, \tag{28}$$

$$t_\Phi = \frac{a_{pp}^{(k)} t_\Psi - a_{pq}^{(k)}}{a_{qq}^{(k)}}. \tag{29}$$

In both cases (c_Φ, s_Φ) and (c_Ψ, s_Ψ) are obtained from t_Φ and t_Ψ according to (20). With these formulae $a_{ii}^{(k)} > 0$ $(i = 1, \ldots, n)$ holds for all k if $a_{ii}^{(0)} > 0$ $(i = 1, \ldots, n)$ holds, which can always be obtained by the initial QRD [6].

3.2. Approximate rotations. For a reduction of the off-diagonal quantity it is not necessary to satisfy $a_{pq}^{(k+1)} := 0$, but it is sufficient that

$$|a_{pq}^{(k+1)}| = |d_K||a_{pq}^{(k)}| \quad \text{with } 0 \le |d_K| < 1$$

holds. If a corresponding approximate formula is used for t_Φ (25) and t_Ψ (28), one has to distinguish between using Case 1 if $a_{qq}^{(k)} \le a_{pp}^{(k)}$ and using Case 2 if $a_{pp}^{(k)} < a_{qq}^{(k)}$ in order to guarantee $|d_K| < 1$ (for the exact scheme using one case is sufficient, but the test $a_{qq}^{(k)} \le a_{pp}^{(k)}$ is also necessary for a stable computation [20], [6]). Obviously, (25) is the same formula for computing t_Φ with τ_{K1} as (5) for computing t_{ex} with τ_J (the same holds for (28) and τ_{K2}). Therefore, all the approximations of Table 1 can also be applied to the TKA (i.e., to (25) and (28)). For (26) and (29) no further approximations are possible, since these formulae are obtained from (22), which must be met exactly in order to preserve the triangular structure.

We will not repeat the numerical examples of the JA for the TKA, since the following theorem shows that if the same approximation is used for the TKA and the JA, the TKA yields a better approximation than the JA (compare Figs. 6.1 and 6.2 in [6]). In [6] the result of the following theorem is only shown for the special case $|\sigma| = \frac{1}{2|\tau|} \to \infty$ in order to show that $|d_K(|\tau| \to 0)| < 1$ is guaranteed for the TKA, while it is not for the JA (note that the NAs guarantee $|d| \ll 1$ anyway).

THEOREM 3.1. *If the same approximation is used for the* TKA *(for t_Φ and $-t_\Psi$, respectively) and the* JA *(for t_{ex}), then, for $|\tau_{Ki}| = |\tau_J|$ $(i = 1, 2)$,*

$$|d_K(\tau_{Ki})| = |r| \cdot |d_J(\tau_J)|$$

holds with $|r| < 1$ and therefore

$$|d_K|_{\max} < |d_J|_{\max}.$$

Proof. We assume that $|a_{pq}^{(k)}| > 0$, since otherwise no transformation (21) is executed.

Case 1 $(a_{qq}^{(k)} \le a_{pp}^{(k)})$. With $\operatorname{sign}(\tau_{K1}) = \operatorname{sign}(t_\Phi)$ and (26) one obtains from (23)

$$|d_K(\tau_{K1})| = \left|\frac{1 - 2|\tau_{K1}||t_\Phi| - t_\Phi^2}{1 + t_\Phi^2}\right| \cdot \left|\frac{a_{qq}^{(k)} \cdot c_\Psi}{a_{pp}^{(k)} \cdot c_\Phi}\right|.$$

Since the same approximation is used for the TKA and the JA we have for each $|\tau_{K1}| = |\tau_J|$,

$$|d_K(\tau_{K1})| = |r| \cdot |d_J(\tau_J)|,$$

with

$$|r| = \left|\frac{a_{qq}^{(k)} \cdot c_\Psi}{a_{pp}^{(k)} \cdot c_\Phi}\right|.$$

It remains to show that $|r| < 1$ holds, i.e., since all variables of $|r|$ are positive,

$$\left(a_{qq}^{(k)}\right)^2 c_\Psi^2 < \left(a_{pp}^{(k)}\right)^2 c_\Phi^2.$$

With $\cos^2\varphi = (1+\tan^2\varphi)^{-1}$ we have

$$\left(a_{qq}^{(k)}\right)^2 \left(1+t_\Phi^2\right) < \left(a_{pp}^{(k)}\right)^2 \left(1+t_\Psi^2\right)$$

and with (26),

$$\left(a_{pp}^{(k)}\right)^2 - \left(a_{qq}^{(k)}\right)^2 + \left(a_{pq}^{(k)}\right)^2 + 2a_{qq}^{(k)} a_{pq}^{(k)} t_\Phi > 0.$$

Since $a_{qq}^{(k)} > 0$ holds the multiplication of this inequality with $\frac{1}{2a_{qq}^{(k)} a_{pq}^{(k)}}$ yields

$$\begin{aligned} \tau_{K1} + t_\Phi &> 0 \quad \text{if } a_{pq}^{(k)} > 0, \\ \tau_{K1} + t_\Phi &< 0 \quad \text{if } a_{pq}^{(k)} < 0. \end{aligned}$$

Since $\operatorname{sign}(\tau_{K1}) = \operatorname{sign}(t_\Phi)$ these inequalities are equivalent to

$$\begin{aligned} \tau_{K1} &> 0 \quad \text{if } a_{pq}^{(k)} > 0, \\ \tau_{K1} &< 0 \quad \text{if } a_{pq}^{(k)} < 0. \end{aligned}$$

With (24) and $a_{qq}^{(k)} > 0$ this is again equivalent to

$$\left(a_{pp}^{(k)}\right)^2 - \left(a_{qq}^{(k)}\right)^2 + \left(a_{pq}^{(k)}\right)^2 > 0,$$

which is guaranteed by $a_{pp}^{(k)} \geq a_{qq}^{(k)}$. This completes the proof for Case 1.

Case 2 ($a_{pp}^{(k)} < a_{qq}^{(k)}$). With (29) and $\operatorname{sign}(\tau_{K2}) = -\operatorname{sign}(t_\Psi)$ one obtains from (23),

$$|d_K(\tau_{K2})| = \left|\frac{1 - 2|\tau_{K2}|\,|t_\Psi| - t_\Psi^2}{1+t_\Psi^2}\right| \cdot \left|\frac{a_{pp}^{(k)} \cdot c_\Phi}{a_{qq}^{(k)} \cdot c_\Psi}\right|.$$

Therefore, $|d_K(\tau_{K2})| = |r| \cdot |d_J(\tau_J)|$ holds for each $|\tau_{K2}| = |\tau_J|$ with $|r| < 1$ if $(a_{pp}^{(k)})^2 c_\Phi^2 < (a_{qq}^{(k)})^2 c_\Psi^2$. Then, with (29), $a_{pp}^{(k)} > 0$, $\operatorname{sign}(\tau_{K2}) = -\operatorname{sign}(t_\Psi)$, and (27), the rest of the proof is identical to Case 1. □

3.3. Factorized approximate rotations. For the TKA it is also possible to derive square root free and square root and division free rotations. The factorization

$$\mathbf{A}^{(k)} = \left[\mathbf{Z}^{(k)}\right]^{-1/2} \mathbf{Y}^{(k)} \left[\mathbf{X}^{(k)}\right]^{-1/2} \qquad \left(a_{ij}^{(k)} = \frac{y_{ij}^{(k)}}{\sqrt{z_i^{(k)} x_j^{(k)}}}\right) \tag{30}$$

must be used and it must be distinguished between approximating t_Φ (Case 1) or t_Ψ (Case 2). As for the exact formula the corresponding approximate formulae are only distinguished by the sign except for $t_{\Phi_1} = \operatorname{sign}(\tau_{K1})$ and $t_{\Psi_1} = -\operatorname{sign}(\tau_{K2})$ for which $t_{\Phi_{f1}} = \operatorname{sign}(\tau_{K1})\rho\sqrt{z_p^{(k)} z_q^{(k)}}$ and $t_{\Psi_{f1}} = \operatorname{sign}(\tau_{K2})\rho\sqrt{x_p^{(k)} x_q^{(k)}}$ must be used, respectively. The methods of §2 can be applied to the TKA and the square root and division free rotations (square root free rotations in a similar way by using (14) instead of (15) in the sequel) can be obtained as follows.

Case 1. Using (30) and a suitable approximation t_{Φ_A} for the exact t_Φ (25), the approximate formula t_{Φ_A} can be written as

$$t_{\Phi_A} = \frac{s_{\Phi_A}}{c_{\Phi_A}} = \frac{s'_{\Phi_A}\sqrt{z_p^{(k)} z_q^{(k)}}}{c_{\Phi_A}}, \tag{31}$$

whereby the computation of $s'_{\Phi_A} = f(y_{ij}^{(k)}, z_i^{(k)}, x_j^{(k)})$ and $c_{\Phi_A} = f(y_{ij}^{(k)}, z_i^{(k)}, x_j^{(k)})$ $(i, j \in \{p, q\})$ requires only additions and multiplications. Then, (15) yields the rotation matrix

$$\tilde{\mathbf{Q}}'_{pq} = \begin{bmatrix} -s'_{\Phi_A} z_q^{(k)} & c_{\Phi_A} \\ c_{\Phi_A} & s'_{\Phi_A} z_p^{(k)} \end{bmatrix}, \quad \begin{aligned} z_q^{(k+1)} &= z_p^{(k)} \cdot \Delta_\Phi \\ z_p^{(k+1)} &= z_q^{(k)} \cdot \Delta_\Phi \end{aligned} \quad \text{with } \Delta_\Phi = -\det\tilde{\mathbf{Q}}'_{pq}.$$

With (30) and (31) one obtains for t_Ψ (26),

$$t_\Psi = \frac{s_\Psi}{c_\Psi} = \frac{a_{qq}^{(k)} t_{\Phi_A} + a_{pq}^{(k)}}{a_{pp}^{(k)}} = \frac{\left(y_{qq}^{(k)} z_p^{(k)} s'_{\Phi_A} + y_{pq}^{(k)} c_{\Phi_A}\right)\sqrt{x_p^{(k)} x_q^{(k)}}}{y_{pp}^{(k)} x_q^{(k)} c_{\Phi_A}} = \frac{s'_\Psi \sqrt{x_p^{(k)} x_q^{(k)}}}{c_\Psi}$$

and thereby

$$\tilde{\mathbf{V}}'_{pq} = \begin{bmatrix} -s'_\Psi x_q^{(k)} & c_\Psi \\ c_\Psi & s'_\Psi x_p^{(k)} \end{bmatrix}, \quad \begin{aligned} x_q^{(k+1)} &= x_p^{(k)} \cdot \Delta_\Psi \\ x_p^{(k+1)} &= x_q^{(k)} \cdot \Delta_\Psi \end{aligned} \quad \text{with } \Delta_\Psi = -\det\tilde{\mathbf{V}}'_{pq}.$$

Case 2. The factorized rotations can be derived in a similar way exchanging the role of t_Φ and t_Ψ.

3.4. Summary. In summary, it is clear why we started with the JA and extended the results to the TKA afterwards. All results for the JA (approximations, derivation of factorized rotations) can be extended to the TKA. Particularly, the measure for the accuracy of the approximations for the TKA, i.e., $|d_K|$, can be derived from the respective measure for the JA, i.e., $|d_J|$, where $|d_K| = |r| \cdot |d_J|$ holds with $|r| < 1$.

4. Convergence. The global convergence of the JA and the original KA has already been proved by Forsythe and Henrici [10] including approximate rotation schemes. The ultimate quadratic convergence of the JA has been proved by Wilkinson [31], and Paige and van Dooren [28] have extended this result to the original KA. However, for the original KA there are problems with the global and (in the case of clusters of singular values) the ultimate quadratic convergence. These problems are completely removed by the TKA [20], [5]. Furthermore, the TKA is advantageous for a parallel (and a sequential) implementation. The global convergence of the TKA has been proved by Hari and Veselić [20] and by Fernando [8] and the ultimate quadratic convergence by Hari [21] and Charlier and van Dooren [5]. All these proofs of the TKA are established for the exact scheme. Although the convergence of the approximate schemes is used in [25], [6] (based on the results of [10]), no explicit proofs have been published yet (e.g., the proof of Fernando [8] cannot be extended to approximate schemes).

In [13] the global and the ultimate quadratic convergence of the TKA and the JA is proved for the approximate schemes. The known results for the exact schemes are obtained as special cases ($d = 0$). Furthermore, for the first time the presented proofs hold for the TKA as well as for the JA. This is achieved by assuming that during the TKA and the JA the matrices remain "essential triangular" [21]. Although for the JA $\mathbf{A}^{(k)^T} = \mathbf{A}^{(k)}$ actually holds, we only use one essential triangular part, such that the JA also proceeds as shown for the TKA in Fig. 2.1.1 of [20].

In the following the theorems concerning the global and the ultimate quadratic convergence are given and the methods used for the proofs of these theorems are briefly outlined. For the details of these proofs see [13].

THEOREM 4.1 (Global convergence). *If* $0 \leq |d| < 1$ *holds throughout the* TKA *and the* JA, *then the column (and row) cyclic schemes of these algorithms are always convergent.*

Proof. The proof of this theorem is modeled after the proof of the global convergence of the TKA with exact rotations [20]. We prove that $\left[S^{(k+M)}\right]^2 \leq c_n \left[S^{(k)}\right]^2$ $(M = n\frac{n-1}{2})$ with $c_n < 1$ holds for all the different algorithms presented in this paper, i.e., for the TKA/JA with exact/approximate rotations (see [13]). □

THEOREM 4.2 (Ultimate quadratic convergence). *Let*

$$|\sigma_i - \sigma_j| \geq 2\delta, \quad (i \neq j) \text{ (distinct singular-/eigenvalues)} \tag{32}$$

and suppose we have reached the stage r, where

$$S^{(r)} < \frac{\delta}{4}, \tag{33}$$

then for some $(k > r)$,

$$S^{(k+M)} \leq \frac{\gamma}{\sqrt{1 - d_{\max}^2}} \cdot \frac{\left[S^{(k)}\right]^2}{\delta} + O\left(\left[S^{(k)}\right]^3\right) \tag{34}$$

holds where $\gamma = \frac{7}{6}$ *for the* TKA *and* $\gamma = 1$ *for the* JA.

Proof. The proof of this theorem (see [13]) follows the proofs of Paige and van Dooren [28] (TKA) and Wilkinson [31] (JA) $(x_i := a_{pq}^{(i)})$ with $|x_{i+1}| = |d_i x_i|$ (instead of $x_{i+1} = 0$). □

In the case of clusters of singular values (eigenvalues) the proofs for the approximate schemes are completely identical to the proofs for the exact scheme of [22], [5], [21]. Only the bound (34) for the case of distinct singular/eigenvalues must be used instead of the bound for the exact scheme.

5. Application in signal processing. By computing the singular value decomposition (SVD) of a data matrix it is possible to extract the signal and noise subspaces of the data. The knowledge of these subspaces is essential in many application fields, e.g., DOA-estimation (ESPRIT, MUSIC) and state-space system identification. In practice, where the subspaces are usually time varying, it is even more important to be able to track these subspaces. Therefore, in recent years different subspace tracking algorithms have been proposed. These algorithms are based on the rank revealing QRD [2], the Lanczos algorithm [7], or the SVD-updating algorithm [9], [26], where the SVD-updating algorithms are favorable in virtue of a parallel implementation. The SVD-updating algorithm works as follows: At timestep k a new measurement vector is incorporated in the upper triangular matrix $\mathbf{R}_{[k-1]}$ by a QRD update resulting in $\mathbf{R}_{[k]}$. Then, the SVD of $\mathbf{R}_{[k]}$ is computed using the TKA.

The main result of [9], [26] is that it is usually sufficient to execute only one sweep [9], or even only a fraction of one sweep [26] (only the elements of the first subdiagonal of $\mathbf{R}_{[k]}$ are annihilated) of the TKA after each QRD update. Since the approximation of the rotations is a marginal approximation compared to the approximation of the TKA (i.e., executing only one sweep or a fraction of one sweep), the use of approximate rotations yields essentially the same results with respect to tracking capability as using exact rotation.

To illustrate this the example of Ferzali and Proakis [9] is used, i.e., 400 sample points of a signal composed of 3 sinusoids:

$$s(k) = 2\cos(2\pi \cdot 0.15k) + 2\cos(2\pi \cdot 0.2k) + 2\cos(2\pi \cdot f \cdot k) + u(k)$$

whereby f jumps from 0.35 to 0.45 at $k = 200$ and $u(k)$ is Gaussian white noise (SNR = 20 dB). Data vectors with dimension $m = 7$ are formed from the samples. Figure 2 shows $e(k) = \|\mathbf{sv}(k) - \mathbf{sve}(k)\|_2$ versus k. $\mathbf{sve}(k)$ is a vector that contains the exact singular values of the $k \times m$ data matrix $\mathbf{X}(k)$ available at time k. $\mathbf{sv}(k)$ is the vector that contains the singular values as obtained by the SVD-updating algorithm. Here, only the first subdiagonal of $\mathbf{R}_{[k]}$ is annihilated by the TKA, i.e., only a part of a complete sweep of the TKA is executed after each QRD update. Obviously, tracking the singular values using the approximation, KA2 (Fig. 2(b)) works as well as using exact rotations (Fig. 2(a)).

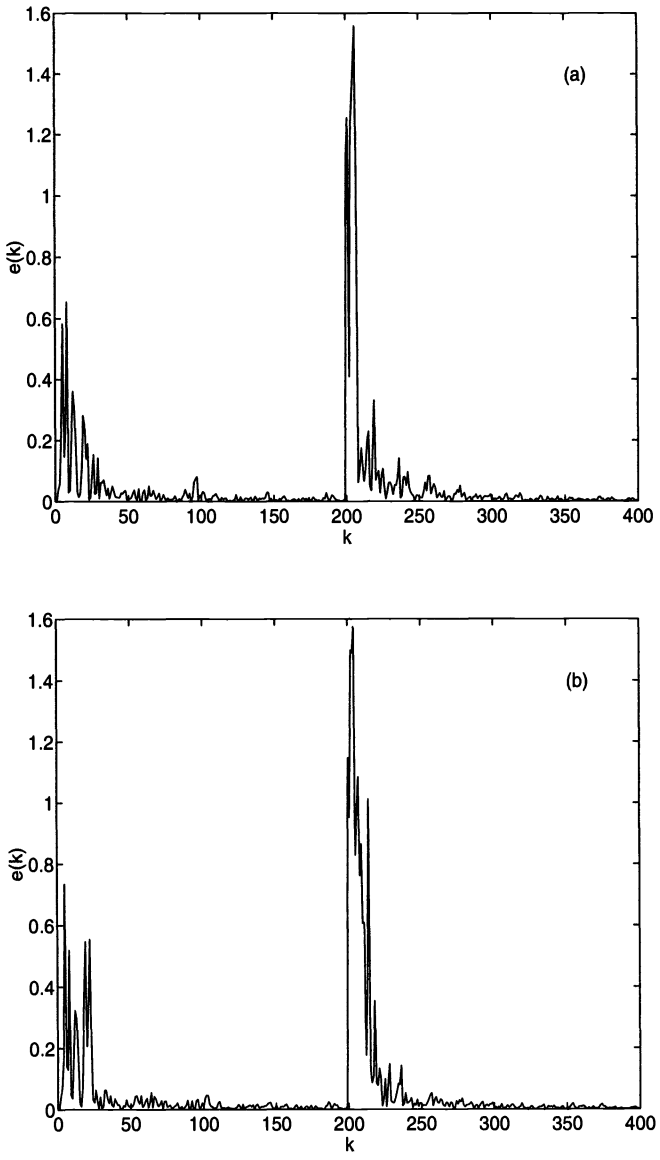


FIG. 2. *e(k) versus k using* (a) *exact rotations and* (b) *the approximate rotation* KA2.

Therefore, an application specific processor array for subspace tracking can be derived, which exhibits all advantages of the approximate factorized schemes, while the slight increase concerning the required number of sweeps is marginal (in many practical applications even a fraction of a sweep is sufficient).

6. Concluding remarks. In this paper new JAs and TKAs were presented by combining methods for modifying orthogonal rotations. Suitable approximate rotations were described in factorized form in order to gain square root free or square root and division free rotations for the JA and the TKA.

In summary, we recommend the use of approximate schemes for the TKA as well as the JA, since

- the new approximations exhibit better properties and require less computational cost than the known approximations. Even for the JA the required number of sweeps of NA4 and NA5 is about the same as for the exact scheme. The TKA always performs better than the JA, since $|d_K|_{\max} < |d_J|_{\max}$ (Theorem 3.1).
- the convergence problems (for the JA) of KA2 and KA3 are removed by the new approximations ($|d| \ll 1$ is guaranteed).
- the use of approximate rotations enables the derivation of square root free or square root and division free factorized rotations, which is not possible for the exact scheme.
- for a parallel implementation the speedups gained by using approximate rotations and by using factorized rotations can be combined. Furthermore, the hardware requirements are significantly decreased (e.g., the square root and division free versions of the JA and the TKA can efficiently be implemented on an array of application specific processors, e.g., transputers or DSPs, which contain only $\{\pm, *\}$ as fast hardware).
- for a VLSI-implementation of a systolic array executing the TKA and the JA only $\{\pm, *\}$ must be implemented in all processor cells (ASICs). In contrary to the operations $\{\div, \sqrt{}\}$, which require $O(w)$ time, the operations $\{\pm, *\}$ can be implemented needing $O(\log_2 w)$ time [30]. This enables a further speedup for the square root and division free algorithms. The approximate rotation scheme can also be combined efficiently with the CORDIC scheme in order to obtain an efficient VLSI-implementation based on the CORDIC algorithm [17], [18].
- the approximate factorized schemes are particularly useful in signal processing applications (e.g., subspace tracking), since the approximate rotations perform as well as exact rotations for many practical applications.

Acknowledgment. Thanks are to I. Ipsen for giving me the opportunity to spend a wonderful year at Yale. Thanks are also to A. Bojanczyk, who brought the use of approximate rotations [6], [25] to my attention.

REFERENCES

[1] J. L. Barlow and I. C. F. Ipsen, *Scaled Givens rotations for the solution of linear least squares problems on systolic arrays*, SIAM J. Sci. Stat. Comput., 5 (1987), pp. 716–733.

[2] C. H. Bischof, *On updating signal subspaces*, IEEE Trans. on Signal Proc., 40 (1992), pp. 96–105.

[3] R. P. Brent and F. T. Luk, *The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 69–84.

[4] R. P. Brent, F. T. Luk, and C. van Loan, *Computation of the singular value decomposition using mesh connected processors*, J. VLSI Computer Systems, 3 (1985), pp. 242–270.

[5] J.-P. Charlier and P. van Dooren, *On Kogbetliantz's SVD algorithm in the presence of clusters*, Linear Algebra Appl., 95 (1987), pp. 135–160.

[6] J.-P. Charlier, M. Vanbegin, and P. van Dooren, *On efficient implementations of Kogbetliantz's algorithm for computing the singular value decomposition*, Numer. Math., 52 (1988), pp. 279–300.

[7] P. COMON AND G. H. GOLUB, *Tracking a few singular values and vectors in signal processing*, Proc. IEEE, 78 (1990), pp. 1327–1343.

[8] K. V. FERNANDO, *Linear convergence of the row cyclic Jacobi and Kogbetliantz methods*, Numer. Math., 56 (1989), pp. 73–91.

[9] W. FERZALI AND J. G. PROAKIS, *Adaptive SVD algorithm for covariance matrix eigenstructure computation*, in Proc. IEEE Int. Conf. on Acoust., Speech and Signal Processing, Toronto, 1990, pp. 2615–2618.

[10] G. E. FORSYTHE AND P. HENRICI, *The cyclic Jacobi method for computing the principal values of a complex matrix*, Trans. Amer. Math. Soc., 94 (1960), pp. 1–23.

[11] W. M. GENTLEMAN, *Least squares computations by Givens transformations without square roots*, J. Inst. Maths. Appl., 12 (1973), pp. 329–336.

[12] G. H. GOLUB AND C. VAN LOAN, *Matrix Computations*, Second ed., The John Hopkins University Press, Baltimore, MD, 1989.

[13] J. GÖTZE, *On the parallel implementation of Jacobi's and Kogbetliantz's algorithm*, Research Report RR-879, Dept. of Computer Science, Yale University, New Haven, CT, 1991.

[14] J. GÖTZE AND U. SCHWIEGELSHOHN, *A square root and division free Givens rotation for solving least squares problems on systolic arrays*, SIAM J. Sci. Stat. Comput., 4 (1991), pp. 800–807.

[15] ———, *VLSI-suited orthogonal solution of systems of linear equations*, J. Parallel & Distributed Comput., 11 (1991), pp. 276–283.

[16] J. GÖTZE, M. ALI, AND U. SCHWIEGELSHOHN, *Efficient orthogonal matrix decompositions for digital signal processing*, in Proc. URSI Int. Symp. on Signals, Systems and Electronics, Erlangen, 1989, pp. 803–806.

[17] J. GÖTZE, S. PAUL, AND M. SAUER, *An efficient Jacobi-like algorithm for parallel eigenvalue computation*, IEEE Trans. on Computers, 42 (1993), pp. 1058–1065.

[18] ———, *A CORDIC-Based Jacobi-like Algorithm for Eigenvalue Computation*, in Proc. IEEE Int. Conf. on Acoust., Speech and Signal Processing, Vol. 3, Minneapolis, 1993, pp. 296–299.

[19] E. R. HANSEN, *On cyclic Jacobi methods*, J. Soc. Indust. Appl. Math., 2 (1963), pp. 448–459.

[20] V. HARI AND K. VESELIĆ, *On Jacobi methods for singular value decomposition*, SIAM J. Sci. Stat. Comput., 5 (1987), pp. 741–754.

[21] V. HARI, *On the quadratic convergence of the serial singular value decomposition Jacobi methods for triangular matrices*, SIAM J. Sci. Stat. Comput., 6 (1989), pp. 1076–1096.

[22] H. P. M. VAN KEMPEN, *On the quadratic convergence of the special cyclic Jacobi method*, Numer. Math., 9 (1966), pp. 19–22.

[23] F. T. LUK AND H. PARK, *On parallel Jacobi orderings*, SIAM J. Sci. Stat. Comput., 1 (1989), pp. 18–26.

[24] F. T. LUK, *A triangular processor array for computing singular values*, Linear Algebra Appl., 77 (1986), pp. 259–273.

[25] J. J. MODI AND J. D. PRYCE, *Efficient implementation of Jacobi's diagonalization method on the DAP*, Numer. Math., 46 (1985), pp. 443–454.

[26] M. MOONEN, P. VAN DOOREN, AND J. VANDEWALLE, *A singular value decomposition updating algorithm for subspace tracking*, SIAM J. Matrix Anal. Appl., 4 (1992), pp. 1015–1038.

[27] W. RATH, *Fast Givens rotations for orthogonal similarity transformations*, Numer. Math., 40 (1982), pp. 47–56.

[28] C. C. PAIGE AND P. VAN DOOREN, *On the quadratic convergence of Kogbetliantz's algorithm for computing the singular value decomposition*, Linear Algebra Appl., 77 (1986), pp. 301–313.

[29] D. A. POPE AND C. TOMPKINS, *Maximizing functions of rotations – Experiments concerning the speed of diagonalization of symmetric matrices using Jacobi's method*, J. ACM, 4 (1957), pp. 459–466.

[30] J. VUILLEMIN, *A very fast multiplication algorithm for VLSI implementation*, Integration: the VLSI J., 1 (1983), pp. 39–52.

[31] J. H. WILKINSON, *Note on the quadratic convergence of the cyclic Jacobi process*, Numer. Math., 6 (1962), pp. 296–300.

[32] ———, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1349–1374, November 1994

006

EFFICIENT VARIANTS OF THE VERTEX SPACE DOMAIN DECOMPOSITION ALGORITHM*

TONY F. CHAN†, TAREK P. MATHEW‡, AND JIAN-PING SHAO†

Abstract. Several variants of the vertex space algorithm of Smith for two-dimensional elliptic problems are described. The vertex space algorithm is a domain decomposition method based on nonoverlapping subregions, in which the reduced Schur complement system on the interface is solved using a generalized block Jacobi-type preconditioner, with the blocks corresponding to the vertex space, edges, and a coarse grid. Two kinds of approximations are considered for the edge and vertex space subblocks, one based on Fourier approximation, and another based on an algebraic *probing* technique in which sparse approximations to these subblocks are computed. Our motivation is to improve the efficiency of the algorithm without sacrificing the optimal convergence rate. Numerical and theoretical results on the performance of these algorithms, including variants of an algorithm of Bramble, Pasciak, and Schatz are presented.

Key words. domain decomposition, Schur complement, interface probe, block Jacobi preconditioner, elliptic equations, preconditioners, vertex spaces

AMS subject classifications. 65N20, 65F10

1. Introduction.

Domain decomposition methods often provide suitable techniques for solving large linear systems of equations arising from discretizations of partial differential equations. In particular, these methods can be advantageous for the efficient and localized treatment of irregular geometries, discontinuous coefficients, local grid refinement, boundary layers, and coupling between equations of different type; see, for instance, [20], [6], [7], and [21].

In this paper, we primarily focus on the development of efficient versions of two divide and conquer-type domain decomposition algorithms based on nonoverlapping subregions for solving self-adjoint elliptic problems in two dimensions. The algorithms we describe are variants of the vertex space (VS) algorithm proposed by Smith [28] and Nepomnyaschikh [26], and an algorithm of Bramble, Pasciak, and Schatz (BPS) [4]. In both cases, a block Jacobi-type preconditioner is used to solve the reduced Schur complement system on the interface. The blocks in the BPS algorithm correspond to the nodes on the edges separating the subdomains and to the collection of vertices of the subregions, while in the VS algorithm additional overlapping blocks, centered about each vertex consisting of nodes on the interface close to the vertex, are included to account for coupling amongst the nonoverlapping blocks.

In order to implement the original version of the VS preconditioner [28], the subblocks of the Schur complement, which are dense matrices, need to be computed and inverted using direct methods. It can, however, be easily shown that if these subblocks are replaced by spectrally equivalent approximations, then the rate of convergence of these algorithms remains asymptotically the same. In order to reduce overhead cost, we therefore focus on constructing approximations that are inexpensive to construct and inexpensive to invert.

Two kinds of approximations will be considered, one based on Fourier approximations of the interface operators, and another based on sparse algebraic approximation of the interface

*Received by the editors February 4, 1992; accepted for publication (in revised form) September 20, 1993. This work was supported in part by Department of Energy contract DE-FG03-87ER25037, Army Research Office contract DAAL03-91-G-150, National Science Foundation grant FDP NSF ASC 9003002, and Office of Naval Research contract N00014-90-J-1695.

†Department of Mathematics, University of California at Los Angeles, Los Angeles, California 90024 (chan@math.ucla.edu, shao@ms.uky.edu).

‡Department of Mathematics, University of Wyoming, Laramie, Wyoming 82071-3036 (mathew@ledaig.uwyo.edu). This research was performed while the author was at the Department of Mathematics, University of California at Los Angeles.

operators by a *probing* technique. The Fourier-based approximations can be shown to be spectrally equivalent with respect to mesh size variations. However, their performance can be sensitive to the coefficients. On the other hand, the probing-based algorithms adapt well to the coefficients, but can be sensitive to mesh size variations.

In §2, we describe the elliptic problem and the Schur complement system on the interface. In §3, we describe the original versions of the BPS and VS preconditioners for the Schur complement on the interface. In §4, we describe the two variants, one based on Fourier approximations, and the other based on the *probing* technique. In §5, we present numerical results comparing the rates of convergence of the various preconditioners.

2. An elliptic problem and its many subdomain decompositions. Here we describe the block structure obtained when a self-adjoint elliptic problem is discretized on a domain Ω partitioned into many nonoverlapping subdomains Ω_i with an interface B separating the subdomains. A reduced Schur complement system is derived for the unknowns on the interface. Some properties of this Schur complement system and an iterative procedure for solving the elliptic problem are described.

2.1. Block partition of elliptic problem. We consider the following second-order self-adjoint elliptic problem on a polygonal domain $\Omega \in R^2$:

$$\begin{aligned} -\nabla \cdot (a(x, y)\nabla u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where $a(x, y) \in R^{2\times 2}$ is a symmetric, uniformly positive definite matrix function having $L^\infty(\Omega)$ entries, and $f \in L^2(\Omega)$.

We assume that the domain Ω is partitioned into N nonoverlapping subdomains $\Omega, \ldots, \Omega_N$ of diameter H, which form the elements of a *quasi-uniform* coarse grid triangulation τ^H; see Fig. 1. We also assume that the subdomains Ω_i are refined to produce a fine grid *quasi-uniform* triangulation τ^h having elements of diameter h. Corresponding to the coarse grid and fine grid triangulations, we discretize (1) either by using piecewise linear finite elements (see [14]), or by using finite difference methods (see [30]), resulting in a symmetric positive definite linear system

$$A_h u_h = f_h \tag{2}$$

on the fine grid and

$$A_H u_H = f_H \tag{3}$$

on the coarse grid.

Let I denote the union of the interiors of the subdomains, and let B denote the interface separating the subdomains:

$$I = \cup_i \Omega_i, \qquad B \equiv (\cup_i \partial\Omega_i) - \partial\Omega.$$

Then, grouping the unknowns in the interior of the subdomains in the vector u_I and the unknowns on the interface B in the vector u_B, we obtain a reordering of the fine grid problem:

$$\begin{bmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{bmatrix} \begin{bmatrix} u_I \\ u_B \end{bmatrix} = \begin{bmatrix} f_I \\ f_B \end{bmatrix}. \tag{4}$$

Here A_{II} corresponds to the coupling between nodes in the interior of the subdomains. For most lower order discretizations, including five-point discretizations, the interior nodes in Ω_i

are coupled only to the nodes on the interface B, and not to adjacent subdomains. In such cases, $A_{II} \equiv \text{blockdiag}(A_{11}, \ldots, A_{NN})$ is a block diagonal matrix.

Eliminating interior unknowns u_I, we obtain u_I in terms of u_B:

$$u_I = A_{II}^{-1}(f_I - A_{IB}u_B), \tag{5}$$

and substituting this in the second block row of (4) yields an equation for u_B:

$$Su_B = f_B - A_{IB}^T A_{II}^{-1} f_I, \tag{6}$$

where $S = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB}$ is referred to as the Schur complement or interface matrix. Some properties of the Schur complement will be discussed in §2.3. First, we will outline a procedure for solving (4).

2.2. Iterative solution of the block partitioned system. System (4) can be solved as follows. First, problem (6) is solved for u_B and then (5) is solved for u_I. If direct methods are used to solve (6) then S needs to be computed explicitly, and this can be expensive in general (though this is standard practice in the substructuring methods used to solve linear elasticity problems), since it involves computing the action of A_{II}^{-1} on all the columns of A_{IB}. This can be implemented more efficiently through subassembly (see [28]), requiring only as many solves on each Ω_i as there are unknowns on $\partial\Omega_i \cap B$. Even if the matrix S has been assembled, it is often preferable to solve (6) by an iterative method, since direct methods to solve (6) require significant memory storage and computational complexity.

Due to the expense of computing S and solving (6) by direct methods, we consider solving (6) by a preconditioned iterative method such as the conjugate gradient method (see [23]), without the explicit construction of S. In this case only matrix-vector products with S are required, and each such matrix-vector product requires the solution of one problem on each subdomain Ω_i. The Schur complement, however, is ill conditioned with $\kappa(S) \approx O(h^{-1})$ (see [2], [4]), and therefore requires a preconditioner M; the construction of efficient preconditioners M for S will be the main focus of this paper.

First, we note that the procedure to solve the linear system (4) by solving the reduced Schur complement system (6) corresponds to a block LU factorization-based solution:

$$A = LU = \begin{bmatrix} A_{II} & 0 \\ A_{IB}^T & I \end{bmatrix} \begin{bmatrix} I & A_{II}^{-1}A_{IB} \\ 0 & S \end{bmatrix} \tag{7}$$

for $S = A_{BB} - A_{IB}^T A_{II}^{-1} A_{IB}$. Thus

$$A^{-1} = \begin{bmatrix} I & -A_{II}^{-1}A_{IB}S^{-1} \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} A_{II}^{-1} & 0 \\ -A_{IB}^T A_{II}^{-1} & I \end{bmatrix},$$

and backsolving requires solving two systems with coefficient matrices A_{II} and one system with coefficient matrix S, which will be done using a preconditioned conjugate gradient method. We note heuristically that it is often possible to construct a global preconditioner $\tilde{A}$ for A by carefully replacing A_{II} by preconditioner $\tilde{A}_{II}$, and by replacing S by preconditioner M. In this case the inverse of the global preconditioner $\tilde{A}$ has the form

$$\tilde{A}^{-1} = \begin{bmatrix} I & -\tilde{A}_{II}^{-1}A_{IB}M^{-1} \\ 0 & M^{-1} \end{bmatrix} \begin{bmatrix} \tilde{A}_{II}^{-1} & 0 \\ -A_{IB}^T \tilde{A}_{II}^{-1} & I \end{bmatrix}.$$

Approximations to the submatrices $\tilde{A}_{ii}$ can be obtained, for instance, by replacing it either with a carefully scaled version of the Laplacian, or by other preconditioners, such as *ILU* (see [9]). Some care must be exercised in the choice of $\tilde{A}_{ii}$, as it has been shown that it can lead to poor performance if the approximations of $\tilde{A}_{ii}$ are not suitably scaled (see Börgers [3]).

2.3. Some properties of the Schur complement S. The Schur complement matrix S is a discrete approximation to a Steklov–Poincaré operator (see [1]), which enforces *transmission boundary* conditions on the interface B. In the continuous problem, these transmission boundary conditions correspond to the requirement that the solution u be continuous across the interface and that the flux $\vec{n} \cdot (a(x, y)\nabla u)$ also be continuous across the interface. In the discrete case, the action of the Schur complement on a grid function u_B on B is the same as the action of the discrete operator A_h on the *discrete harmonic extension* of u_B into the subdomains; More specifically, let $E^h u_B$ denote the *discrete harmonic extension* on B to the interior of the subdomains:

$$E^h u_B \equiv \left[-A_{II}^{-1} A_{IB} u_B, u_B\right]^T; \tag{8}$$

then

$$\begin{bmatrix} A_{II} & A_{IB} \\ A_{IB}^T & A_{BB} \end{bmatrix} \begin{bmatrix} -A_{II}^{-1} A_{IB} u_B \\ u_B \end{bmatrix} = \begin{bmatrix} 0 \\ S u_B \end{bmatrix}.$$

Thus, if R_B denotes the pointwise restriction of nodal values of a grid function onto the nodes on B, then $S u_B = R_B A_h E^h u_B$. In addition,

$$x_B^T S x_B = (E x_B)^T A_h (E x_B). \tag{9}$$

This property shows the positive definiteness of the Schur complement. In addition to S being positive definite, it is an M-matrix when A_h is an M-matrix, i.e., $S_{ij} \leq 0$ for $i \neq j$ and $(S^{-1})_{ij} \geq 0$ for all i, j (see [30] and [12]).

Remark. For finite element discretizations, let $A^{(i)}$ denote the stiffness matrix obtained by integrating the bilinear form on Ω_i, i.e., the discretization of the Neumann problem on Ω_i. For finite difference methods, let $A^{(i)}$ correspond to the discretization with discontinuous coefficients, which is $a(x, y)$ in Ω_i and zero outside Ω_i. Then, if $x^{(i)}$ denotes the vector x restricted to Ω_i, the energy $x^T A x$ can be partitioned as

$$x^T A_h x = \sum_{i=1}^{N} x^{(i)^T} A^{(i)} x^{(i)}, \tag{10}$$

and correspondingly, the Schur complement S can be partitioned:

$$x_B^T S x_B = \sum_{i=1}^{N} x_B^{(i)^T} S^{(i)} x_B^{(i)}, \tag{11}$$

where

$$S^{(i)} = A_{BB}^{(i)} - A_{IB}^{(i)T} A_{II}^{(i)^{-1}} A_{IB}^{(i)} \quad \text{with } A^{(i)} = \begin{bmatrix} A_{II}^{(i)} & A_{IB}^{(i)} \\ A_{IB}^{(i)^T} & A_{BB}^{(i)} \end{bmatrix}. \tag{12}$$

Each $S^{(i)}$ is a map of the Dirichlet value u_B to the *normal derivative* on $\partial\Omega_i \cap B$ of the discrete harmonic extension $E^h u_B$, and this is not a local operator, i.e., the matrix $S^{(i)}$ is dense on $\partial\Omega_i \cap B$ (see [2]). In the two-subdomain case, $S = S^{(1)} + S^{(2)}$ is thus a map of the Dirichlet value u_B to the jump in the normal derivative on B of the discrete harmonic extension $E^h u_B$, which corresponds to a discrete approximation of the transmission boundary condition. In the two-dimensional case, the entries of S decay as $|S_{ij}| = 0(1/|i - j|^2)$ (see Golub and Mayers [22]), and preconditioners for S have been studied extensively (see [2], [8], [5], [19], and

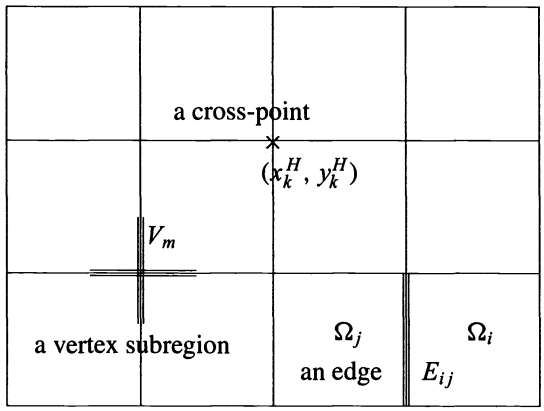


FIG. 1. *The vertex space partitioning of the interface.*

[10]). The important properties of the two-subdomain Schur complement is that its entries decay away from its main diagonal, and that it is uniformly spectrally equivalent to the square root of the Laplace operator on B, as the mesh size goes to zero. Due to this connection, it can be shown that its condition number grows as $\kappa(S) = O(\frac{1}{h})$ (see [2]). Applications of both of these properties will be discussed in §§4.1 and 4.2.

3. The BPS and VS preconditioners for S. We will describe two preconditioners for S in this section, one introduced by Bramble, Pasciak, and Schatz [4], and another, the VS preconditioner, introduced by Smith [28] (related to a preconditioner proposed by Nepomnyaschikh [26]). Both of these can be interpreted as generalized block Jacobi-type preconditioners for (6) with overlapping blocks and involving residual correction on a coarse grid. Variants of these preconditioners will be discussed in §4.

3.1. Notations for a partition of the interface B. In the case of many subdomains, the interface B can be partitioned as a union of edges E_{ij} and cross-points V (see Fig. 1):

$$B = \cup_{ij} E_{ij} \cup V,$$

where E_{ij} denotes the edge separating subdomains Ω_i and Ω_j, and V denotes the collection of cross-points (vertices (x_k^H, y_k^H) of the subdomains). Note that the edges E_{ij} are assumed not to include their endpoints.

For each edge E_{ij} we define $R_{E_{ij}}$ as the pointwise restriction of nodal values to E_{ij}, i.e., if g_B is a grid function defined on B, and if E_{ij} contains n_{ij} interior nodes, then its restriction $R_{E_{ij}} g_B$ is a vector with n_{ij} components defined on E_{ij} by

$$R_{E_{ij}} g_B = g_B \quad \text{on } E_{ij}.$$

Its transpose $R_{E_{ij}}^T$ extends the grid functions in E_{ij} by zero to the rest of B:

$$R_{E_{ij}}^T g_{E_{ij}} = \begin{cases} g_{E_{ij}} & \text{on } E_{ij}, \\ 0 & \text{on } B - E_{ij}. \end{cases}$$

Similarly, we define R_V as the pointwise restriction map onto the cross-points; if g_B is a grid function on B, and if there are n_V cross-points on B, then $R_V g_B$ is a vector with n_V components defined by

$$R_V g_B = g_B \quad \text{on } V.$$

Its transpose R_V^T thus corresponds to extension by zero of nodal values in V to B:

$$R_V^T g_V = \begin{cases} g_V & \text{on } V, \\ 0 & \text{on } B - V. \end{cases}$$

3.2. The BPS preconditioner. In order to motivate the construction of the BPS preconditioner, we first define a block Jacobi preconditioner M_J consisting of diagonal blocks of the Schur complement S in the following block partitioning of the interface B. Let us suppose there are n edges E_{ij} with some ordering $E_1, \ldots, E_n$. If the unknowns on each edge E_i are grouped together in u_{E_i}, and if the unknowns on the cross-points are grouped in u_V, then S has the following block partitioning corresponding to $(u_{E_1}, \ldots, u_{E_n}, u_V)$:

$$S = \begin{bmatrix} S_{E_1} & \cdots & S_{E_1 E_n} & S_{E_1 V} \\ \vdots & \cdots & \vdots & \vdots \\ S_{E_1 E_n}^T & \cdots & S_{E_n} & S_{E_n V} \\ S_{E_1 V}^T & \cdots & S_{E_n V}^T & S_V \end{bmatrix}.$$

Here, $S_{E_i E_j} \equiv R_{E_i} S R_{E_j}^T$ denotes the coupling in S between nodes on E_i and E_j, and $S_{E_i V} \equiv R_{E_i} S R_V^T$ denotes the coupling in S between nodes on E_i and V. Note that edges E_i and E_j will be coupled in S only if they are part of the boundary of a common subdomain Ω_k. This can be seen by using the relation between Schur complement and discrete harmonic extension, since, for instance, the discrete harmonic extension of a grid function on edge E_i is nonzero only in the subdomains for which E_i is part of its boundary. S is thus a block sparse matrix and corresponding to each edge E_{ij}, the submatrix $S_{E_{ij}}$ is identical to the two-subdomain Schur complement on interface E_{ij} separating Ω_i and Ω_j. The submatrix S_V which corresponds to coupling in S between cross-points is a sparse matrix in which the diagonal entries are dominant, since the cross-points are weakly coupled in S. In the case of five-point discretizations on rectangular subdomains, S_V is diagonal since the corner nodes (cross-points) of rectangular domains do not influence the solution in the interior.

For this block partition of S, we define the action of the inverse of the block Jacobi preconditioner M_J:

$$M_J^{-1} g_B = \sum_{\text{edges } ij} R_{E_{ij}}^T S_{E_{ij}}^{-1} R_{E_{ij}} g_B + R_V^T S_V^{-1} R_V g_B. \tag{13}$$

This block Jacobi preconditioned system can be shown to have a condition number satisfying

$$c_1 \leq \frac{\lambda_{\max}(M_J^{-1} S)}{\lambda_{\min}(M_J^{-1} S)} \leq c_2 H^{-2}(1 + \log^2(H/h)),$$

where c_1 and c_2 are independent of H and h (see [4] and [31]). This indicates that as $H \to 0$, i.e., as the number of subdomains increases, the rate of convergence deteriorates. This can be attributed to the absence of global communication of information amongst all the edges in the preconditioning step.

The original version of the BPS algorithm [4] involves two changes to this block Jacobi preconditioner. One is that the submatrices $S_{E_{ij}}$ are replaced by Fourier-based approximations $\tilde{S}_{E_{ij}}$, which will be described in §4. The second change is to incorporate global coupling in order to obtain a rate of convergence which does not deteriorate as the number of subdomains

is increased. In order to do this, the cross-points correction term $R_V^T S_V^{-1} R_V$ in (13) is replaced by a coarse grid correction term $R_H^T A_H^{-1} R_H$ as in two-level multigrid methods (involving weighted restriction and interpolation maps R_H and R_H^T, respectively). These are defined below. Let $\phi_{k,H}$ denote the kth coarse grid piecewise linear finite element basis function

$$\phi_{k,H}(x_l^H, y_l^H) = \begin{cases} 1 & \text{if } l = k, \\ 0 & \text{if } l \neq k, \end{cases}$$

where (x_l^H, y_l^H) is the lth cross-point. Then

$$(R_H f_B)(x_k^H, y_k^H) \equiv \sum_{(x_j, y_j) \in B} \phi_{k,H}(x_j^H, y_j^H) f_B(x_j^H, y_j^H).$$

Its transpose R_H^T thus denotes linear interpolation of the nodal values on the endpoints of edges E_{ij}:

$$\left(R_H^T g_V\right)(x, y) \equiv \sum_k g_V(x_k^H, y_k^H)\phi_{k,H}(x, y), \qquad (x, y) \in B.$$

With these changes, the BPS preconditioner can be defined:

$$M_{\text{BPS}}^{-1} f_B = \sum_{\text{edges } i,j} R_{E_{ij}}^T \tilde{S}_{E_{ij}}^{-1} R_{E_{ij}} f_B + R_H^T A_H^{-1} R_H f_B.$$

These changes improve the condition number over that of the block Jacobi version.

THEOREM 3.1. *The* BPS *preconditioner satisfies*

$$\frac{\lambda_{\max}(M_{\text{BPS}}^{-1} S)}{\lambda_{\min}(M_{\text{BPS}}^{-1} S)} \leq c_2(1 + \log^2(H/h)),$$

where c_2 is independent of H and h.

Proof. See [4] and [31] for the proof. □

Remark. It can be easily verified that for five-point discretizations of the Laplacian, the coarse grid Schur complement matrix $S_H \equiv R_H S_h R_H^T$ is equal to the coarse grid discretization $A_H = R_H^T A_h R_H$, since piecewise linear interpolation results in grid functions that are discrete harmonic on the subdomains. In the case of more general coefficients, it can be shown that A_H and S_H are spectrally equivalent with respect to coarse grid size H.

3.3. The vertex space algorithm of Smith and Nepomnyaschikh. The logarithmic growth in the condition number of the BPS preconditioner can be attributed to the neglect of coupling between adjacent edges of B. The VS preconditioner of Smith [28] and Nepomnyaschikh [26] incorporates some coupling between adjacent edges through the use of certain overlapping blocks of S corresponding to nodes on certain *vertex regions* V_k, which will be defined, and it leads to a condition number independent of mesh parameters.

Let V_k denote the portion of B within a distance of βH from (x_k^H, y_k^H) for some positive fraction $0 < \beta < 1$ (see Fig. 1). We refer to each V_k as a vertex region or vertex space. We define the corresponding pointwise restriction map R_{V_k} to be

$$R_{V_k} g_B = g_B \quad \text{on } V_k.$$

Its transpose $R_{V_k}^T$ thus corresponds to extension by zero outside V_k:

$$R_{V_k}^T g_{V_k} = \begin{cases} g_{V_k} & \text{on } V_k, \\ 0 & \text{on } B - V_k. \end{cases}$$

Corresponding to each vertex region V_k, the submatrix S_{V_k} is defined by $S_{V_k} = R_{V_k} S R_{V_k}^T$. The action of the inverse of the vertex space preconditioner M_{VS} involves the inversion of these new overlapping blocks in addition to the blocks used in the BPS preconditioner:

$$M_{VS}^{-1} f_B = R_H^T A_H^{-1} R_H f_B + \sum_{E_{ij}} R_{E_{ij}}^T (S_{E_{ij}})^{-1} R_{E_{ij}} f_B + \sum_{V_k} R_{V_k}^T (S_{V_k})^{-1} R_{V_k} f_B. \tag{14}$$

The following result is proved in [28] and [26].

THEOREM 3.2. *Suppose the overlap of the vertex regions V_k is βH; then*

$$\frac{\lambda_{\max}(M_{\mathrm{VS}}^{-1} S)}{\lambda_{\min}(M_{\mathrm{VS}}^{-1} S)} \leq C(1 + H/\beta),$$

where C is independent of H, h, and β.

Remark. Other bounds are available for the condition number of the vertex space preconditioned system:

$$\frac{\lambda_{\max}(M_{\mathrm{VS}}^{-1} S)}{\lambda_{\min}(M_{\mathrm{VS}}^{-1} S)} \leq c_1(1 + c_2(1/\beta)\log(H/h)),$$

where c_1 and c_2 are independent of H, h, β, and the coefficients $a(x, y)$ provided the coefficients are constant in each subdomain Ω_i (see [28], [32], and [17]).

4. Two variants of the vertex space method. An important consideration in the implementation of the algorithms is the expense of computing the edge and vertex matrices $S_{E_{ij}}$ and S_{V_k}, respectively, and the cost of solving the subproblems using direct methods. If there are n_i nodes on each $\partial\Omega_i \cap B$, then computing all the submatrices $S_{E_{ij}}$ and S_{V_k} would require solving n_i problems on each Ω_i, and this increases as the mesh size h is reduced. If n_{ij} is the number of nodes on E_{ij}, the cost of using direct methods to solve edge problems is $O(n_{ij}^2)$ once the Cholesky factorizations have been determined (see [28] and [29]), since the edge submatrices $S_{E_{ij}}$ are dense. n_{ij} increases as the mesh size h is reduced.

This expense can be significantly reduced if the exact edge and vertex matrices are replaced by approximations which can be computed at significantly less cost, and which can be inverted at less cost. If these approximations are spectrally equivalent to the exact submatrices, then the overall preconditioner would remain spectrally equivalent to the exact VS preconditioner, and the number of iterations required to solve (6) would remain independent of h; see Theorem 4.3.

In this section, we describe two variants of the vertex space and BPS algorithms in which the exact edge and vertex matrices are replaced by approximations. One variant is based on Fourier approximations of both the edge and vertex matrices, while the other variant is based on sparse algebraic approximation of both these matrices using a *probing* technique. Combinations of Fourier and probe approximations are also possible, but will not be considered here for simplicity (e.g., see [11]).

4.1. Fourier approximations. Fourier-based approximations of the edge and vertex matrices are constructed based on the property that, restricted to simple curves (curves which do not intersect themselves), the Schur complement is spectrally equivalent to the square root of the Laplace operator on it, and this has been studied extensively (see [2], [22], [8], [5], [19], and [10]).

4.1.1. Fourier edge approximations. First, we consider Fourier approximations of the edge matrices $S_{E_{ij}}$. Let edge E_{ij} separate Ω_i and Ω_j. Since the submatrix $S_{E_{ij}}$ is identical to the

two-subdomain Schur complement on E_{ij}, standard preconditioners for the two-subdomain case can be applied (see [2], [22], [8], [5], [19], and [10]).

Let J denote the discrete Laplacian on a uniform grid containing n_{ij} interior nodes with mesh size $h = 1/(n_{ij}+1)$:

$$-h^2\frac{d^2}{dx^2} \approx J \equiv \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$

Then, $J^{1/2}$ is uniformly spectrally equivalent to $S_{E_{ij}}$ as the mesh size h is varied (see [2]). Since the discrete Laplacian is diagonalized by the sine transform, $J = W\Lambda W$, where

$$W_{ij} = \sqrt{2h}\sin(ij\pi h),$$

and $\Lambda = \operatorname{diag}(\lambda_i)$ with $\lambda_i = 4\sin^2(\frac{i\pi h}{2})$, it follows that $J^{1/2} = W\Lambda^{1/2}W$. By using Fast Sine Transforms, it is possible to compute the action of the inverse of $J^{1/2}$ in $O(n_{ij}\log(n_{ij}))$ flops.

The Fourier-based preconditioners M considered here are all based on the sine transform W, but vary with the choice of eigenvalues:

$$M = W\operatorname{diag}(\mu_k)W.$$

The eigenvalues μ_k are chosen to better approximate the eigenvalues of the exact Schur complement $S_{E_{ij}}$. In the model case of Laplace's equation on $\Omega_i \cup \Omega_j$ with rectangular subdomains $\Omega_i = [0,1]\times[0,l_i]$ and $\Omega_j = [0,1]\times[-l_j,0]$, where m_i and m_j are positive integers with $l_i = (m_i+1)h$ and $l_j = (m_j+1)h$, the eigendecomposition of the Schur complement is known exactly. These exact eigenvalues are given below in M_{Chan}, along with the eigenvalues of three other preconditioners:

(15)

Dryja preconditioner M_{D} (see [16]): $\mu_k = \lambda_k^{1/2}$,

Golub–Mayers preconditioner M_{GM} (see [22]): $\mu_k = \sqrt{\lambda_k + \frac{1}{4}\lambda_k^2}$,

BPS preconditioner M_{BPS} (see [4]): $\mu_k = \sqrt{\lambda_k(1-\frac{\lambda_k}{6})}$,

Chan preconditioner M_{Chan} (see [8]):

$$\mu_k = \left(\frac{1-\gamma_k^{m_i+1}}{1+\gamma_k^{m_i+1}} + \frac{1-\gamma_k^{m_j+1}}{1+\gamma_k^{m_j+1}}\right) \times\sqrt{\lambda_k + \tfrac{1}{4}\lambda_k^2},$$

$$\text{where } \gamma_k = \frac{1+\frac{\lambda_k}{2}-\sqrt{\lambda_k+\frac{\lambda_k^2}{4}}}{1+\frac{\lambda_k}{2}+\sqrt{\lambda_k+\frac{\lambda_k^2}{4}}}.$$

We have the following result.

LEMMA 4.1. *Let M denote the Dryja, Golub–Mayers,* BPS, *or Chan preconditioners for $S_{E_{ij}}$. Then*

$$\frac{\lambda_{\max}(M^{-1}S_{E_{ij}})}{\lambda_{\min}(M^{-1}S_{E_{ij}})} \le C_1,$$

where C_1 is independent of h. For $S_{E_{ij}}$ corresponding to Laplace's equation on the model domain $\Omega_i \cup \Omega_j$ with rectangular subdomains $\Omega_i = [0, 1] \times [0, l_i]$ and $\Omega_j = [0, 1] \times [-l_j, 0]$, the condition number of the Dryja, Golub–Mayers, and BPS *preconditioners satisfy*

$$\frac{\lambda_{\max}(M^{-1}S_{E_{ij}})}{\lambda_{\min}(M^{-1}S_{E_{ij}})} \leq C_2\left(1 + \frac{1}{l_i} + \frac{1}{l_j}\right),$$

where C_2 is independent of h, l_i, and l_j, while the condition number of the Chan preconditioner satisfies

$$\frac{\lambda_{\max}(M_{\text{Chan}}^{-1}S_{E_{ij}})}{\lambda_{\min}(M_{\text{Chan}}^{-1}S_{E_{ij}})} \leq C_2.$$

Proof. See Bjorstad and Widlund [2] and Chan [8] for the proof. □

The Fourier preconditioners described so far do not depend on the coefficients $a(x, y)$ of the elliptic problem, and thus the rate of convergence can be sensitive to the coefficients (see [12]). In order to incorporate some information about the coefficients, we scale the Fourier preconditioners by a scaling matrix. In the original BPS algorithm [4], a scalar coefficient α_{ij} representing the average of the eigenvalues of $a(x, y)$ at a point in Ω_i and a point in Ω_j was used as scaling on each edge E_{ij}. Here we use a diagonal matrix D_{ij} as scaling, where D_{ij} denotes the diagonal of A_h restricted to E_{ij}, and we define the diagonally scaled Fourier preconditioners by

$$\tilde{S}_{E_{ij}}^F \equiv D_{ij}^{1/2} W \operatorname{diag}(\mu_k) W D_{ij}^{1/2}. \tag{16}$$

For most applications to isotropic coefficients, these diagonally scaled Fourier preconditioners perform well.

4.1.2. Fourier vertex space approximation. Next, we describe approximations of the vertex space matrices S_{V_k} based on Fourier techniques. For the case of the discrete Laplacian, it is possible to express the eigendecomposition of S_{V_k} for cross-shaped vertex regions in terms of sine transforms, thereby enabling the use of fast transforms to invert S_{V_k} (see [26]). However, it is not easily generalized to the case of varying coefficients, and instead we construct approximations to the vertex matrices by using a direct sum of smaller matrices that will be described in the following.

We will describe the procedure for the model geometry of Fig. 2. Let u_{V_k} be a grid function on B which is zero outside the vertex region V_k, i.e., zero on $B - V_k$. Then, by the property of the Schur complement (11), we obtain

$$u_{V_k}^T S_{V_k} u_{V_k} = \sum_{i=1}^{4} u_{V_k}^T S^{(i)} u_{V_k}, \tag{17}$$

where $S^{(i)}$ is the component of the Schur complement originating from Ω_i, as described in (12). For $i = 1, 2, 3, 4$, let L_i^k denote the L-shaped segment $V_k \cap \partial\Omega_i$, and further let $R_{L_i^k}$ denote the pointwise restriction onto L_i^k. Then, as in the case for the edges, $(R_{L_i^k} u_B)^T S^{(i)} (R_{L_i^k} u_B)$ is spectrally equivalent to $(R_{L_i^k} u_B)^T M_i^k (R_{L_i^k} u_B)$ where M_i^k is any of the unscaled Fourier approximations to the square root of the Laplacian on L_i^k (see (15)). Let D_i^k denote the diagonal of $A^{(i)}$ restricted to L_i^k. Then, by including the effects of coefficients, we define the following scaled Fourier-based preconditioner for S_{V_k}:

$$\tilde{S}_{V_k}^F \equiv \sum_{i=1}^{4} R_{L_i^k}^T (D_i^k)^{1/2} M_i^k (D_i^k)^{1/2} R_{L_i^k}. \tag{18}$$

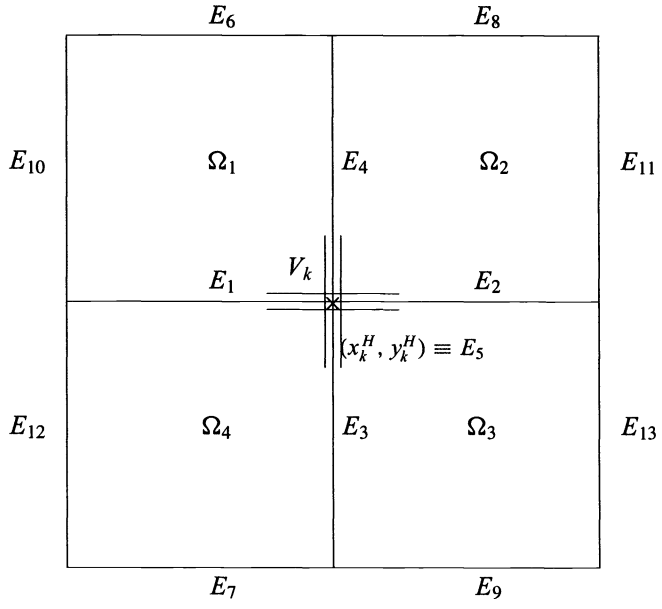


FIG. 2. *Numbering of edges.*

For most applications we considered, it was sufficient to choose the number of nodes on the vertex regions V_k to be small, say five or nine, and so the matrices $\tilde{S}_{V_k}^F$ can be computed at little expense, and can be inverted inexpensively by direct methods.

THEOREM 4.2. *The matrices $\tilde{S}_{V_k}^F$ are spectrally equivalent to S_{V_k}, i.e., there exist constants c_0, c_1 independent of h such that*

$$c_0 \leq \frac{\lambda_{\max}\left((\tilde{S}_{V_k}^F)^{-1} S_{V_k}\right)}{\lambda_{\min}\left((\tilde{S}_{V_k}^F)^{-1} S_{V_k}\right)} \leq c_1.$$

Proof. The proof follows trivially by application of the standard result (see [2] and [5]) that the square root of the Laplacian on a simple edge such as L_i^k is spectrally equivalent to the local Schur complement, i.e., there exist constants $c_0^{(i)}, c_1^{(i)}$ independent of h such that

$$c_0^{(i)} \leq \frac{x_{V_k}^T S_{V_k}^{(i)} x_{V_k}}{x_{V_k}^T M_i^k x_{V_k}} \leq c_1^{(i)}.$$

Similar bounds hold when M_i^k is replaced by $(D_i^k)^{1/2} M_i^k (D_i^k)^{1/2}$, with suitably modified constants $c_0^{(i)}, c_1^{(i)}$, since the entries of D_i^k can be bounded in terms of the upper and lower bounds for $a(x, y)$ in the neighborhood of L_i^k, independent of h. From this the result follows immediately, since

$$\min_i\{c_0^{(i)}\} \leq \frac{\sum_{i=1}^4 x_{V_k}^T S_{V_k}^{(i)} x_{V_k}}{\sum_{i=1}^4 x_{V_k}^T M_i^k x_{V_k}} \leq \max_i\{c_1^{(i)}\},$$

for the suitably modified coefficients $c_0^{(i)}$ and $c_1^{(i)}$. □

4.1.3. Fourier-based preconditioner. Based on the approximations $\tilde{S}^F_{E_{ij}}$ and $\tilde{S}^F_{V_k}$, we define the Fourier vertex space (FVS) preconditioner by

$$M^{-1}_{\text{FVS}} \equiv R_H^T A_H^{-1} R_H + \sum_{ij} R^T_{E_{ij}} (\tilde{S}^F_{E_{ij}})^{-1} R_{E_{ij}} + \sum_k R^T_{V_k} (\tilde{S}^F_{V_k})^{-1} R_{V_k}, \tag{19}$$

and the Fourier BPS (FBPS) preconditioner by (see [4])

$$M^{-1}_{\text{FBPS}} \equiv R_H^T A_H^{-1} R_H + \sum_{ij} R^T_{E_{ij}} (\tilde{S}^F_{E_{ij}})^{-1} R_{E_{ij}}. \tag{20}$$

Note that the Fourier edge approximations $\tilde{S}^F_{E_{ij}}$ can be inverted in $O(n_{ij} \log(n_{ij}))$ flops, using the Fast Sine Transform. Direct methods can be used to solve the Fourier vertex problems $\tilde{S}^F_{V_k}$. The coarse grid matrix problem A_H can be solved using either direct or iterative methods.

Remark. In the original BPS preconditioner [4], the edge approximations were chosen to be

$$\tilde{S}_{E_{ij}} = \alpha_{ij} W \operatorname{diag}(\mu_k) W,$$

where α_{ij} is the average of the eigenvalues of $a(x, y)$ at a point from Ω_i and a point from Ω_j, and $\mu_k = \sqrt{\lambda_k(1 - \lambda_k/6)}$. This differs from the version described in (20) because of the scaling matrix D_i^k.

THEOREM 4.3. *The Fourier preconditioner M_{FVS} satisfies:*

$$c_0 \le \frac{\lambda_{\max}(M^{-1}_{\text{FVS}} S)}{\lambda_{\min}(M^{-1}_{\text{FVS}} S)} \le c_1,$$

where c_0, c_1 are independent of H, h, but may depend on the overlap ratio β.

Proof. Bounds for the extreme eigenvalues of $M^{-1}_{\text{FVS}} S$ are obtained from bounds for the Rayleigh quotient:

$$\lambda_{\min}(M^{-1}_{\text{FVS}} S) \le \left(\frac{x_B^T S x_B}{x_B^T M_{\text{VS}} x_B}\right)\left(\frac{x_B^T M_{\text{VS}} x_B}{x_B^T M_{\text{FVS}} x_B}\right) \le \lambda_{\max}(M^{-1}_{\text{FVS}} S).$$

The fraction $x_B^T S x_B / x_B^T M_{VS} x_B$ has an upper and lower bound independent of H and h (see [28]). It therefore suffices to obtain an upper and lower bound for the fraction $x_B^T M_{VS} x_B / x_B^T M_{FVS} x_B$ or equivalently for

$$\lambda_{\min}(M_{\text{FVS}} M^{-1}_{\text{VS}}) \le \frac{x_B^T M^{-1}_{\text{VS}} x_B}{x_B^T M^{-1}_{\text{FVS}} x_B} \le \lambda_{\max}(M_{\text{FVS}} M^{-1}_{\text{VS}}).$$

By spectral equivalence of the edge Fourier approximations, Lemma 4.1, there exist constants c_{ij} and C_{ij} independent of H and h such that

$$c_{ij} \le \frac{x_B^T S^{-1}_{E_{ij}} x_B}{x_B^T (\tilde{S}^F_{E_{ij}})^{-1} x_B} \le C_{ij}.$$

Similarly, for the vertex spaces, by Theorem 4.2, there exist constants c_k and C_k independent of H and h such that

$$c_k \le \frac{x_B^T S^{-1}_{V_k} x_B}{x_B^T (\tilde{S}^F_{V_k})^{-1} x_B} \le C_k.$$

Letting $C = \max\{C_{ij}, C_k\}$ and $c = \min\{c_{ij}, c_k\}$, we obtain

$$c \le \frac{\sum_{ij} x_B^T S_{E_{ij}}^{-1} x_B + \sum_k x_B^T S_{V_k}^{-1} x_B + x_B^T R_H^T A_H^{-1} R_H x_B}{\sum_{ij} x_B^T (\tilde{S}_{E_{ij}}^F)^{-1} x_B + \sum_k x_B^T (\tilde{S}_{E_k}^F)^{-1} x_B + x_B^T R_H^T A_H^{-1} R_H x_B} \le C,$$

and hence our result follows. □

4.2. Probe approximations. Next, we describe another variant of the VS and BPS preconditioners in which the edge and vertex matrices are approximated by sparse matrices obtained using an extension of the *probing* technique of Chan and Resasco [13], Keyes and Gropp [24], [25], and Eisenstat [18]. Unlike Fourier-based approximations, the construction of the probe approximations requires solving six problems on each subdomain, and thus has a greater overhead cost than the Fourier approximations, but still considerably less than the exact submatrices. An advantage of these approximations is that they often adapt well to coefficient variations and aspect ratios. However a disadvantage is that they do not adapt optimally to mesh size variations.

We will describe the construction of these probe approximations for the model rectangular geometry of Fig. 1. The techniques are easily extended to more general geometries.

4.2.1. Edge probe approximations. We first describe how sparse approximations to the edge matrices can be constructed [13]. In its basic form, the *probing* technique consists of approximating each $S_{E_{ij}}$ by a tridiagonal matrix $\tilde{S}_{E_{ij}}$, which is chosen on the assumption that each node on an edge is strongly coupled in S only to nodes adjacent to it and weakly coupled to the other nodes. A heuristic motivation for this is that the entries of each $S_{E_{ij}}$ are known to decay rapidly away from the main diagonals:

$$|(S_{E_{ij}})_{lm}| = O\left(\frac{1}{|l-m|^2}\right);$$

see Golub and Mayers [22].

To obtain a tridiagonal approximation $\tilde{S}_{E_{ij}}$ to $S_{E_{ij}}$, we equate the matrix-vector products $S_{E_{ij}} p_i$ to $\tilde{S}_{E_{ij}} p_i$ for the following three *probe* vectors p_i:

$$p_1 = [1,0,0,1,0,0,\ldots]^T, \quad p_2 = [0,1,0,0,1,0,\ldots]^T, \quad p_3 = [0,0,1,0,0,1,\ldots]^T.$$

These matrix-vector products $[\tilde{S}_{E_{ij}} p_1, \tilde{S}_{E_{ij}} p_2, \tilde{S}_{E_{ij}} p_3]$ result in

$$\begin{bmatrix} (\tilde{S}_{E_{ij}})_{11} & (\tilde{S}_{E_{ij}})_{12} & & \\ (\tilde{S}_{E_{ij}})_{21} & (\tilde{S}_{E_{ij}})_{22} & (\tilde{S}_{E_{ij}})_{23} & \\ & (\tilde{S}_{E_{ij}})_{32} & (\tilde{S}_{E_{ij}})_{33} & \ddots \\ & & \ddots & \ddots \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$= \begin{bmatrix} (\tilde{S}_{E_{ij}})_{11} & (\tilde{S}_{E_{ij}})_{12} & 0 \\ (\tilde{S}_{E_{ij}})_{21} & (\tilde{S}_{E_{ij}})_{22} & (\tilde{S}_{E_{ij}})_{23} \\ (\tilde{S}_{E_{ij}})_{34} & (\tilde{S}_{E_{ij}})_{32} & (\tilde{S}_{E_{ij}})_{33} \\ \vdots & \vdots & \vdots \end{bmatrix},$$

and equating this with $[S_{E_{ij}} p_1, S_{E_{ij}} p_2, S_{E_{ij}} p_3]$ gives

$$
\begin{bmatrix}
(\tilde{S}_{E_{ij}})_{11} & (\tilde{S}_{E_{ij}})_{12} & 0 \\
(\tilde{S}_{E_{ij}})_{21} & (\tilde{S}_{E_{ij}})_{22} & (\tilde{S}_{E_{ij}})_{23} \\
(\tilde{S}_{E_{ij}})_{34} & (\tilde{S}_{E_{ij}})_{32} & (\tilde{S}_{E_{ij}})_{33} \\
\vdots & \vdots & \vdots
\end{bmatrix} := [S_{E_{ij}} p_1, S_{E_{ij}} p_2, S_{E_{ij}} p_3], \tag{21}
$$

from which the nonzero entries of $\tilde{S}_{E_{ij}}$ can be easily read off. In general, $\tilde{S}_{E_{ij}}$ will not preserve the symmetry of $S_{E_{ij}}$, and so we symmetrize it to obtain $\tilde{S}^P_{E_{ij}}$ using a *minimum-modulus* procedure described below:

$$
(\tilde{S}^P_{E_{ij}})_{ij} \equiv \begin{cases} (\tilde{S}_{E_{ij}})_{ji} & \text{if } |(\tilde{S}_{E_{ij}})_{ji}| \le |(\tilde{S}_{E_{ij}})_{ij}|, \\ (\tilde{S}_{E_{ij}})_{ij} & \text{if } |(\tilde{S}_{E_{ij}})_{ij}| \le |(\tilde{S}_{E_{ij}})_{ji}|. \end{cases}
$$

We will denote the construction of $\tilde{S}^P_{E_{ij}}$ from $S_{E_{ij}} p_1$, $S_{E_{ij}} p_2$, $S_{E_{ij}} p_3$ by the notation

$$
\tilde{S}^P_{E_{ij}} = \text{PROBE}(S_{E_{ij}} p_1, S_{E_{ij}} p_2, S_{E_{ij}} p_3). \tag{22}
$$

The resulting approximations can be shown to preserve row-wise diagonal dominance (see [12]). This idea is motivated by Curtis, Powell, and Reid [15]. In an analogous way, using a symmetrized variant of [15] (see Powell and Toint [27]), it is possible to obtain a symmetric tridiagonal approximation directly using just two probe vectors (see [24] and [25]).

Computing the three matrix-vector products $S_{E_{ij}} p_i$ requires three solves on each subdomain Ω_i and Ω_j. Thus, in order to compute edge approximations $\tilde{S}^P_{E_{ij}}$ on the edges of all the subdomains, twelve solves on each subdomain would be required, since the boundary of rectangular subdomains consists of four edges.

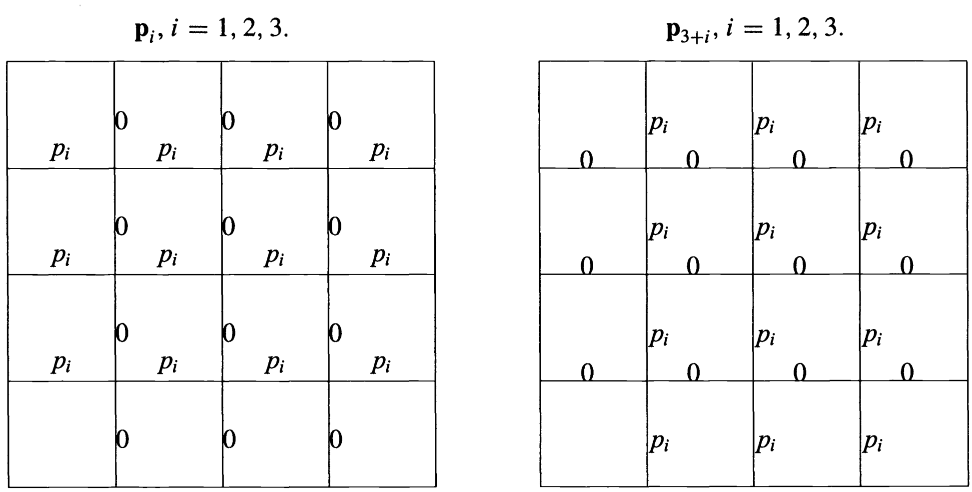


FIG. 3. *Simultaneous probe vectors.*

We now describe a procedure for computing all the edge approximations using only six solves on each subdomain, by simultaneously prescribing boundary conditions on other edges, an idea first used in Keyes and Gropp [24], [25]. To minimize the approximation errors arising from the coupling between vertical and horizontal edges, we will specify probe vectors

p_i either on all horizontal edges simultaneously, or on all vertical edges simultaneously. For $i = 1, 2, 3$ (see Fig. 3), define:

$$\mathbf{p}_i \equiv \begin{cases} p_i & \text{on all horizontal edges,} \\ 0 & \text{on all vertical edges,} \end{cases}$$

$$\mathbf{p}_{3+i} \equiv \begin{cases} 0 & \text{on all horizontal edges,} \\ p_i & \text{on all vertical edges.} \end{cases}$$

On the horizontal edges, the probe vectors p_i can be ordered from left to right, and on vertical edges from bottom to top. For these six probe vectors, we compute he discrete harmonic extensions $E^h\mathbf{p}_i = [-A_{II}^{-1}A_{IB}\mathbf{p}_i, \mathbf{p}_i]^T$, and this involves six solves on each subdomain. If E_{ij} is a horizontal edge, we define

$$\tilde{S}^P_{E_{ij}} = \text{PROBE}(R_{E_{ij}}A_hE^h\mathbf{p}_1, R_{E_{ij}}A_hE^h\mathbf{p}_2, R_{E_{ij}}A_hE^h\mathbf{p}_3).$$

If E_{ij} is a vertical edge, then we define

$$\tilde{S}^P_{E_{ij}} = \text{PROBE}(R_{E_{ij}}A_hE^h\mathbf{p}_4, R_{E_{ij}}A_hE^h\mathbf{p}_5, R_{E_{ij}}A_hE^h\mathbf{p}_6).$$

We have the following result on the nonsingularity and diagonal dominance of the resulting probe approximations.

THEOREM 4.4. *If the coefficient matrix A_h for the model rectangular geometry of Fig. 1 satisfies the discrete strong maximum principle (as is the case for standard five point discretizations), then the probe approximations $\tilde{S}^P_{E_{ij}}$ obtained above are strictly diagonally dominant.*

Proof. We will prove the diagonal dominance of approximation $\tilde{S}^P_{E_1}$ on edge E_1 in the model geometry of Fig. 2; the proof for the other edge approximations are analogous. By construction,

$$\tilde{S}^P_{E_1} = \text{PROBE}(R_{E_1}A_hE^h\mathbf{p}_1, R_{E_1}A_hE^h\mathbf{p}_2, R_{E_1}A_hE^h\mathbf{p}_3).$$

Due to the effects of the boundary conditions on the adjacent edges, it is easily verified that (see §3.2 for notation)

$$R_{E_1}A_hE^h\mathbf{p}_i = S_{E_1}p_i + S_{E_1E_6}p_i + S_{E_1E_7}p_i \quad \text{for } i = 1, 2, 3,$$

and from this we obtain:

$$\begin{aligned} (\tilde{S}^P_{E_1})_{i,i} &\equiv \textstyle\sum_{\text{mod}(i-j,3)=0}(S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{i,j}, \\ (\tilde{S}^P_{E_1})_{i,i-1} &\equiv \textstyle\sum_{\text{mod}(i-j,3)=1}(S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{i,j}, \\ (\tilde{S}^P_{E_1})_{i,i+1} &\equiv \textstyle\sum_{\text{mod}(i-j,3)=-1}(S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{i,j}. \end{aligned} \tag{23}$$

For discretizations A_h satisfying the discrete strong maximum principle, S is a diagonally dominant M-matrix (see [12]), and so its off-diagonal entries are nonpositive and its row sums are nonnegative. Using this in (23) we obtain that $(\tilde{S}^P_{E_1})_{ij} \le 0$ for $j \ne i$ and the row sum

$$(\tilde{S}^P_{E_1})_{i,i-1} + (\tilde{S}^P_{E_1})_{i,i} + (\tilde{S}^P_{E_1})_{i,i+1} = \sum_j (S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{i,j} > 0,$$

which shows that diagonal dominance is preserved. Finally, the *min-mod* procedure preserves diagonal dominance by definition. □

4.2.2. Probe vertex approximations. Next, we describe how sparse algebraic approximations to the vertex submatrices S_{V_k} can be constructed. Unlike the tridiagonal edge approximations $\tilde{S}^P_{E_{ij}}$, which enabled the use of fast direct solvers, the sparse approximations of the vertex matrices are usually small in general and will be solved by direct methods that do not make use of the sparsity of the matrices. The procedure we will describe results from a slight modification of a technique described in [11]. This new variant can be proven to result in nonsingular approximations that preserve diagonal dominance.

For simplicity, we will describe this procedure for the vertex region V_k in the center of the subdomains $\Omega_1, \ldots, \Omega_4$ of Fig. 2. We partition V_k into five disjoint regions:

$$V_k = (V_k \cap E_1) \cap (V_k \cap E_2) \cap (V_k \cap E_3) \cap (V_k \cap E_4) \cap (x_k^H, y_k^H), \tag{24}$$

and we obtain a corresponding 5×5 block partition of the vertex matrix S_{V_k}:

$$S_{V_k} = \begin{bmatrix} S_{11} & 0 & S_{13} & S_{14} & S_{15} \\ 0 & S_{22} & S_{23} & S_{24} & S_{25} \\ S_{13}^T & S_{23}^T & S_{33} & 0 & S_{35} \\ S_{14}^T & S_{24}^T & 0 & S_{44} & S_{45} \\ S_{15}^T & S_{25}^T & S_{35}^T & S_{45}^T & S_{55} \end{bmatrix},$$

where each S_{ij} corresponds to the coupling between nodes in block i and block j. The submatrices S_{12} and S_{34} and their transposes are zero, since there is no coupling in S between nodes in E_1 and E_2, and between nodes in E_3 and E_4. We will construct a vertex matrix approximation $\tilde{S}^P_{V_k}$ having the same block structure as S_{V_k}, with subblocks $\tilde{S}_{ij}$ which will be chosen to be sparse.

To facilitate description of the sparsity pattern, we will use the following ordering of nodes within V_k; for each of the four edge segments $E_i \cap V_k$, the nodes will be numbered to increase away from the cross-point (x_k^H, y_k^H), which is ordered last. This ordering is shown in Fig. 4 where each segment $E_i \cap V_k$ contains just two nodes.

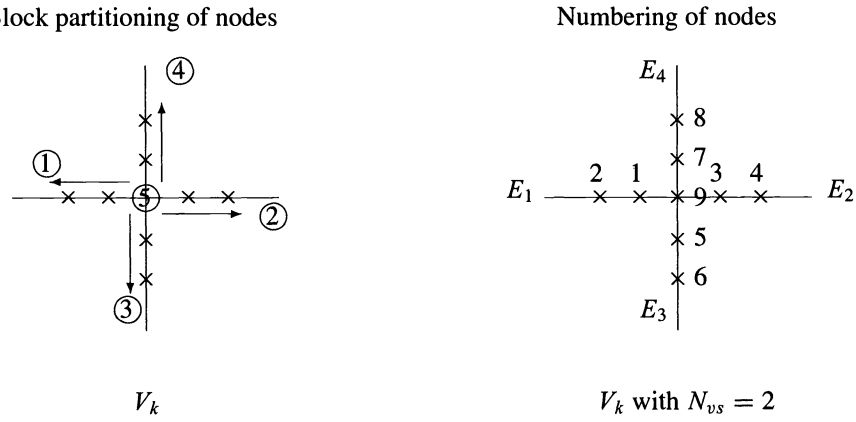


FIG. 4. *Ordering of unknowns within each vertex subregion V_k.*

Our choice of the sparsity pattern for the subblocks $\tilde{S}_{ij}$ is based on the assumption that the elements of S_{V_k} decay with increasing distance between nodes.

Definition and computation of the edge blocks $\tilde{S}_{ii}$ for $i = 1, 2, 3, 4$. Within each edge segment $E_i \cap V_k$ we assume the coupling in S_{V_k} is strong only between adjacent nodes. Based on this assumption, S_{ii} will be approximated by tridiagonal matrices $\tilde{S}_{ii}$ which are chosen to be the submatrices of the tridiagonal edge matrices $\tilde{S}^P_{E_i}$ for $i = 1, 2, 3, 4$, which were computed in §4.2.

Definition and computation of the blocks $\tilde{S}_{i5}$ for $i = 1, \ldots, 5$. We assume the cross-point (x_k^H, y_k^H) is coupled strongly in S_{V_k} only to the nodes adjacent to it. Based on this assumption, we choose the vectors $\tilde{S}_{i5}$ to have zero entries except in the first entry:

$$\tilde{S}_{i5} = \begin{bmatrix} (\tilde{S}_{i5})_1 \\ 0 \\ \vdots \end{bmatrix} \quad \text{for } i = 1, \ldots, 5.$$

For five-point discretizations on the rectangular geometry of Fig. 1, it can easily be shown that the last row and column of S_{V_k} is exactly equal to the last row and column of $R_{V_k} A_h R_{V_k}^T$, the matrix A_h restricted to V_k. Therefore, we define

$$\begin{aligned} \tilde{S}_{i5} &\equiv A_{i5} = S_{i5}, \qquad i = 1, \ldots, 5, \\ \tilde{S}_{5i} &\equiv A_{5i} = S_{5i}, \qquad i = 1, \ldots, 5. \end{aligned}$$

To see that $A_{i5} = S_{i5}$, first note that S_{i5} is equal to the restriction of Su_B to the ith edge of V_k, where u_B corresponds to boundary data which is 1 on the kth vertex, and zero elsewhere. Now, recall that $Su_B = R_B A_h E^h u_B$. For five-point discretizations on rectangular subdomains, the boundary conditions on the corner nodes do not influence the solution in the interior. Consequently, the discrete harmonic extension $E^h u_B$ is zero in the interior of subdomains, and $A_h E^h u_B$ simply gives the column of A_h corresponding to the kth vertex. Thus $S_{i5} = A_{i5}$.

Definition and computation of $\tilde{S}_{ij}$ for $i = 1, 2$ and $j = 3, 4$. We assume the couplings in S_{V_k} between edge segments $E_i \cap V_k$ and $E_j \cap V_k$ are strong only between the nodes that are closest (adjacent) to the cross-point (x_k^H, y_k^H). Based on this assumption, we choose the submatrices $\tilde{S}_{13}$, $\tilde{S}_{14}$, $\tilde{S}_{23}$, and $\tilde{S}_{24}$ and their transposes to have all zero entries except for the (1, 1)th entry

$$\tilde{S}_{ij} = \begin{bmatrix} (\tilde{S}_{ij})_{11} & 0 & \cdots \\ 0 & 0 & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \quad \text{for } i = 1, 2; \quad j = 3, 4.$$

So there are only eight nonzero entries to define.

Consider, for example, the entry $(\tilde{S}_{14})_{11}$, which we would like to be an approximation to $(S_{14})_{11}$, the coupling in S between node $(x_k^H - h, y_k^H)$ and node $(x_k^H, y_k^H + h)$. Note that $(S_{14})_{11} = (S\delta_k)(x_k^H - h, y_k^H)$ (i.e., the component of $S\delta_k$ corresponding to the point $(x_k^H - h, y_k^H)$) where δ_k is the boundary data which is 1 on $(x_k^H, y_k^H + h)$ and zero elsewhere, and therefore computing $(S_{14})_{11}$ requires one subdomain solve. In order to reduce this overhead, we would like to extract an approximation from the subdomain solves we already used for the probe edge approximations. For example, one could define $(\tilde{S}_{14})_{11} = (S\mathbf{p}_4)(x_k^H - h, y_k^H)$. However, it turns out that this definition can lead to a nondiagonally dominant (and possibly singular) $\tilde{S}_{V_k}$. This can be seen by noting that

$$(S\mathbf{p}_4)(x_k^H - h, y_k^H) = (S_{E_1E_4}p_1 + S_{E_1E_{10}}p_1 + S_{E_1E_3}p_1 + S_{E_1E_{12}}p_1)(x_k^H - h, y_k^H).$$

The last two terms on the right correspond to extra influence from Ω_4 on the coupling between nodes $(x_k^H - h, y_k^H)$ and $(x_k^H, y_k^H + h)$ (which should only involve couplings within Ω_1).

These extra couplings could cause loss of diagonal dominance, since, in case the coefficients are large in Ω_4, the last two terms will dominate the sum on the right. In order to eliminate the influence from Ω_4, we now define

$$(\tilde{S}_{14})_{11} = (S_{E_1E_4}p_1 + S_{E_1E_{10}}p_1)(x_k^H - h, y_k^H) \quad \left(\equiv (R_{E_1}A^{(1)}E^h\mathbf{p}_4)_1\right),$$

where we recall that $A^{(1)}$ is the local stiffness matrix on Ω_1. The last equality comes from the definition of the local Schur complement, and can be extracted from the subdomain solves used to construct the edge approximations.

Analogously, we define the seven remaining nonzero entries by

$$\begin{aligned}
(\tilde{S}_{13})_{11} &\equiv (R_{E_1}A^{(4)}E^h\mathbf{p}_4)_1,\\
(\tilde{S}_{24})_{11} &\equiv (R_{E_2}A^{(2)}E^h\mathbf{p}_4)_1,\\
(\tilde{S}_{23})_{11} &\equiv (R_{E_2}A^{(3)}E^h\mathbf{p}_4)_1,\\
(\tilde{S}_{31})_{11} &\equiv (R_{E_3}A^{(4)}E^h\mathbf{p}_1)_1,\\
(\tilde{S}_{32})_{11} &\equiv (R_{E_3}A^{(3)}E^h\mathbf{p}_1)_1,\\
(\tilde{S}_{41})_{11} &\equiv (R_{E_4}A^{(1)}E^h\mathbf{p}_1)_1,\\
(\tilde{S}_{42})_{11} &\equiv (R_{E_4}A^{(2)}E^h\mathbf{p}_1)_1.
\end{aligned} \tag{25}$$

Symmetrization of $\tilde{S}_{V_k}$. Finally, in order to obtain a symmetric vertex approximation $\tilde{S}_{V_k}^P$ we use the *minimum-modulus* procedure

$$(\tilde{S}_{V_k}^P)_{ij} \equiv \begin{cases} (\tilde{S}_{V_k})_{ij} & \text{if } |(\tilde{S}_{V_k})_{ij}| \le |(\tilde{S}_{V_k})_{ji}|,\\ (\tilde{S}_{V_k})_{ji} & \text{if } |(\tilde{S}_{V_k})_{ji}| \le |(\tilde{S}_{V_k})_{ij}|. \end{cases} \tag{26}$$

THEOREM 4.5. *The vertex matrix approximations $\tilde{S}_{V_k}^P$ are nonsingular, diagonally dominant M-matrices.*

Proof. First, we note that since the fifth block row of $\tilde{S}_{V_k}^P$ is identical to the fifth block row of S_{V_k}, it has zero row sum. For any other row of $\tilde{S}_{V_k}^P$ centered about nodes not adjacent to the cross-point, the nonzero entries are the nonzero entries of the diagonal blocks $\tilde{S}_{ii}$, for $i = 1, 2, 3, 4$. These diagonal blocks were chosen as submatrices of $\tilde{S}_{E_1}^P$, $\tilde{S}_{E_2}^P$, $\tilde{S}_{E_3}^P$, and $\tilde{S}_{E_4}^P$, respectively, which were shown to be diagonally dominant M-matrices in Theorem 4.4, and therefore these rows are more diagonally dominant than the corresponding rows of S.

We now prove the diagonal dominance of the rows centered about nodes adjacent to the cross-point (x_k^H, y_k^H). Consider, for instance, the row sum corresponding to node $(x_k^H - h, y_k^H)$ to the left of the cross-point (x_k^H, y_k^H). The nonzero entries of this row are $(\tilde{S}_{11})_{11}$, $(\tilde{S}_{11})_{12}$, $(\tilde{S}_{13})_{11}$, $(\tilde{S}_{14})_{11}$, and $(\tilde{S}_{15})_{11}$. By construction,

$$\begin{aligned}
(\tilde{S}_{11})_{11} &= \textstyle\sum_{\mathrm{mod}(j-1,3)=0}(S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{1,j} > 0,\\
(\tilde{S}_{11})_{12} &= \textstyle\sum_{\mathrm{mod}(j-2,3)=0}(S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{1,j} \le 0,\\
(\tilde{S}_{13})_{11} &= \textstyle\sum_{\mathrm{mod}(j-1,3)=0}(S_{E_1E_3} + S_{E_1E_{10}})_{1,j} \le 0,\\
(\tilde{S}_{14})_{11} &= \textstyle\sum_{\mathrm{mod}(j-1,3)=0}(S_{E_1E_4} + S_{E_1E_{13}})_{1,j} \le 0,\\
(\tilde{S}_{15})_{11} &= (S_{E_1E_5})_{11} \le 0.
\end{aligned}$$

By summing all these nonzero entries, we obtain

$$\begin{aligned}
\textstyle\sum_j(\tilde{S}_{V_k})_{1j} = &\textstyle\sum_{\mathrm{mod}(j-1,3)=0}(S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{1,j}\\
&+\textstyle\sum_{\mathrm{mod}(j-2,3)=0}(S_{E_1} + S_{E_1E_6} + S_{E_1E_7})_{1,j}\\
&+\textstyle\sum_{\mathrm{mod}(j-1,3)=0}(S_{E_1E_3} + S_{E_1E_{10}})_{1,j}\\
&+\textstyle\sum_{\mathrm{mod}(j-1,3)=0}(S_{E_1E_4} + S_{E_1E_{13}})_{1,j}\\
&+(S_{E_1E_5})_{11}.
\end{aligned}$$

The right-hand side is a *subset* of the corresponding row of S, which is strictly diagonally dominant, which shows that this row of $\tilde{S}_{V_k}^P$ is diagonally dominant. The proof of the diagonal dominance of the other rows centered about the nodes adjacent to (x_k^H, y_k^H) is analogous. Thus $\tilde{S}_{V_k}^P$ is strictly diagonally dominant in all rows except the one corresponding to the cross-point. This last property, together with the fact that $\tilde{S}_{V_k}^P$ has positive diagonal elements and nonpositive off-diagonal elements, implies that $\tilde{S}_{V_k}^P$ is a nonsingular M-matrix. □

4.2.3. Probe-based preconditioner. We now define the probe vertex space (PVS) preconditioner by

$$M_{PVS}^{-1} \equiv R_H^T A_H^{-1} R_H + \sum_{ij} R_{E_{ij}}^T (\tilde{S}_{E_{ij}}^P)^{-1} R_{E_{ij}} + \sum_k R_{V_k}^T (\tilde{S}_{V_k}^P)^{-1} R_{V_k}, \tag{27}$$

and the probe BPS (PBPS) preconditioner by:

$$M_{PBPS}^{-1} \equiv R_H^T A_H^{-1} R_H + \sum_{ij} R_{E_{ij}}^T (\tilde{S}_{E_{ij}}^P)^{-1} R_{E_{ij}}. \tag{28}$$

5. Numerical results. We now present results of numerical tests on the rate of convergence of the Fourier and probe variants of the BPS and VS algorithms. The tests were conducted for the following elliptic problem:

$$\begin{aligned} -\nabla \cdot (a(x, y)\nabla u) &= f \quad \text{in } \Omega = [0, 1]^2, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned}$$

for five choices of coefficients $a(x, y)$, various subdomain sizes H, and fine grid sizes h. The five coefficients used were:

1. $a(x, y) = I$, the Laplacian; see Table 1.
2. $a(x, y) = I + 10(x^2 + y^2)I$, slowly varying smooth coefficients; see Table 2.
3. $a(x, y) = e^{10xy}I$, highly varying smooth coefficients; see Table 3.
4. $a(x, y) = \text{diag}(1, \epsilon)$, anisotropic coefficients; see Table 4.
5. Highly discontinuous coefficients of Fig. 5; see Table 5.

TABLE 1
Laplace's equation: $a(x, y) = I$.

h^{-1}	Ovlp	FBPS		PBPS		EVS		FVS		PVS	
H^{-1}	h/H	κ	ITN	κ	ITN	κ	ITN	κ	ITN	κ	ITN
32_2	1/16	14.3	11	9.9	9	3.4	7	5.7	11	3.2	8
32_8	1/4	6.4	12	5.4	11	2.5	8	3.5	10	2.4	8
64_4	1/16	14.5	14	11.3	12	3.4	9	5.9	13	3.2	9
64_16	1/4	6.5	13	5.6	11	2.6	8	3.6	10	2.5	8
128_4	1/32	19.8	16	18.4	15	4.4	10	7.4	13	4.1	10
128_16	1/8	10.4	14	8.3	13	2.8	9	4.6	11	2.7	9
256_4	1/64	25.4	16	33.0	19	5.5	10	9.1	13	7.2	13
256_16	1/16	14.7	16	12.4	13	3.5	9	5.9	13	3.3	9
256_64	1/4	6.5	13	5.7	11	2.6	8	3.6	10	2.4	8

TABLE 2
Mildly varying coefficients: $a(x, y) = \left(1 + 10(x^2 + y^2)\right) I$.

h^{-1}	Ovlp	FBPS		PBPS		FVS		PVS	
$\_H^{-1}$	h/H	κ	ITN	κ	ITN	κ	ITN	κ	ITN
32_2	1/16	15.2	11	10.6	9	6.0	11	3.4	8
32_8	1/4	6.4	12	5.4	11	3.6	10	2.4	8
64_4	1/16	14.9	14	11.6	12	5.8	12	3.2	9
64_16	1/4	6.5	13	5.6	11	3.6	10	2.4	8
128_4	1/32	20.0	16	18.4	15	7.3	13	4.2	10
128_16	1/8	10.4	14	8.4	13	4.6	11	2.7	8
256_4	1/64	25.8	17	33.2	19	9.3	13	7.2	13
256_16	1/16	14.7	16	12.3	13	5.9	13	3.4	9
256_64	1/4	6.5	13	5.7	11	3.6	10	2.4	8

TABLE 3
Highly varying coefficients: $a(x, y) = e^{10xy} I$.

h^{-1}	Ovlp	FBPS		PBPS		nsFVS		FVS		PVS	
$\_H^{-1}$	h/H	κ	ITN	κ	ITN	κ	ITN	κ	ITN	κ	ITN
32_2	1/16	22.5	11	18.4	9	16.1	18	7.5	11	4.4	9
32_8	1/4	7.0	12	6.2	11	4.0	10	3.9	10	2.5	8
64_4	1/16	17.6	16	15.5	15	11.3	16	6.5	12	4.0	9
64_16	1/4	6.6	12	5.8	11	3.7	10	3.7	10	2.5	8
128_4	1/32	24.4	16	23.3	15	16.1	19	8.4	13	5.1	10
128_16	1/8	10.4	14	8.4	11	4.7	12	4.6	11	2.8	9
256_4	1/64	29.3	17	41.4	22	16.2	19	10.1	13	8.5	13
256_16	1/16	15.0	15	12.4	13	5.0	11	6.1	13	3.3	9
256_64	1/4	6.5	12	5.6	11	2.9	9	3.6	10	2.4	8

TABLE 4
Anisotropic problem: $\partial^2 u/\partial x^2 + \epsilon(\partial^2 u/\partial y^2) = f$.

$h^{-1}\_H^{-1}$	64_2				64_4				64_16			
	PVS		FVS		PVS		FVS		PVS		FVS	
ϵ	κ	ITN	κ	ITN	κ	ITN	κ	ITN	κ	ITN	κ	ITN
0.1	7.4	10	14.5	17	5.9	12	12.0	18	9.0	16	12.9	19
0.06	8.9	10	18.4	19	7.5	13	15.2	21	13.6	18	19.7	23
0.02	13.0	10	31.4	24	13.0	16	29.4	28	34.4	27	50.1	33
10^{-3}	29.3	8	115.6	38	60.3	25	151.5	47	215.8	59	351.8	73
10^{-5}	41.8	6	193.8	48	105.1	27	253.6	59	396.6	73	583.6	87
10^{-7}	42.1	6	195.4	48	102.1	25	273.1	59	405.6	73	654.0	92

$a = 300$	$a = 10^{-4}$	$a = 31400$	$a = 5$
$a = 0.05$	$a = 6$	$a = 0.07$	$a = 2700$
$a = 10^6$	$a = 0.1$	$a = 200$	$a = 9$
$a = 1$	$a = 6000$	$a = 4$	$a = 140000$

FIG. 5. *Discontinuous coefficients* $a(x, y)$.

TABLE 5
Discontinuous coefficients: see $a(x, y)$ *of Fig.* 5.

h^{-1}	Ovlp	FBPS		PBPS		FVS		PVS	
$\_H^{-1}$	h/H	κ	ITN	κ	ITN	κ	ITN	κ	ITN
32_4	1/8	10.2	13	7.5	11	6.1	12	8.1	11
64_4	1/16	14.7	15	11.1	11	9.3	14	10.1	11
64_16	1/4	6.5	13	5.6	11	6.9	12	4.1	9
128_8	1/16	14.4	16	12.1	14	11.5	15	5.9	11
128_32	1/4	6.6	13	5.7	11	6.8	12	4.1	9
256_4	1/64	25.4	19	33.0	17	14.9	15	7.8	13
256_16	1/16	14.8	16	12.3	13	12.4	16	6.9	11
256_64	1/4	6.5	13	5.7	11	6.0	12	4.1	9

The elliptic problem was discretized using the standard five-point difference stencil (see [30]) on an $(n + 1) \times (n + 1)$ uniform fine grid with mesh size $h = 1/n$. The subdomains were chosen to be the subrectangles of an $(n_s + 1) \times (n_s + 1)$ uniform coarse grid with mesh size $H = 1/n_s$. Each subdomain, therefore, consisted of $(n/n_s - 1) \times (n/n_s - 1)$ interior nodes. The coarse grid matrix A_H was chosen to be the five-point difference approximation of the elliptic problem on the coarse grid.

The entries of the exact solution were chosen randomly from the uniform distribution on $[-1, 1]$ and the initial guess in the conjugate gradient method was chosen to be zero. The estimated condition number, κ, of the preconditioned system, and the number of iterations, ITN, required to reduce the initial residual by a factor of 10^{-5} (i.e., $\|r_k\|_2/\|r_0\|_2 \leq 10^{-5}$) are listed in the tables. During each iteration, the coarse grid problem and the subdomain problems were solved to high precision using a diagonally scaled preconditioned conjugate

gradient method for simplicity, though this was not the most practical choice. The eigenvalues μ_k in the edge approximations $\tilde{S}^F_{E_{ij}}$ of (16) were chosen to be the Bramble, Pasciak, and Schatz eigenvalues listed in (15), while the eigenvalues of the submatrices M_i^k of (18) were chosen to be the Dryja eigenvalues in (15). The Fourier and probe BPS versions are denoted by FBPS and PBPS, respectively, while the Fourier and Probe versions of the VS algorithms are denoted FVS and PVS, respectively. Unless otherwise stated, the number of nodes of overlap, N_{vs}, in the vertex regions is 1, i.e., there is one node on each vertex segment $V_k \cap E_{ij}$. The overlap ratio $\beta = h/H$ is listed as *Ovlp*.

Discussion. Tables 1–5 compare the performance of the various methods for the five sets of coefficients listed above. Table 1 corresponds to the Laplacian. In this case, the exact version of the VS algorithm, denoted by EVS, was also tested, because the eigenvalues of edge matrices $S_{E_{ij}}$ can be computed inexpensively using analytical formulas; see M_{Chan} in (15). In

TABLE 6
Different edge Fourier preconditioners for Laplace equation.

h^{-1}	Ovlp	FBPS		CFBPS		FVS		CFVS		PVS	
$\_H^{-1}$	h/H	κ	ITN	κ	ITN	κ	ITN	κ	ITN	κ	ITN
32_2	1/16	14.3	11	9.5	7	5.7	11	4.6	8	3.2	8
32_8	1/4	6.4	12	5.3	11	3.5	10	2.9	9	2.4	8
64_4	1/16	14.5	14	10.7	11	5.9	13	4.7	10	3.2	9
64_16	1/4	6.5	13	5.5	11	3.6	10	2.9	9	2.5	8
128_2	1/64	25.0	13	17.8	8	9.0	11	7.3	8	6.5	11
128_8	1/16	14.7	16	11.5	14	5.9	13	4.7	10	3.4	9
128_32	1/4	6.5	13	5.5	11	3.6	10	2.9	9	2.5	8
256_4	1/64	25.4	16	19.2	13	9.0	14	7.3	10	7.2	13
256_16	1/16	14.7	16	11.7	14	5.9	13	4.7	10	3.3	9
256_64	1/4	6.5	13	5.5	11	3.6	10	2.9	9	2.5	8

TABLE 7
Different edge Fourier preconditioners for $a(x, y) = e^{10xy}I$.

h^{-1}	Ovlp	FBPS		CFBPS		FVS		CFVS		PVS	
$\_H^{-1}$	h/H	κ	ITN	κ	ITN	κ	ITN	κ	ITN	κ	ITN
32_2	1/16	22.5	11	18.1	8	7.5	11	6.2	9	4.4	9
32_8	1/4	7.0	12	5.8	11	3.9	10	3.3	9	2.5	8
64_4	1/16	17.6	16	14.7	12	6.5	12	5.4	9	4.0	9
64_16	1/4	6.6	12	5.6	11	3.7	10	3.0	9	2.5	8
128_2	1/64	36.3	13	28.5	9	11.8	12	9.6	9	8.6	11
128_8	1/16	15.7	14	12.5	11	6.0	12	5.1	10	3.6	10
128_32	1/4	6.5	12	5.5	11	3.6	10	3.0	9	2.4	8
256_4	1/64	29.3	17	23.3	14	10.1	13	8.3	10	8.5	13
256_16	1/16	15.0	15	11.9	12	6.1	13	4.8	10	3.3	9
256_64	1/4	6.5	12	5.4	11	3.6	10	2.9	9	2.4	8

TABLE 8
Variation of vertex sizes for $H = 1/2$, $h = 1/128$, *and* $a(x, y) = I$.

N_{VS}	0	1	2	3	4	5	6	7
κ_{FVS}	7.45	8.97	8.07	7.66	6.85	6.98	6.71	6.53
ITN	10	11	12	12	12	13	12	12

TABLE 9
Variation of vertex sizes for $H = 1/2$, $h = 1/128$, *and* $a(x, y) = e^{10xy}I$.

N_{VS}	0	1	2	3	4	5	6	7
κ_{FVS}	9.85	11.80	10.25	10.00	9.41	9.01	8.63	8.40
ITN	11	12	12	13	12	12	12	13

TABLE 10
Variation of vertex sizes for $H = 1/2$, $h = 1/128$, $a(x, y) = I$.

N_{VS}	0	1	2	3	4	5	6	7
κ_{PVS}	8.3	6.6	5.6	5.0	4.8	3.2	4.6	4.5
ITN	11	11	11	11	11	9	11	11

TABLE 11
Variation of vertex sizes for $H = 1/2$, $h = 1/128$, *and* $a(x, y) = e^{10xy}I$.

N_{VS}	0	1	2	3	4	5	6	7
κ_{PVS}	10.8	9.1	7.3	6.6	6.6	6.6	6.8	6.9
ITN	12	11	10	10	10	11	11	11

agreement with the theory, these results indicate that the Fourier variant FVS has an observed rate of convergence independent of the mesh parameters H, h for fixed overlap ratio Ovlp. Moreover, the actual iteration numbers are quite insensitive to the choice of parameters H, h, and Ovlp. For the range of subdomain and fine grid sizes tested, the performance of PVS is very similar to EVS. However, as the number of nodes per edge increases significantly, it is expected that the PVS version would deteriorate, based on properties of the probe preconditioner for two subdomains in [12]. The condition numbers for the variants of the BPS algorithms grow mildly with H/h, in agreement with theory. In most cases, due to clustering of eigenvalues of the preconditioned system, the number of iterations, ITN, was often better than that predicted by the condition numbers.

Tables 2 and 3 correspond to smoothly varying coefficients. Here again, the results are similar to those for the Laplacian, and are in agreement with the theory. Moreover, the rate of convergence of most variants are quite insensitive to the variations in the coefficients $a(x, y)$. In order to see the importance of scalings, in Table 3 we also tested a variant nsFVS of the FVS preconditioner, in which the edge approximations were not diagonally scaled, but were instead scaled by a scalar α_{ij} on each edge E_{ij}, i.e.,

$$\tilde{S}^F_{E_{ij}} \equiv \alpha_{ij} W \operatorname{diag}(\mu_k) W,$$

where

$$\alpha_{ij} \equiv \frac{a(x_i, y_i) + a(x_j, y_j)}{2},$$

for some point $(x_i, y_i) \in \Omega_i$ and $(x_j, y_j) \in \Omega_j$. As the results indicate, this variant was sensitive to the variations in the coefficients.

Table 4 concerns the case of anisotropic coefficients. Here, the results are qualitatively different from the preceding cases. Note that the rate of convergence of all variants of the VS and BPS algorithms deteriorates to a fixed rate as $\epsilon \to 0$. The limiting condition numbers seem to depend on the coarse mesh size, as $1/H$. A possible explanation for this deterioration is the following. For $\epsilon = 0$, the unknowns are essentially coupled only along the x axis and adjacent vertical edges are coupled strongly in the Schur complement. This coupling is not represented in the VS preconditioner, and may cause the deterioration in the convergence rate. The results in Table 4 also indicate that the probe versions perform slightly better than the Fourier versions. This can be explained as follows. For $\epsilon = 0$, the edge matrices $S_{E_{ij}}$ on the horizontal edges become a discrete approximation of $-d^2/dx^2$, while on vertical edges $S_{E_{ij}}$ becomes a nearly diagonal matrix, similar to the identity. The FVS edge matrices $\tilde{S}^P_{E_{ij}}$ approximate the square root of the Laplacian, and are therefore invalid in this case. By construction, the tridiagonal probing technique approximates diagonal and tridiagonal matrices well, and consequently, they perform better than the Fourier versions we tested. The algorithms for anisotropic problems need further study.

Table 5 refers to the case of the highly discontinuous coefficients of Fig. 5. The performance is similar to the case of smooth coefficients, and the results indicate that the rate of convergence of all variants is quite insensitive to the jumps in the coefficients.

In Tables 6 and 7, we compare various preconditioners for different choices of eigenvalues μ_k in the Fourier approximations (16). Here, CFBPS denotes that the eigenvalues of the Fourier edge approximations in the FBPS preconditioner were those of M_{Chan} in (15), while CFVS denotes that the same eigenvalues were used in the FVS preconditioner. In agreement with theory, the Fourier versions were spectrally equivalent with respect to variations in H and h, for fixed overlap Ovlp. Amongst the various eigenvalues tested, the exact eigenvalues of the Schur complement of the Laplacian used in CFBPS and CFVS gave the best results. Corresponding rates for the probe version are also listed for comparison.

Finally, in Tables 8, 9, 10, and 11, we present a comparison of the FVS and PVS preconditioners, as the amount of overlap N_{VS} in the vertex regions is increased. Here, $N_{VS} = 0$ indicates that only the vertex node was used, i.e., the vertex matrices were 1×1. We note that the improvement in condition number of the VS algorithms as the overlap Ovlp is increased is mild, as also noted in [28]. In particular, the performance is quite satisfactory even when the vertex region consists of just one point (see Widlund [32]).

6. Conclusions. Both the Fourier and probe variants of the VS algorithm are designed to be efficient alternatives to the original VS algorithm. Our experiments for a wide range of coefficients and grid sizes show that this efficiency does not come at the price of deterioration in performance. We hope that these variants will provide flexible and efficient methods for solving second-order elliptic problems using the domain decomposition approach.

REFERENCES

[1] V. I. AGOSHKOV, *Poincaré-Steklov operators and domain decomposition methods in finite dimensional spaces*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[2] P. E. BJØRSTAD AND O. B. WIDLUND, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal., 23 (1986), pp. 1093–1120.

[3] C. BÖRGERS, *The Neumann-Dirichlet domain decomposition method with inexact solvers on the subdomains*, Numer. Math., 55 (1989), pp. 123–136.

[4] J. H. BRAMBLE, J. E. PASCIAK, AND A. H. SCHATZ, *The construction of preconditioners for elliptic problems by substructuring, I*, Math. Comp., 47 (1986), pp. 103–134.

[5] ———, *An iterative method for elliptic problems on regions partitioned into substructures*, Math. Comp., 46 (1986), pp. 361–369.

[6] T. CHAN, R. GLOWINSKI, J. PÉRIAUX, AND O. WIDLUND, EDS., *Domain Decomposition Methods*, Proceedings of the Second International Symposium on Domain Decomposition Methods, Los Angeles, California, January 14–16, 1988, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[7] ———, EDS., *Domain Decomposition Methods*, Proceedings of the Third International Symposium on Domain Decomposition Methods, Houston, Texas, 1989, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[8] T. F. CHAN, *Analysis of preconditioners for domain decomposition*, SIAM J. Numer. Anal., 24 (1987), pp. 382–390.

[9] T. F. CHAN AND D. GOOVAERTS, *A note on the efficiency of domain decomposed incomplete factorizations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 794–803.

[10] T. F. CHAN AND T. Y. HOU, *Eigendecomposition of domain decomposition interface operators for constant coefficient elliptic problems*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 1471–1479.

[11] T. F. CHAN AND T. P. MATHEW, *An application of the probing technique to the vertex space method in domain decomposition*, in 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, 1990.

[12] ———, *The interface probing technique in domain decomposition*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 212–238.

[13] T. F. CHAN AND D. C. RESASCO, *A survey of preconditioners for domain decomposition*, Tech. Rep. DCS/RR-414, Yale University, New Haven, CT, 1985.

[14] P. G. CIARLET, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978.

[15] A. R. CURTIS, M. J. POWELL, AND J. K. REID, *On the estimation of sparse Jacobian matrices*, J. Inst. Maths. Applics., 13 (1974), pp. 117–120.

[16] M. DRYJA, *A capacitance matrix method for Dirichlet problem on polygon region*, Numer. Math., 39 (1982), pp. 51–64.

[17] M. DRYJA AND O. B. WIDLUND, *Some domain decomposition algorithms for elliptic problems*, in Proceedings of the Conference on Iterative Methods for Large Linear Systems held in Austin, Texas, October 1988, to celebrate the Sixty-fifth Birthday of David M. Young, Jr., Academic Press, Orlando, FL, 1989.

[18] S. C. EISENSTAT, personal communication, 1985.

[19] D. FUNARO, A. QUARTERONI, AND P. ZANOLLI, *An iterative procedure with interface relaxation for domain decomposition methods*, SIAM J. Numer. Anal., 25 (1988), pp. 1213–1236.

[20] R. GLOWINSKI, G. H. GOLUB, G. A. MEURANT, AND J. PÉRIAUX, EDS., *Domain Decomposition Methods for Partial Differential Equations*, Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Paris, France, January 1987, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[21] R. GLOWINSKI, Y. A. KUZNETSOV, G. A. MEURANT, J. PÉRIAUX, AND O. WIDLUND, EDS., *Domain Decomposition Methods*, Proceedings of the Fourth International Symposium on Domain Decomposition Methods, Moscow, USSR, 1990, Society for Industrial and Applied Mathematics, Philadephia, PA, 1990.

[22] G. GOLUB AND D. MAYERS, *The use of preconditioning over irregular regions*, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J. L. Lions, eds., Proceedings of a conference held in Versailles, France, December 12–16, 1983, North–Holland, Amsterdam, 1984, pp. 3–14.

[23] G. H. GOLUB AND C. F. V. LOAN, *Matrix Computations*, Second Ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[24] 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. Stat. Comput., 8 (1987), pp. s166–s202.

[25] ———, *Domain decomposition techniques for the parallel solution of nonsymmetric systems of elliptic bvps*, in Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[26] S. V. NEPOMNYASCHIKH, *On the application of the method of bordering for elliptic mixed boundary value problems and on the difference norms of* $W_2^{1/2}(S)$, Tech. Rep. 106, Computing Center of the Siberian Branch of the USSR Academy of Sciences, Novosibirsk, 1984. (In Russian.)

[27] M. J. POWELL AND P. L. TOINT, *On the estimation of sparse Hessian matrices*, SIAM J. Numer. Anal., 16 (1979), pp. 1060–1074.

[28] B. F. SMITH, *An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems*, Tech. Rep. 482, Department of Computer Science, Courant Institute, New York, 1989; SIAM J. Sci. Comput., 13 (1992), pp. 364–378. (Proceedings of Copper Mountain Conference on Iterative Methods.)

[29] ———, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Ph.D. thesis, Courant Institute, New York, 1990.

[30] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962.

[31] O. B. WIDLUND, *Iterative substructuring methods: Algorithms and theory for elliptic problems in the plane*, in First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988.

[32] ———, *Some Schwarz methods for symmetric and non-symmetric elliptic problems*, Tech. Rep. 0581, Courant Institute, New York, 1991.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1375–1400, November 1994

007

SOLUTION OF THE SYSTEMS ASSOCIATED WITH INVARIANT TORI APPROXIMATION. II: MULTIGRID METHODS*

LUCA DIECI† AND GEORG BADER‡

Abstract. In this paper, the authors continue (see [*Solution of the Systems Associated with Invariant Tori Approximation.* I: *Block Iterations and Compactifcation*, manuscript, available from the first author]) the study of solution strategies for the linear systems arising from an upwind discretization of the periodic PDEs with same principal part associated with invariant tori. Here, multigrid approaches for the solution of these systems are considered. Several choices of smoothers are considered, and opportune choices of prolongations and restrictions are discussed. Some convergence results, as well as implementation details and numerical results for linear model problems are given. A nonlinear upwind discretization is discussed, and results are presented for the tori associated with two physically important nonlinear problems: the forced Josephson junction and a system of two coupled oscillators.

Key words. invariant tori, numerical computation, PDEs with same principal part, upwind discretization, multigrid methods, Josephson junction, coupled oscillators

AMS subject classifications. 65L99, 65N07, 65F10

1. Introduction.

This paper is the second part of our study of solution strategies for the systems arising from a first-order discretization of upwinding type for the systems of hyperbolic partial differential equations (PDEs) with same principal part associated with *invariant tori of dynamical systems*. Whereas in [DB1] we looked at some of the standard iterative methods and direct compactification algorithms, here we consider multigrid approaches for the solution of the linear systems. The reason for considering multigrid approaches is simple. The basic iterative schemes considered in [DB1], applied to the matrices arising after discretization of the invariant tori PDEs, show a potentially slow speed of convergence if the tori are not sufficiently attractive.

Multigrid theory for hyperbolic PDEs is not as well settled as it is for elliptic PDEs (see [Hac]), and even analyzing seemingly simple model problems often requires considerable effort. The widespread interest in multigrid techniques for hyperbolic problems is largely motivated by the Euler equations of fluid dynamics. A quantitative analysis of the convergence rates for this class of problems would be very important, but it has apparently long eluded researchers in the field, although successful multigrid codes for the Euler equations have been used for quite some time [HS]. The recent work [BY] (and references therein) highlights some of the difficulties involved.

Multigrid techniques have not been used before for our class of PDEs, and we cannot give general convergence results either. However, we will gain some insight on convergence behavior of multigrid techniques from analyzing model problems. An upwind discretization is chosen to exploit the fact that the PDEs have same principal part, and appropriate prolongation operators are discussed to deal with the periodicity of the problem. We will then be able to prove that the convergence of the two-grid method on a scalar problem is $O(h)$. The same result holds true for the V-cycle multigrid method for this same problem. We will also give convergence estimates for the multigrid iteration on constant-coefficients two-dimensional problems. To this end, we use Fourier techniques, in much the same spirit as of [Hac, §8.1.2].

*Received by the editors September 16, 1992; accepted for publication (in revised form) September 21, 1993. This work was supported in part by National Science Foundation grant DMS-9104564.

†School of Mathematics, Georgia Institute of Technology, Atlanta, Georgia 30332 (dieci@math.gatech.edu). This work was initiated while the author was visiting the Institut fur Angewandte Mathematik, Universitat Heidelberg. The kind hospitality provided by Professor W. Jager and the support of the SFB 123 are gratefully acknowledged.

‡Institut fur Angewandte Mathematik, Universitat Heidelberg, Heidelberg, D6900 Germany (bader@gaia.iwr.uni-heidelberg.de).

An outline of the paper is as follows. In §2 we review some background material and look at the problem to be solved. We also present some simple results which have an impact on the properties of the discretization. In §3 we discuss the chosen upwind discretization. For nonlinear PDEs, we explicitly look at the structure resulting when performing the discrete Newton method. In §4 we consider multigrid algorithms, discuss prolongation and restriction operators, consider both point and block smoothings, and show $\mathcal{O}(h)$-convergence for a scalar variable-coefficients problem. We also give some convergence estimates for multigrid iteration on a constant-coefficients two-dimensional problem. We illustrate the theory with numerical results for linear problems. In §5 we present numerical results for two physically important nonlinear problems: the forced Josephson junction and a system of two coupled oscillators. In §6 we give conclusions.

2. Background. The problem to be solved. We consider general autonomous dynamical systems, in which a subset of the system variables are angular coordinates. More precisely, we first consider linear systems

$$\dot{\boldsymbol{\theta}} = \mathbf{f}(\boldsymbol{\theta}), \qquad \dot{\mathbf{v}} = -C(\boldsymbol{\theta})\mathbf{v} + \mathbf{g}(\boldsymbol{\theta}), \tag{2.1}$$

where $\boldsymbol{\theta}(t) = (\theta_1(t), \ldots, \theta_p(t))^T \in T^p$, $\mathbf{v}(t) = (v_1(t), \ldots, v_q(t))^T \in \mathbb{R}^q$, and T^p is the standard p-torus: $T^p = \{\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p) : \theta_i \in [0, 2\pi], (\mathrm{mod} 2\pi)\}$. The functions $\mathbf{f}(\boldsymbol{\theta}) = (f_1(\boldsymbol{\theta}), \ldots, f_p(\boldsymbol{\theta}))^T : T^p \to \mathbb{R}^p$, $C(\boldsymbol{\theta}) = (c_{ij}(\boldsymbol{\theta}))_{i,j=1}^q : T^p \to \mathbb{R}^{q\times q}$, and $\mathbf{g}(\boldsymbol{\theta}) = (g_1(\boldsymbol{\theta}), \ldots, g_p(\boldsymbol{\theta}))^T : T^p \to \mathbb{R}^q$ are assumed to be periodic in each angular coordinate and in C^r, $r \geq 2$. We also make the following positive-definiteness assumption on the coefficient matrix $C(\boldsymbol{\theta})$ (this implies attractivity of the invariant torus):

$$C(\boldsymbol{\theta}) \geq \eta I, \quad \eta > 0 \quad (\text{i.e., } \mathbf{x}^T C(\boldsymbol{\theta})\mathbf{x} > 0, \ \forall \mathbf{x} \neq \mathbf{0}) \quad \forall\, \boldsymbol{\theta} \in T^p, \tag{2.2}$$

and we are interested in determining a manifold $\mathcal{M}$, diffeomorphic to T^p, hence a torus, of the form

$$\mathcal{M} = \{(\boldsymbol{\theta}, \mathbf{u}(\boldsymbol{\theta})) : \boldsymbol{\theta} \in T^p\}, \qquad \mathbf{u} : T^p \to \mathbb{R}^q, \tag{2.3}$$

invariant under the flow of (2.1). As we know [K], if $\mathbf{u}$ is a C^1-function, then $\mathcal{M}$ is an invariant torus under the flow of (2.1) if and only if $\mathbf{u}$ satisfies the following periodic system of PDEs:

$$\sum_{\nu=1}^{p} \frac{\partial \mathbf{u}}{\partial \theta_\nu}(\boldsymbol{\theta}) f_\nu(\boldsymbol{\theta}) + C(\boldsymbol{\theta})\mathbf{u}(\boldsymbol{\theta}) = \mathbf{g}(\boldsymbol{\theta}), \qquad \boldsymbol{\theta} \in T^p. \tag{2.4}$$

Generally, we consider nonlinear systems of the following form:

$$\dot{\boldsymbol{\theta}} = \mathbf{f}(\boldsymbol{\theta}, \mathbf{v}), \qquad \dot{\mathbf{v}} = \mathbf{g}(\boldsymbol{\theta}, \mathbf{v}), \tag{2.5}$$

with $\boldsymbol{\theta}$, $\mathbf{v}$, and T^p as before, and $\mathbf{f} : T^p \times \mathbb{R}^q \to \mathbb{R}^p$, $\mathbf{g} : T^p \times \mathbb{R}^q \to \mathbb{R}^q$, C^r-functions, periodic in each angular coordinate. Again, if we require a manifold $\mathcal{M}$, as in (2.3), to be invariant under the flow of (2.5), and if $\mathbf{u}$ is a C^1-function, it must satisfy the following periodic nonlinear system of PDEs with same principal part:

$$\sum_{\nu=1}^{p} \frac{\partial \mathbf{u}}{\partial \theta_\nu}(\boldsymbol{\theta}) f_\nu(\boldsymbol{\theta}, \mathbf{u}) = \mathbf{g}(\boldsymbol{\theta}, \mathbf{u}), \qquad \boldsymbol{\theta} \in T^p. \tag{2.6}$$

A formal linearization of (2.6) around $\bar{\mathbf{u}}(\boldsymbol{\theta})$ gives for the correction $\mathbf{v}(\boldsymbol{\theta})$ the linear PDE

$$\sum_{\nu=1}^{p} \frac{\partial \mathbf{v}}{\partial \theta_\nu} f_\nu(\boldsymbol{\theta}, \bar{\mathbf{u}}) + \left[\sum_{\nu=1}^{p} \frac{\partial \bar{\mathbf{u}}}{\partial \theta_\nu} \cdot \nabla f_\nu - \frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right](\boldsymbol{\theta}, \bar{\mathbf{u}})\mathbf{v} = \mathbf{g}(\boldsymbol{\theta}, \bar{\mathbf{u}}) - \sum_{\nu=1}^{p} \frac{\partial \bar{\mathbf{u}}}{\partial \theta_\nu} f_\nu(\boldsymbol{\theta}, \bar{\mathbf{u}}), \tag{2.7}$$

where $\nabla f_\nu = (\frac{\partial f_\nu}{\partial u_1}, \ldots, \frac{\partial f_\nu}{\partial u_q})(\boldsymbol{\theta}, \bar{\mathbf{u}})$ and $\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$ is the Jacobian of $\mathbf{g}$. In analogy with (2.2), we will require that

$$\left[\sum_{\nu=1}^{p} \frac{\partial \bar{\mathbf{u}}}{\partial \theta_\nu} \cdot \nabla f_\nu - \frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right](\boldsymbol{\theta}, \bar{\mathbf{u}}) \geq \eta I, \quad \eta > 0, \quad \forall \boldsymbol{\theta} \in T^p, \tag{2.8}$$

hold for $\bar{\mathbf{u}}(\boldsymbol{\theta})$ in a neighborhood of $\mathbf{u}^*(\boldsymbol{\theta})$, the exact invariant torus solution of (2.6).

It is well understood [S] that if the above linearization procedure is taken as the basis for an iterative process (continuous Newton's method), then at each iteration there is a potential loss of differentiability for the new iterate. To see it, it suffices to consider the one-dimensional PDE (an ordinary differential equation (ODE) on the circle)

$$u_\theta f(\theta, u) = g(\theta, u), \qquad \theta \in T^1, \qquad u(0) = u(2\pi). \tag{2.9}$$

The continuous Newton's method would read: given u^0, find u^{n+1}, the periodic solution of

$$\begin{aligned} u_\theta^{n+1} f(\theta, u^n) + (u_\theta^n f_u - g_u)(\theta, u^n)u^{n+1} &= g(\theta, u^n) + (u_\theta^n f_u - g_u)(\theta, u^n)u^n, \\ \theta \in T^1, \qquad n &= 0, 1, \ldots. \end{aligned} \tag{2.10}$$

Now, if f, g, and u^0 are all C^r-functions, then we should expect the right-hand side of (2.10) to be only a C^{r-1}-function; hence u^1 can only be expected to be a C^{r-1}-function, and so forth. The trouble is given by the term

$$u_\theta^0 f_u, \qquad \sum_{\nu=1}^{p} \frac{\partial \bar{\mathbf{u}}}{\partial \theta_\nu} \cdot \nabla f_\nu$$

in (2.7). Clearly, if the functions f_ν are independent of $\mathbf{u}$, then the process is well defined and no loss of differentiability is involved. From the practical viewpoint, we first discretize the nonlinear PDE and then use a discrete Newton's process; see §3. It is our impression that the above-outlined difficulty is circumvented by our discretization at the discrete level, but that the convergence properties of the discrete Newton's process can ultimately be affected by the form of the function $f(\theta, u)$ in a similar way to what happens at the continuous level. In our experiments, we have observed convergence to be quadratic throughout.

Condition (2.2), or (2.8), is a stringent assumption, but it is instructive to appreciate what can go wrong when it is violated. We restrict ourselves to scalar one- and two-dimensional problems. First, consider (2.9), and let $u^*(\theta)$ be a smooth isolated solution. We want the variational equation around u^*:

$$v_\theta f(\theta, u^*) + [u_\theta^* f_u(\theta, u^*) - g_u(\theta, u^*)]v = 0, \qquad v(0) = v(2\pi),$$

to only have the trivial torus as a solution: $v = 0$. Now, this is certainly guaranteed if the term in brackets has a constant sign, say > 0. In fact, let $\theta(t, \theta_0)$ be the characteristics

$$\begin{aligned} \dot{\theta} &= a(\theta), \qquad a(\theta) := f(\theta, u^*(\theta)), \\ \theta(0) &= \theta_0, \qquad \theta_0 \in [0, 2\pi]; \end{aligned}$$

then $v(t) = v_0 \exp(-\int_0^t c(\theta(t,\theta_0))\,dt)$, $c(\theta) := u^*_\theta f_u(\theta, u^*) - g_u(\theta, u^*)$, and this has the unique solution $v = 0$ when $c(\theta) > 0$. The same situation occurs for scalar two-dimensional problems of the form $u_{\theta_1} + u_{\theta_2} f(\theta_1, \theta_2, u) = g(\theta_1, \theta_2, u)$, subject to the usual periodicity conditions. These problems are important, as they arise from nonautonomous second-order ODEs with periodic forcing term (forced oscillators). By taking the variational equation around $u^*(\theta_1, \theta_2)$ we obtain

$$
\begin{aligned}
& v_{\theta_1} + v_{\theta_2} F(\theta_1, \theta_2) = G(\theta_1, \theta_2) v, \\
& F := f(\theta_1, \theta_2, u^*), \\
& G := g_u(\theta_1, \theta_2, u^*) - u^*_{\theta_2} f_u(\theta_1, \theta_2, u^*).
\end{aligned} \tag{2.11}
$$

We want $v = 0$ to be the only periodic solution. This is guaranteed if G does not change sign. To see it, we can use the following argument. Consider (2.11) subject to periodic boundary conditions (BCs) in θ_2 only: $v(0, \theta_2) = w(y)$, $w(y) = w(y + 2\pi)$. Then we integrate along characteristics $d\theta_2/d\theta_1 = F(\theta_1, \theta_2)$, $\theta_2(0) = \alpha \in [0, 2\pi]$, to obtain $\theta_2(\theta_1, \alpha)$; the total derivative reads $dv/d\theta_1 = G(\theta_1, \theta_2(\theta_1, \alpha))v$, so we have $v = v(\theta_1, \theta_2(\theta_1, \alpha)) = c \cdot \exp \int_0^{\theta_1} G(\theta_1, \theta_2(\theta_1, \alpha))d\theta_1$. To have v periodic in θ_1 we must have $w(\alpha) = v(\theta_1 = 0) = v(\theta_1 = 2\pi) = w(\alpha) \exp \int_0^{2\pi} G(\theta_1, \theta_2(\theta_1, \alpha))d\theta_1$. The dependence on α is only fictitious, since $\partial\theta_2/\partial\alpha$ satisfies

$$
\frac{d}{d\theta_1}\left(\frac{\partial\theta_2}{\partial\alpha}\right) = F_{\theta_2}\frac{\partial\theta_2}{\partial\alpha}, \qquad \frac{\partial\theta_2}{\partial\alpha}(0) = 1,
$$

with solution $(\partial\theta_2/\partial\alpha)(\theta_1) = \exp \int_0^{\theta_1} F_{\theta_2} d\theta_1$. By letting $\gamma(\alpha) = \theta_2(2\pi, \alpha)$, then $\gamma(\alpha) = \exp \int_0^{2\pi} F_{\theta_2} d\theta_1 > 0$, γ is invertible, and $\alpha = \gamma^{-1}(\theta_2(2\pi))$. Thus, we only have the trivial solution when G does not change sign; otherwise we might have a nontrivial null space. To summarize, if $C(\boldsymbol{\theta}) > 0$ is not verified, we might not have an isolated solution.

Remark. A given dynamical system does not generally arise in the form (2.11), which we have assumed. Often the problem of finding a proper parametrization for a given problem is the hardest task. In theory, however, this is always locally possible (see [Hal]). In practice, we will show examples in §5.

3. Discretization. For the linear system (2.4), we maintain the choice made in [DB1]. Let $h_\nu = 2\pi/N_\nu$, $\nu = 1, \ldots, p$, $\mathbf{h} = (h_1, \ldots, h_p)$, and let T^p_h be the discrete analog of the torus T^p. T^p_h is a periodic grid with $N = N_1 \ldots N_p$ mesh points. Let $\mathbf{u}_h, \mathbf{u}_h : T^p_h \to \mathbb{R}^q$, be the unknown grid function, and replace (2.4) by the periodic difference equations of *upwinding type*

$$
\sum_{\nu=1}^{p}\left[f_\nu(\boldsymbol{\theta}_h) D_{0\nu}\mathbf{u}_h - \frac{1}{2}h_\nu \sigma_\nu(f_\nu(\boldsymbol{\theta}_h)) D_{+\nu}D_{-\nu}\mathbf{u}_h\right] + C(\boldsymbol{\theta}_h)\mathbf{u}_h = \mathbf{g}(\boldsymbol{\theta}_h), \qquad \boldsymbol{\theta}_h \in T^p_h, \tag{3.1}
$$

where $D_{+\nu}D_{-\nu} = \frac{1}{h_\nu^2}(E_\nu - 2I + E_\nu^{-1})$, $(E_\nu \mathbf{u}_h)(\boldsymbol{\theta}_h) := \mathbf{u}_h(\ldots, (j_\nu + 1)h_\nu, \ldots)$, $(j_\nu + 1) \in \mathbb{Z} \bmod N_\nu$, $D_{+\nu} = \frac{1}{h_\nu}(E_\nu - I)$, $D_{-\nu} = \frac{1}{h_\nu}(I - E_\nu^{-1})$, and $D_{0\nu} = \frac{1}{2}(D_{+\nu} + D_{-\nu})$. The functions $\sigma_\nu(f_\nu(\boldsymbol{\theta}_h))$ are the *smoothing functions* and are defined as

$$
\sigma_\nu(f_\nu(\boldsymbol{\theta}_h)) := \left(\epsilon_\nu + f_\nu(\boldsymbol{\theta}_h)^2\right)^{1/2}, \qquad \epsilon_\nu \geq 0, \tag{3.2}
$$

where ϵ_ν are the *smoothing parameters*. The choice of $\epsilon_\nu = 0$, for all ν, corresponds to what is known as *classical upwind*. Obviously, $\sigma_\nu \geq |f_\nu|$ with strict inequality when $\epsilon_\nu > 0$. As

seen in [DB1], this upwinding discretization leads to a block M-matrix structure, and hence to stability and convergence results.

While the upwind discretization theory for linear problems is fairly complete, the theory for the nonlinear case is not. We replace (2.6) by the following periodic difference equations of *upwinding type*:

$$\sum_{\nu=1}^{p}\left[f_\nu(\boldsymbol{\theta}_h, \mathbf{u}_h) D_{0\nu}\mathbf{u}_h - \frac{1}{2}h_\nu \sigma_\nu(f_\nu(\boldsymbol{\theta}_h, \mathbf{u}_h)) D_{+\nu}D_{-\nu}\mathbf{u}_h\right] = \mathbf{g}(\boldsymbol{\theta}_h, \mathbf{u}_h), \quad \boldsymbol{\theta}_h \in T_h^p, \tag{3.3}$$

where the smoothing functions σ_ν are defined as

$$\sigma_\nu(f_\nu(\boldsymbol{\theta}_h, \mathbf{u}_h)) := \left(\epsilon_\nu + f_\nu(\boldsymbol{\theta}_h, \mathbf{u}_h)^2\right)^{1/2}, \qquad \epsilon_\nu \geq 0. \tag{3.4}$$

To solve the nonlinear system (3.3) we use Newton's method. To this end, it is imperative that the σ_ν be differentiable functions (with respect to f_ν), and hence that $\epsilon_\nu > 0$ if f_ν change sign. With this in mind, a formal Newton iteration on (3.3) around a guess $\mathbf{u}_h^0$ gives for the correction $\mathbf{v}_h$, $\mathbf{v}_h : T_h^p \to \mathbb{R}^q$, the periodic linear system

$$\begin{aligned}
&\sum_{\nu=1}^{p}\left[f_\nu\, D_{0\nu}\mathbf{v}_h - \frac{1}{2}h_\nu \sigma_\nu(f_\nu) D_{+\nu}D_{-\nu}\mathbf{v}_h\right] \\
&\quad + \left\{-\frac{\partial \mathbf{g}}{\partial \mathbf{u}} + \sum_{\nu=1}^{p}\left[D_{0\nu}\mathbf{u}_h^0 - \frac{1}{2}h_\nu \sigma_\nu'(f_\nu) D_{+\nu}D_{-\nu}\mathbf{u}_h^0\right]\cdot \nabla f_\nu\right\}\mathbf{v}_h \\
&\quad = \mathbf{g} - \sum_{\nu=1}^{p}\left[f_\nu D_{0\nu}\mathbf{u}_h^0 - \frac{1}{2}h_\nu \sigma_\nu(f_\nu) D_{+\nu}D_{-\nu}\mathbf{u}_h^0\right],
\end{aligned} \tag{3.5}$$

where all quantities are evaluated at $(\boldsymbol{\theta}_h, \mathbf{u}_h^0)$, $\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$ and ∇f_ν are as in (2.7), and

$$\sigma_\nu'(f_\nu(\boldsymbol{\theta}, \mathbf{u})) = \frac{f_\nu(\boldsymbol{\theta}, \mathbf{u})}{\sigma_\nu(f_\nu(\boldsymbol{\theta}, \mathbf{u}))}.$$

Remark. The choice of σ_ν as in (3.4) guarantees that

$$|\sigma_\nu'| \leq 1 \quad \forall f_\nu, \epsilon_\nu, \qquad |\sigma_\nu''| \leq \frac{1}{\sqrt{\epsilon_\nu}}, \tag{3.6}$$

and this bound is achieved when f_ν changes sign.

Consider the term in braces in (3.5). By ordering the unknowns reverse-lexicographically and keeping the components at each grid point together, this is a $q \times q$ matrix block which goes on the diagonal of the $qN \times qN$ matrix associated with (3.5). We want this $q \times q$ matrix to be positive definite. This is actually to be expected for h_ν small enough, because by using Taylor's formula with remainder in integral form, and by letting $\mathbf{u}^0(\boldsymbol{\theta})$ correspond to the grid function $\mathbf{u}_h^0$, we have

$$\frac{\partial \mathbf{u}^0}{\partial \theta_\nu}(\boldsymbol{\theta}) = D_{0\nu}\mathbf{u}^0(\boldsymbol{\theta}) + \frac{h_\nu^2}{2}\mathbf{T}_\nu, \quad |\mathbf{T}_\nu| \leq \left\|\frac{\partial^2 \mathbf{u}^0}{\partial \theta_\nu^2}\right\|, \quad D_{+\nu}D_{-\nu}\mathbf{u}^0(\boldsymbol{\theta}) \leq \left\|\frac{\partial^2 \mathbf{u}^0}{\partial \theta_\nu^2}\right\|.$$

Therefore, the term in braces can be rewritten as

$$\left[\sum_{\nu=1}^{p}\frac{\partial \mathbf{u}^0}{\partial \theta_\nu}\cdot \nabla f_\nu - \frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right](\boldsymbol{\theta}_h, \bar{\mathbf{u}}_h^0) + \mathbf{E}(\boldsymbol{\theta}_h, \mathbf{u}_h^0),$$

with

$$\mathbf{E}(\theta_h, \mathbf{u}_h^0) = \left\{ \sum_{\nu=1}^{p} \left[\frac{1}{2} h_\nu \mathbf{T}_\nu - \frac{1}{2} h_\nu \sigma_\nu'(f_\nu) D_{+\nu} D_{-\nu} \mathbf{u}_h^0 \right] \cdot \nabla f_\nu \right\} (\theta_h, \mathbf{u}_h^0)$$

and because of (3.6) we have $\|\mathbf{E}\| \le |\mathbf{h}|_\infty \cdot \|D^2\mathbf{u}\| \cdot \|\nabla \mathbf{f}\|$, where we have set

$$\|\nabla \mathbf{f}\| = \max_{1\le\nu\le p} \max_{\in T^p} |\nabla f_\nu|.$$

Thus, we have that the term in braces gives a consistent approximation of the condition (2.8), and in particular the discrete system inherits the block M-matrix structure [DB1]. Because of Theorem 2.6 in [DB1], this makes the Newton iterates well defined, and allows for a Newton attraction-type result. However, this is not the full story. Some of the difficulties of the continuous Newton's method carry over at the discrete level as well. Since all the basic difficulties can already arise in the one-dimensional case, we limit our discussion once again to the PDE (2.9). In this case, the discretization (3.3) gives the (periodic) nonlinear system

$$F_i(\mathbf{u}) := \frac{u_{i+1} - u_{i-1}}{2h} f_i - \frac{h}{2} \sigma_i \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} - g_i = 0, \quad i = 1, \dots, N,$$

where we have used the notation $\mathbf{u} = (u_i, i = 1, \dots, N)$, $u_i = u(\theta_i)$, $\theta_i = \frac{(i-1)}{N} 2\pi$, $\theta_{N+1} = \theta_1$, $f_i = f(\theta_i, u_i)$, $g_i = g(\theta_i, u_i)$, and $\sigma_i = \sigma(f_i) = (\epsilon + f_i^2)^{1/2}$, $\epsilon > 0$. For this discrete nonlinear system $\mathbf{F}(\mathbf{u}) = \mathbf{0}$, it is easy to argue that there is a $\mathbf{u}_0 \in \mathbb{R}^N$ such that $F'(\mathbf{u}_0)$ is nonsingular (F' is the Jacobian of $\mathbf{F}$), and to give bounds for $\|F'(\mathbf{u}_0)^{-1}\|$ and $\|F'(\mathbf{u}_0)^{-1}\mathbf{F}(\mathbf{u}_0)\|$. These are the first two steps in a typical Newton attraction theorem (see [OR]). To estimate the Lipschitz constant for the Jacobian we need a bound of the type $\|F'(\mathbf{u}) - F'(\mathbf{v})\| \le \gamma \|\mathbf{u} - \mathbf{v}\|$. To see how this bound reads we can use the mean-value theorem:

$$\|F'(\mathbf{u}) - F'(\mathbf{v})\| \le \sup_{0\le t\le 1} \|F''(\mathbf{v} + t(\mathbf{u} - \mathbf{v}))\| \cdot \|\mathbf{u} - \mathbf{v}\|.$$

As is well known,

$$\sup_{0\le t\le 1} \|F''(\mathbf{z})\| = \sup_{0\le t\le 1} \sup_{\|\mathbf{l}\|=\|\mathbf{k}\|=1} \|F''(\mathbf{z})\mathbf{l}\,\mathbf{k}\|, \qquad \mathbf{l}, \mathbf{k} \in \mathbb{R}^N,$$

$$(F''(\mathbf{z})\mathbf{l}\mathbf{k})^T = (\mathbf{k}^T H_1 \mathbf{l}, \dots, \mathbf{k}^T H_N \mathbf{l}),$$

where $H_i(\mathbf{z})$ is the Hessian relative to F_i. The concrete representation of $H_i(\mathbf{z})$ has at most five nonzero entries in position $(i-1, i)$, $(i, i-1)$, (i, i), $(i, i+1)$, $(i+1, i)$, subject to the usual periodicity structure; that is, the nonzero part of H_i is

$$H_i(\mathbf{z}) = \begin{pmatrix} 0 & F_{i,i-1,i} & 0 \\ F_{i,i,i-1} & F_{i,i,i} & F_{i,i,i+1} \\ 0 & F_{i,i+1,i} & 0 \end{pmatrix}, \quad i = 1, \dots, N \pmod N,$$

where we have set $F_{i,k,j}$ as the second partial of F_i with respect to θ_k and θ_j. Explicitly, we have (omitting the obvious θ_i dependence)

$$F_{i,i-1,i} = F_{i,i,i-1} = \frac{-f_u(z_i)(1 + \sigma_i')}{2h}, \qquad \sigma_i' = \sigma'(f(z_i)),$$

$$F_{i,i,i+1} = F_{i,i+1,i} = \frac{f_u(z_i)(1 - \sigma_i')}{2h}, \qquad \sigma_i'' = \sigma''(f(z_i)),$$

$$F_{i,i,i} = -g_{uu}(z_i) + \frac{2\sigma_i' f_u(z_i)}{h} + f_{uu}(z_i) \frac{z_{i+1} - z_{i-1}}{2h}$$

$$-(\sigma_i'' f_u^2(z_i) + \sigma_i' f_{uu}(z_i)) \frac{z_{i+1} - 2z_i + z_{i-1}}{2h}.$$

So, by using the above definitions and (3.6), we can obtain a bound for γ of the type

$$\gamma \leq \sup_{0\leq t\leq 1}\sup_{1\leq i\leq n}\left\{\frac{4}{h}|f_u(\cdot)| + |g_{uu}(\cdot)| + |f_{uu}(\cdot)|\left|\frac{(1-t)(v_{i+1}-v_{i-1})+t(u_{i+1}-u_{i-1})}{2h}\right| + \frac{h}{2}\left(\frac{1}{\sqrt{\epsilon}}|f_u^2(\cdot)| + |f_{uu}(\cdot)|\right)\cdot\left|\frac{(1-t)(v_{i+1}-2v_i+v_{i-1})+t(u_{i+1}-2u_i+u_{i-1})}{h^2}\right|\right\}, \tag{3.7}$$

where $(\cdot) = (v_i + t(u_i - v_i))$. The first point to observe is that for fixed h and $\epsilon > 0$, γ is a bounded quantity. This implies that the Newton iterates must converge quadratically provided the initial guess is sufficiently close to the discrete solution. This is why we indeed observe quadratic convergence in practice. However, the constant in the error estimate of the Newton iteration behaves essentially like $\|f_u\|/h$, and thus it gets larger for finer discretization unless $f(\theta, u)$ is independent of u. If we restrict the class of functions $f(\theta, u)$ to be of the type $f(\theta, u) = F_1(\theta) + \delta F_2(\theta, u)$, with δ small and $F_{2,u}$ bounded, then in practice we are hardly affected by this fact; otherwise it might affect the convergence of the Newton iterates. This same difficulty shows up when trying to enforce the bounds for a Newton–Kantorovich result. The other factor to note in (3.7) is the $\frac{1}{\sqrt{\epsilon}}$ term; this tells us that we should not let the smoothing parameter be too small (again, unless f_u is 0 or small). In our numerical implementations, we have carefully monitored the behavior of the Newton iterates on a fixed grid, and we have not observed any particular convergence difficulties, as we will report in §5.

4. Multigrid strategies for the linear systems. Here below, we consider applying multigrid techniques for our class of PDEs. We only consider V-cycle methods. We refer to [Hac] for a foundation of this approach, and for the terminology used.

In what follows, our arguments are motivated by the understanding of what happens when we use a classical upwind discretization for the PDEs (i.e., the smoothing parameters $\epsilon_i = 0$), and the coefficients in front of the first derivatives do not change sign, in particular, as far as the choice of prolongation operators is concerned. The reasoning below remains valid for smooth upwind discretization with small smoothing parameters, but not in general. We refer to Examples 1 and 2, below, for typical convergence behavior on PDEs not satisfying the restrictions we impose on the model problems.

First, consider the one-dimensional problem (the ODE on the circle)

$$u_\theta + c(\theta)u = g(\theta), \qquad c(\theta) > 0, \qquad u(0) = u(2\pi). \tag{4.1}$$

Discretize this problem by classical upwind, and obtain the linear system

$$A_h\mathbf{u}_h = (L_h - \gamma_h\mathbf{e}_1\mathbf{e}_N^T)\mathbf{u}_h = \mathbf{g}_h, \tag{4.2}$$

where L_h is the lower triangular matrix

$$L_h = \gamma_h\begin{pmatrix} d_1 & & & \\ -1 & d_2 & & \\ & \ddots & \ddots & \\ & & -1 & d_N \end{pmatrix},$$

and $\gamma_h = \frac{1}{h}$, $d_i = 1 + hc(x_i)$, $\mathbf{e}_1 = (1, 0, \ldots, 0)^T$, $\mathbf{e}_N = (0, \ldots, 0, 1)^T$. Perform the (forward) Gauss–Seidel iteration as a smoothing iteration: $L_h\mathbf{u}_h^{j+1} = \gamma_h\mathbf{e}_1\mathbf{e}_N^T\mathbf{u}_h^j + \mathbf{g}_h$,

$j = 0, 1, \ldots$. Then the iteration for this smoother can be written as $\mathbf{f}_h^{j+1} = S_h \mathbf{f}_h^j$, $\mathbf{f}_h^{j+1} := \mathbf{u}_h^{j+1} - \mathbf{u}_h^j$, $j = 0, 1, \ldots$, where the iteration matrix S_h is

$$S_h = \gamma_h L_h^{-1} \mathbf{e}_1 \mathbf{e}_N^T = \mathbf{w}_h \mathbf{e}_N^T = \begin{pmatrix} 0 & \cdots & 0 & w_h^1 \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & w_h^N \end{pmatrix} \quad \text{with} \quad w_h^i = (d_1 \ldots d_i)^{-1}. \tag{4.3}$$

LEMMA 4.1. *Let the coefficient $c(\theta)$ in (4.1) be sufficiently smooth, $c(\theta) \in C^2[0, 2\pi]$. Then the entries w_h^i in (4.3), $i = 1, \ldots, N$, satisfy the asymptotic error expansion*

$$w_h^i = w(\theta_{i+1}) + h v(\theta_{i+1}) + \mathcal{O}(h^2), \tag{4.4}$$

where the $\mathcal{O}(h^2)$ term is uniformly bounded for $h \le h_0$, and $w(\theta)$ and $v(\theta)$ solve the initial value problems (IVPs)

$$\begin{aligned} w' + c(\theta) w &= 0, & w(0) &= 1, \\ v' + c(\theta) v &= \tfrac{1}{2} w'', & v(0) &= 0. \end{aligned} \tag{4.5}$$

Proof. It suffices to realize that the w_h^i in (4.3) are exactly the same values given by the implicit Euler scheme with stepsize h on the IVP

$$w' + c(\theta) w = 0, \qquad w(0) = 1,$$

that is, $w_{i+1} = w_i / d_{i+1}$, $w_0 = 1$. Hence (e.g., see [G]), there exists an h_0 such that for all stepsizes $h \le h_0$ the above expansion holds. □

The iteration matrix of the two-grid scheme is given by

$$M_2^{\nu_1, \nu_2} = S^{\nu_2} (I - P_h A_{2h}^{-1} R_h A_h) S^{\nu_1}, \tag{4.6}$$

where ν_1 and ν_2 are the numbers of pre- and postsmoothings performed. We consider only $M_2^{1,0}$ from now on. In (4.6), R_h and P_h are the *restriction* and *prolongation* operators, respectively. The simplest choice of a restriction operator is the *injection operator*

$$R_h = \begin{pmatrix} 1 & 0 & & & & \\ & & 1 & 0 & & \\ & & & & \ddots & \ddots \\ & & & & & 1 \quad 0 \end{pmatrix};$$

however, taking the periodicity of (4.1) into account, a better choice for the restriction operator from the grid-size h to the grid-size $2h$ is given by the *linear restriction*

$$R_h = \frac{1}{4} \begin{pmatrix} 2 & 1 & & & & 1 \\ & 1 & 2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 2 & 1 \end{pmatrix}. \tag{4.7}$$

Substitution of (4.7) into $M_2^{1,0}$, and use of rank-one update formulas, gives the following explicit form:

$$M_2^{1,0} = \left(\mathbf{w}_h - \frac{1 - w_h^N}{1 - w_{2h}^{N/2}} P_h \mathbf{w}_{2h} \right) \mathbf{e}_N^T. \tag{4.8}$$

Using (4.4) gives

$$\frac{1-w_h^N}{1-w_{2h}^{N/2}} = 1 + h\frac{v(2\pi)}{1-w(2\pi)} + \mathcal{O}(h^2).$$

Hence, as long as $w(2\pi) = \exp\left(-\int_0^{2\pi} c(s)ds\right)$ is bounded away from unity (which is certainly true when $c(\theta) > 0$), we can consider only the term $\mathbf{w}_h - P_h\mathbf{w}_{2h}$ in order to obtain bounds on the norm of (4.8), for h sufficiently small. The crucial step in bounding (4.8) is the choice of the prolongation operator. The standard choice [Hac] corresponding to the restriction operator (4.7) is the *linear interpolant*

$$P_h = \frac{1}{2}\begin{pmatrix} 2 & & & & & \\ 1 & 1 & & & & \\ & 2 & & & & \\ & 1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & 2 & \\ 1 & & & & 1 & \end{pmatrix}. \tag{4.9}$$

Applying once more the asymptotic result from Lemma 4.1 shows that all but the (N, N)-entry of $M_2^{1,0}$ are $\mathcal{O}(h)$. This last entry, for large positive $c(\theta)$, approaches $\frac{1}{2}$ instead of an $\mathcal{O}(h)$ quantity. This effect is due to the periodic boundary conditions. One way around this unpleasant effect is a modification of the prolongation operator. Consider the choice

$$P_h = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & 1 & \end{pmatrix}. \tag{4.10}$$

The prolongation operator (4.10) is based on a simple constant extrapolation. Hence, we call it *piecewise constant prolongation*. The key difference in terms of convergence is that for (4.10) the norm of $M_2^{1,0}$ is $\mathcal{O}(h)$, regardless of $c(\theta)$, while for (4.9) this is not always the case.

To analyze the three-level scheme, we replace the exact solution on the $2h$-grid by further restriction and smoothing steps, and then solve exactly on the $4h$-grid. A simple calculation shows that the resulting iteration matrix $M_3^{1,0}$ is

$$M_3^{1,0} = \left(I - P_h L_{2h}^{-1} R_h A_h - P_h P_{2h} A_{4h}^{-1} R_{2h}(I - A_{2h}L_{2h}^{-1})R_h A_h\right)S_h. \tag{4.11}$$

Applying formulas for rank-one updates, and recalling (4.2)–(4.3), we obtain (with obvious notation)

$$M_3^{1,0} = \left(\mathbf{w}_h - (1-w_h^N)P_h\mathbf{w}_{2h} - \frac{w_h^N(1-w_{2h}^{N/2})}{1-w_{4h}^{N/4}}P_hP_{2h}\mathbf{w}_{4h}\right)\mathbf{e}_N^T. \tag{4.12}$$

Now, as in the two-grid case, we can use the asymptotic expansion from Lemma 4.1. This gives

$$1 - w_h^N = 1 - w(2\pi) - hv(2\pi) + \mathcal{O}(h^2),$$

$$\begin{aligned} \frac{w_h^N(1 - w_{2h}^{N/2})}{1 - w_{4h}^{N/4}} &= w_h^N \left(1 + 2h\frac{v(2\pi)}{1 - w(2\pi)} + \mathcal{O}(h^2)\right) \\ &= w(2\pi) + hv(2\pi)\frac{1 + w(2\pi)}{1 - w(2\pi)} + \mathcal{O}(h^2). \end{aligned} \tag{4.13}$$

It is now again imperative to use the prolongation operator (4.10) for P_h (and similarly for P_{2h}) in order to obtain

$$\begin{aligned} \mathbf{w}_h - P_h\mathbf{w}_{2h} &= \mathcal{O}(h), \\ \mathbf{w}_h - P_hP_{2h}\mathbf{w}_{4h} &= \mathcal{O}(h). \end{aligned}$$

Substituting these estimates into (4.12), from (4.13) we obtain the desired result:

$$M_3^{1,0} = \mathcal{O}(h).$$

Extension to an arbitrary number of levels is clearly possible, the chief theoretical difficulty being the validity of the expansion (4.4), which only holds for h small enough.

Remarks.

(1) The same results are obtained if we let u in (4.1) be in $\mathbb{R}^q$, $q > 1$.

(2) The assumption $c(\theta) > 0$ in (4.1) is not really needed. Clearly, it suffices that $\int_0^{2\pi} c(\theta)d\theta \neq 0$ (see the discussion at the end of §2).

(3) The previous result holds basically unchanged if we use the cyclic Gauss–Seidel or the incomplete LU (ILU) iteration as a smoother (see [DB1]). In fact, in these cases, we would obtain "w_h^i," which are only $\mathcal{O}(h)$ perturbations of the w_h^i obtained with the forward Gauss–Seidel smoother.

(4) If in (4.1) we had $f(\theta)u_\theta$, and $f(\theta)$ never vanished, the results above would still be valid. In particular, if $f(\theta) > 0$, no modifications are needed, while if $f(\theta) < 0$ the results apply to the choice of backward Gauss–Seidel iteration as a smoother, with the obvious modification of the prolongation operator (4.10). As a consequence, if $f(\theta)$ never vanishes, the results are valid for the cyclic Gauss–Seidel and for ILU as smoothers.

(5) The situation is more complicated if in (4.1) we have $f(\theta)u_\theta$, and $f(\theta)$ changes sign. In general, the zeroes of $f(\theta)$ inhibit a direct comparison of discretizations on different grids.

(6) For completeness, we describe here the generalization of the piecewise constant prolongation we have adopted in the general case of having $f(\theta)u_\theta$ in (4.1), and f not necessarily positive. For points of both the coarse and fine grids, we maintain the old function values. If $x : \theta_i < x < \theta_{i+1}$ is a new grid point, then we take $u_h(x) = u_{2h}(\theta_i)$ if both $f(\theta_i)$ and $f(\theta_{i+1})$ are positive, we take $u_h(x) = u_{2h}(\theta_{i+1})$ if both $f(\theta_i)$ and $f(\theta_{i+1})$ are negative, and we take $u_h(x) = \frac{1}{2}(u_{2h}(\theta_i) + u_{2h}(\theta_{i+1}))$ otherwise. The rationale is again based on the classical upwind discretization, to keep the prolonged function moving along with the discrete flow.

Next, consider two-dimensional scalar PDEs of the form

$$f_1(\boldsymbol{\theta})\frac{\partial u}{\partial \theta_1} + f_2(\boldsymbol{\theta})\frac{\partial u}{\partial \theta_2} + c(\boldsymbol{\theta})u = g(\boldsymbol{\theta}), \qquad \boldsymbol{\theta} = (\theta_1, \theta_2). \tag{4.14}$$

Also for this case, we understand better the situation arising when a classical upwind discretization is performed and the functions $f_i(\boldsymbol{\theta})$ do not change sign. The discussion below,

and our implementation choices, are also motivated by the desire to improve the convergence behavior of the simpler iterations (the smoothers) in case $0 < c(\boldsymbol{\theta}) < 1$ (in case $c(\boldsymbol{\theta})$ is reasonably large, the basic iterations converge nicely).

For restrictions, we can consider the obvious injection operator, and the generalization of (4.7):

$$\mathbf{R}_h = \frac{1}{4}\begin{pmatrix} 2R_{h_2} & R_{h_2} & & & & R_{h_2} \\ & R_{h_2} & 2R_{h_2} & R_{h_2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & R_{h_2} & 2R_{h_2} & R_{h_2} \end{pmatrix}, \tag{4.15}$$

where R_{h_2} denotes the corresponding one-dimensional operator. In our notation the blocks are taken with respect to the θ_2 coordinate. As for prolongation operators, we considered generalizations of both (4.9) and (4.10). The generalization of (4.9), which we will still call the *linear interpolant* prolongation, is

$$\mathbf{P}_h := \frac{1}{2}\begin{pmatrix} 2P_{h_2} & & & & \\ P_{h_2} & P_{h_2} & & & \\ & 2P_{h_2} & & & \\ & P_{h_2} & P_{h_2} & & \\ & & \ddots & & \\ & & & 2P_{h_2} & \\ P_{h_2} & & & P_{h_2} & \end{pmatrix}, \tag{4.16}$$

with P_{h_2} given by (4.9). To generalize the *piecewise constant prolongation* (4.10), let us think of $f_1(\boldsymbol{\theta})$ and $f_2(\boldsymbol{\theta})$ being positive. In this case, we generalize (4.10) as

$$\mathbf{P}_h := \begin{pmatrix} P_{h_2} & & & \\ P_{h_2} & & & \\ & P_{h_2} & & \\ & P_{h_2} & & \\ & & \ddots & \\ & & & P_{h_2} \\ & & & P_{h_2} \end{pmatrix}, \tag{4.17}$$

where again P_{h_2} is the corresponding one-dimensional prolongation. When $f_1(\boldsymbol{\theta})$ and/or $f_2(\boldsymbol{\theta})$ are not positive, the piecewise constant prolongation $\mathbf{P}_h$ gets modified in a similar way, as explained in the previous Remark (6) for the one-dimensional case.

As smoothers, we either use pointwise relaxation schemes or circle block schemes. Let us call *point multigrid* and *block multigrid* the multigrid iterations which are obtained when using pointwise or circle block smoothings, respectively. (It is to be understood that when we have $q > 1$ radial coordinates, as for some of the examples below, the pointwise smoothing is a $(q \times q)$ block smoothing.) As pointwise smoothers, we considered damped Jacobi with damping factor equal to $\frac{1}{2}$, forward and cyclic Gauss–Seidel, and ILU iterations (see [DB1] for these schemes).

When doing a point multigrid iteration, if $f_1(\boldsymbol{\theta})$ and $f_2(\boldsymbol{\theta})$ have constant sign, then one should choose $\mathbf{P}_h$ as in (4.17), and P_{h_2} as in (4.10), because we have to deal with a doubly periodic problem and the arguments of the one-dimensional convergence analysis carry over to the two-dimensional case. This reasoning, of course, is amply confirmed by our numerical results.

As the convergence results for the basic iterative schemes predict a better convergence behavior for the circle block methods, we have also considered these as smoothers in the

multigrid context. In particular, we tested the circle block damped Jacobi, Gauss–Seidel, and cyclic Gauss–Seidel iterations. When such methods are applied as smoothers, we have a clear distinction in behavior with respect to the two coordinates. Hence it could be opportune to choose the prolongation operators correspondingly. Besides the above linear interpolant and piecewise constant prolongation operators, we have also tested the "mixed" prolongation operator (4.17) with P_{h_2} defined by (4.9) instead of (4.10).

As for convergence analysis of this block multigrid iteration, we again consider a model problem:

$$\frac{\partial u}{\partial \theta_1} + \frac{\partial u}{\partial \theta_2} + c(\boldsymbol{\theta})u = g(\boldsymbol{\theta}). \tag{4.18}$$

Classical upwind discretization with $\mathbf{h} = (h_1, h_2)$ gives the $N_1 \times N_2$ linear system

$$\mathbf{A}_h \mathbf{u}_h = \mathbf{g}_h, \qquad \mathbf{A}_h = \mathbf{L}_h - \frac{1}{h_1} E_1 E_{N_1}^T, \tag{4.19}$$

where $E_1^T = (I, 0, \ldots, 0)$, $E_{N_1}^T = (0, \ldots, 0, I)$, and

$$\mathbf{L}_h = \begin{pmatrix} A_1 & & & \\ -\frac{1}{h_1} I & A_2 & & \\ & \ddots & \ddots & \\ & & -\frac{1}{h_1} I & A_{N_1} \end{pmatrix}, \qquad A_i = \begin{pmatrix} d_{i1} & & & -\frac{1}{h_2} \\ -\frac{1}{h_2} & d_{i2} & & \\ & \ddots & \ddots & \\ & & -\frac{1}{h_2} & d_{iN_2} \end{pmatrix},$$

$i = 1, \ldots, N_1$, with diagonal entries $d_{ij} = \frac{1}{h_1} + \frac{1}{h_2} + c(\boldsymbol{\theta}_{ij})$. As in the one-dimensional case, the iteration matrix for the two-level multigrid method is of the form (4.6) with the above-defined block-matrices. Hence we find for the circle block (forward) Gauss–Seidel smoother the iteration matrix

(4.20)

$$\mathbf{S}_h = \frac{1}{h_1} \mathbf{L}_h^{-1} E_1 E_{N_1}^T = \mathbf{W}_h E_{N_1}^T = \begin{pmatrix} 0 & \cdots & 0 & W_1^h \\ \vdots & & \vdots & \vdots \\ 0 & \cdots & 0 & W_{N_1}^h \end{pmatrix} \quad \text{with} \quad W_i^h = \left(\prod_{j=1}^{i} h_1 A_j \right)^{-1}.$$

Using the Sherman–Morrison formula gives for the iteration matrix

$$\mathbf{M}_2^{1,0} = \left(\mathbf{W}_h - \mathbf{P}_h \mathbf{W}_{2h} (I - W_{N_1/2}^{2h})^{-1} R_{h_2} (I - W_{N_1}^h) \right) E_{N_1}^T. \tag{4.21}$$

An asymptotic expansion result for the matrices W_i^h is now not possible, because the matrices W_h^i are the companion matrices (or Greens functions) built from fundamental solutions of an initial boundary value problem corresponding to nonsmooth initial functions (delta functions as $h \to 0$), hence they fail to satisfy an asymptotic expansion. Nonetheless, something can still be inferred by further restricting to the constant-coefficients case and using spectral arguments. Hence, here below, we restrict ourselves to the case $c(\boldsymbol{\theta}) = c > 0$ in (4.18), and (for simplicity) we let $h_1 = h_2 = h$ (i.e., $N_1 = N_2 = N$, an even number). Also, we limit our discussion to the case of $\mathbf{P}_h$ in (4.21) given by (4.17).

LEMMA 4.2. *Let $c(\boldsymbol{\theta}) = c \geq 0$ in (4.18), and let $A_h = A_i$ in (4.19). Then, the matrix $B_h = (hA_h)^{-1}$ is circulant, and we have*

$$B_h = \frac{1}{1 - b^N} \begin{pmatrix} b & b^N & \cdots & b^3 & b^2 \\ b^2 & b & b^N & & b^3 \\ b^3 & b^2 & b & \ddots & \vdots \\ \vdots & & \ddots & \ddots & b^N \\ b^N & b^{N-1} & \cdots & b^2 & b \end{pmatrix}, \quad \text{with} \quad b = \frac{1}{2 + hc}. \tag{4.22}$$

The matrix B_h can be decomposed like

$$B_h = X_h \Lambda_h X_h^{-1}, \tag{4.23}$$

with eigenvalues

$$\Lambda_h = \operatorname{diag}(\lambda_j), \quad \lambda_j = (b^{-1} - \omega^{-j})^{-1}, \quad 0 \le j < N,$$

where $\omega = \cos(\frac{2\pi}{N}) + \sqrt{-1}\sin(\frac{2\pi}{N})$ *is the nth root of unity, and eigenvector matrices*

$$X_h = \left(\omega^{ij}\right)_{i,j=0,\ldots,N-1}, \qquad X_h^{-1} = \frac{1}{N}\left(\omega^{-ij}\right)_{i,j=0,\ldots,N-1}.$$

Using this, we have for the restriction operator (4.7) (R_{h_2} *in* (4.15), (4.21))

$$X_{2h}^{-1} R_h X_h = (D_1,\ D_2) \tag{4.24}$$

with

$$\begin{aligned} D_1 &= \frac{1}{2}\operatorname{diag}\left(1 + \cos\left(\frac{2\pi j}{N}\right)\right), \\ D_2 &= \frac{1}{2}\operatorname{diag}\left(1 - \cos\left(\frac{2\pi j}{N}\right)\right), \end{aligned} \qquad j = 0, \ldots, N/2 - 1, \tag{4.24a}$$

while for the injection-type restriction we have $D_1 = D_2 = I_{N/2}$. *For the prolongation operators* (4.9) *and* (4.10) (P_{h_2} *in* (4.17)), *we have*

$$X_h^{-1} P_h X_{2h} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix}, \tag{4.25}$$

where for (4.9) *the matrices* F_1, F_2 *are given by*

$$\begin{aligned} F_1 &= \frac{1}{2}\operatorname{diag}\left(1 + \cos\left(\frac{2\pi j}{N}\right)\right), \\ F_2 &= \frac{1}{2}\operatorname{diag}\left(1 - \cos\left(\frac{2\pi j}{N}\right)\right), \end{aligned} \qquad 0 \le j < N/2, \tag{4.25a}$$

and for (4.10) *we have*

$$\begin{aligned} F_1 &= \frac{1}{2}\operatorname{diag}\left(1 + \omega^{-j}\right), \\ F_2 &= \frac{1}{2}\operatorname{diag}\left(1 - \omega^{-j}\right), \end{aligned} \qquad 0 \le j < N/2. \tag{4.25b}$$

Proof. The proof is by direct verification. (Details can be found in [DB0].) □

Now, consider $\mathbf{M}_2^{1,0}$. Block-multiplying with X_h from the right and X_h^{-1} from the left gives for the block-rows of (4.21)

$$\Lambda_h^{ii} - \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \Lambda_{2h}^{i} (I - \Lambda_{2h}^{N/2})^{-1} (D_1,\ D_2)(I - \Lambda_h^N), \qquad 0 \le i < N/2, \tag{4.26}$$

where it is understood that $ii = 2i$ and $ii = 2i + 1$ for the even and odd block-rows. Given (4.21) and (4.26), we are left to estimate the spectral radius of the $N \times N$ block

$$\Lambda_h^N - \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \Lambda_{2h}^{N/2} (I - \Lambda_{2h}^{N/2})^{-1} (D_1,\ D_2)(I - \Lambda_h^N). \tag{4.27}$$

Since in (4.27) we essentially have a (2×2) block structure, we are left to consider the (2×2) matrices (with obvious notation)

$$C_j := \begin{pmatrix} c_{11}^{(j)} & c_{12}^{(j)} \\ c_{21}^{(j)} & c_{22}^{(j)} \end{pmatrix},$$

given by

$$C_j = \begin{pmatrix} (\lambda_j^{(1)})^N - (\lambda_j^{(2)})^{N/2} f_j^{(1)} \hat{d}_j^{(1)} & -(\lambda_j^{(2)})^{N/2} f_j^{(1)} \hat{d}_j^{(2)} \\ -(\lambda_j^{(2)})^{N/2} f_j^{(2)} \hat{d}_j^{(1)} & (\lambda_{j+N/2}^{(1)})^N - (\lambda_j^{(2)})^{N/2} f_j^{(2)} \hat{d}_j^{(2)} \end{pmatrix}, \quad j = 0, \ldots, N/2 - 1, \tag{4.28}$$

where $\left(\hat{D}_1,\ \hat{D}_2\right) = (I - \Lambda_{2h}^{N/2})^{-1} (D_1,\ D_2)(I - \Lambda_h^N)$. Estimation of the roots of these complex quadratic equations is not trivial.

For $j = 0$, because of (4.24)–(4.25), the product $c_{12}^{(j)} c_{21}^{(j)}$ is equal to zero for all $c \geq 0$ (if $c = 0$, a limiting argument is needed). Hence, for $j = 0$, we have that

$$\rho(C_0) \leq \max\left(|c_{11}^{(0)}|,\ |c_{22}^{(0)}|\right). \tag{4.29}$$

LEMMA 4.3. *Let $c(\theta) = c \geq 0$ in (4.18), and let $q = hc$. Then, we have*

$$\rho(C_0) \leq \min\left(\frac{1}{2},\ \frac{1}{(1+2q)^{N/2}}\right). \tag{4.30}$$

Proof. For $c_{11}^{(0)}$, we find

$$G(q) := |c_{11}^{(0)}| = \frac{(1+q)^N - (1+2q)^{N/2}}{(1+q)^N((1+2q)^{N/2} - 1)},$$

and an easy estimation gives

$$G(q) \leq \frac{1}{(1+2q)^{N/2}} \frac{1 - \frac{1}{(1+q)^{N/2}}}{1 - \frac{1}{(1+2q)^{N/2}}} \leq \frac{1}{(1+2q)^{N/2}} \quad \forall q \geq 0$$

(equality is actually obtained if we let $c \to \infty$). That $G(q) \leq 1/2$ is equivalent to $3 \leq (1+2q)^{N/2}\left(1 + \frac{2}{(1+q)^N}\right) =: g(q)$. But this is immediate, since $g(0) = 3$ and $g'(q) > 0$. For $c_{22}^{(0)}$ we have $c_{22}^{(0)} = \left(\frac{1}{3+q}\right)^N$, so that $|c_{22}^{(0)}| \leq 1/2$ is immediate and

$$\frac{1}{(3+q)^N} \leq \frac{1}{(1+2q)^{N/2}}$$

is clear. □

Now consider the general case $0 < j < N/2$. The following are useful identities.

LEMMA 4.4. *Let $c(\theta) = c \geq 0$ in (4.18). With obvious notation, we have the following identities, which are valid for all restrictions and prolongations considered:*

$$\begin{aligned} &\lambda_j^{(1)} = \overline{\lambda_{N-j}^{(1)}}, \qquad 0 \leq j \leq N, \\ &\lambda_{N/2-j}^{(1)} = \overline{\lambda_{N/2+j}^{(1)}}, \quad \lambda_j^{(2)} = \overline{\lambda_{N/2-j}^{(2)}}, \quad 0 \leq j \leq N/2, \\ &f_j^{(1)} = \overline{f_{N/2-j}^{(2)}}, \quad d_j^{(1)} = d_{N/2-j}^{(2)} = \overline{d_{N/2-j}^{(2)}}, \quad 0 \leq j \leq N/2. \end{aligned} \tag{4.31}$$

Notice that the indices are now up to N (or $N/2$); this must be understood in the sense of the periodic extension (e.g., $\lambda_0^{(1)} = \lambda_N^{(1)}$, $f_{N/2}^{(1)} = f_0^{(2)}$). Furthermore,

$$c_{11}^{(j)} = \overline{c_{22}^{(N/2-j)}}, \quad c_{12}^{(j)} = \overline{c_{21}^{(N/2-j)}}, \quad 0 \le j \le N/2. \tag{4.31a}$$

Proof. The proof is by direct verification. □

Write the eigenvalue relation for the C_j's as

$$(z^{(j)})^2 - 2\beta^{(j)} z^{(j)} - \gamma^{(j)} = 0, \quad \beta^{(j)} := \frac{c_{11}^{(j)} + c_{22}^{(j)}}{2}, \quad \gamma^{(j)} := c_{12}^{(j)} c_{21}^{(j)} - c_{11}^{(j)} c_{22}^{(j)}. \tag{4.32}$$

Use of (4.31a) allows us to conclude that $\beta^{(j)} = \overline{\beta^{(N/2-j)}}$, $\gamma^{(j)} = \overline{\gamma^{(N/2-j)}}$, and hence that $\overline{z^{(j)}} = z^{(N/2-j)}$, $j = 0, \ldots, N/2$. Thus, to bound $|z^{(j)}|$, we can restrict ourselves to considering $0 < j \le N/4$. Based on several attempts, we are convinced that the bound (4.30) for $\rho(C_0)$ is essentially also a bound for $\rho(\mathbf{M}_2^{1,0})$. More precisely, we make the following claim.

CLAIM 4.5. *Let $c(\boldsymbol{\theta}) = c \ge 0$ in (4.18), and let $q = hc$. With the previous notation, we have*

$$\rho(\mathbf{M}_2^{1,0}) \le \min\left(\frac{1}{2}, \frac{1}{(1+2q)^{N/2}} + \mathcal{O}(h)\right). \tag{4.33}$$ □

We know that this claim is asymptotically correct for $q \to \infty$, but the case of finite c is much more involved, and we do not have a complete proof that the bound (4.33) holds also for $\rho(\mathbf{M}_2^{1,0})$. We have ample numerical evidence that this claim is true, as well as rigorous proofs for some specific ranges of values of c and all choices of restrictions and prolongations, but at some stage we do need to resort to some computer-aided verification. This is the reason for leaving it as a claim. Still, we have several results in the direction of (4.33). For instance, a typical result we can give is the following.

FACT 4.6. *Consider the injection as a restriction and the prolongation (4.9), so that $d_j^{(1)} = d_j^{(2)} = 1$ and $f_j^{(1),(2)} = \frac{1}{2}(1 \pm \cos\frac{2\pi j}{N})$ in (4.28). Let $c(\boldsymbol{\theta}) = c \ge 0$ in (4.18), and let $q = hc$. Then we have*

$$\rho(\mathbf{M}_2^{1,0}) \le \frac{1}{2} + \mathcal{O}(h). \tag{4.34}$$

Below we present a verification of Fact 4.6 according to one possible line of action. To do this, we need the following lemma.

LEMMA 4.7. *For all $z \in \mathbb{C}$, we have*

$$|-1 + \sqrt{1+z}| \le |z|^{1/2}. \tag{4.35}$$

Proof. Let $z = re^{i\phi}$, so that (4.35) is equivalent to $1 + \rho - 2\rho^{1/2}\cos\frac{\theta}{2} \le r$, where $\rho = (1 + r^2 + 2r\cos\phi)^{1/2}$ and $\tan\theta = \frac{r\sin\phi}{1+r\cos\phi}$. Let $f(\phi) := 1 + \rho - 2\rho^{1/2}\cos\frac{\theta}{2}$, so that f' vanishes only at 0 and π, and $f(0) = (1 - \sqrt{1+r})^2$, $f(\pi) = (1 - \sqrt{|1-r|})^2$. So, we need to show that

$$\forall u \in \mathbb{R}, \quad \text{(a)} \quad |-1 + \sqrt{1+u}| \le \sqrt{|u|} \quad \text{and} \quad \text{(b)} \quad |-1 + \sqrt{|1-u|}| \le \sqrt{|u|}.$$

These two inequalities are proven similarly. We only show the proof of (a). First, let $-1 \le u$. If also $0 \le u$, then (a) is clear. If $-1 \le u < 0$, then let $u = \alpha - 1$, $\alpha \in [0, 1)$,

and (a) reduces to $\alpha < \sqrt{\alpha}$, which is true. If $u < -1$, let $u = -1 - \alpha$, $\alpha \geq 0$. Then $|-1+\sqrt{1+u}| = \sqrt{1+\alpha}$. □

Verification of Fact 4.6. For the moduli of the roots we have

$$|z^{(j)}| \leq \frac{|R_j|}{2}\,|c_{11}^{(j)} - c_{22}^{(j)}| + \max(|c_{11}^{(j)}|,\ |c_{22}^{(j)}|), \qquad 1 \leq j \leq N/4,$$

where

$$R_j := -1 + \left(1 + 4\frac{c_{12}^{(j)} c_{21}^{(j)}}{(c_{11}^{(j)} - c_{22}^{(j)})^2}\right)^{1/2}.$$

Now use (4.35) and (4.31a) to get (for simplicity, omit the index j)

$$|z| \leq |c_{12}c_{21}|^{1/2} + |c_{11}|\,. \tag{4.36}$$

To proceed, we need bounds on $|c_{12}c_{21}|$ and $|c_{11}|$. We have explicitly formed the moduli and maximized first in q, then in the angle (i.e., in j). Here we see in some detail what is involved for $|c_{12}c_{21}|$.

Let N be given, and for given j, let $\alpha = \alpha_j = 2\pi j/N$. Let $f(\alpha, q) := |c_{12}c_{21}|$ and write $f(\alpha, q) = g_1(\alpha, q)g_2(\alpha, q)F(\alpha)$, where (see (4.25), (4.28)) $F(\alpha) := 1/4 \sin^2\alpha$,

$$g_1 := \left(1 + A_1^{N/2} - 2A_1^{N/4}\cos\frac{N}{2}\phi_1\right)^{-1},$$

$$g_2 := \left(\frac{1 + A_3^N - 2A_3^{N/2}\cos(N\phi_3)}{A_3^N}\ \frac{1 + A_2^N - 2A_2^{N/2}\cos(N\phi_2)}{A_2^N}\right)^{1/2},$$

and $A_1(\alpha, q) = 1 + (2+2q)^2 - 2(2+2q)\cos 2\alpha$, $A_2(\alpha, q) = 1 + (2+q)^2 - 2(2+q)\cos\alpha$, $A_3(\alpha, q) = 1 + (2+q)^2 + 2(2+q)\cos\alpha$, and

$$\tan\phi_1 = \frac{(2+2q)\sin 2\alpha}{(2+2q)\cos 2\alpha - 1}, \quad \tan\phi_2 = \frac{(2+q)\sin\alpha}{(2+q)\cos\alpha - 1}, \quad \tan\phi_3 = \frac{(2+q)\sin\alpha}{(2+q)\cos\alpha + 1}.$$

Now, differentiation of g_1 and g_2 with respect to q shows a monotone decrease as $c \geq 0$, so that $f(\alpha, q) \leq f(\alpha, 0)$. Differentiation with respect to the angle shows that the maximum of $f(\alpha, 0)$ is obtained for $j = 1$, that is, for $\alpha = 2\pi/N$. Finally, $\lim_{N\to\infty} f(\alpha, 0) = 0$.

Similarly, for $|c_{11}|$ a double maximization argument shows that $|c_{11}(\alpha, q)| \leq |c_{11}(\alpha, 0)| \leq |c_{11}(2\pi/N, 0)|$, and as $N \to \infty$, this quantity approaches $1/2$. □

Final Remarks.

(1) The bound (4.34), unlike (4.33), hides the fact that $\rho(\mathbf{M}_2^{1,0})$ decays very rapidly for large values of c, as it should, just based on the smoothing process alone. In fact, for (4.18) with $c(\theta) = c > 0$, the block multigrid iteration is not a big improvement over the performance of the block smoothing by itself, unless c is rather small, in which case the block multigrid gives a sizable improvement (see Table 3(a) below). It also gives a substantial improvement when the coefficients in front of the derivatives are constant but of widely different magnitude. It is interesting (and perhaps surprising) that the different prolongation and restriction options all perform similarly on the model problem we analyzed.

(2) The above result is only for the constant-coefficients case. A standard perturbation argument allows us to extrapolate the above reasoning and to obtain similar bounds for the case

of $c(\boldsymbol{\theta})$ slowly varying, and sufficiently smooth and positive. What we cannot do, however, is to draw any conclusion in the case of nonconstant $f_1(\boldsymbol{\theta})$, $f_2(\boldsymbol{\theta})$ in (4.14). It is reasonable to expect that, for slowly varying f_1, f_2 which do not vanish, the above results are qualitatively correct. However, the case in which these coefficients change sign is much harder to analyze, and we do not have a clear picture as yet of the convergence behavior. The numerical tests at the end of the section provide some insight.

(3) It is very interesting that the bound of 1/2 is also observed for the two-level point multigrid iteration on the model problem (4.18) with $c(\boldsymbol{\theta}) = c > 0$, no matter how small c is (this bound deteriorates slightly for a k-level iteration). In this case, it is imperative to use the piecewise constant prolongation. Notice that the spectral radius of the simple smoothing iteration matrix is basically 1 for c close to 0.

Numerical Examples. All computations below have been performed in double precision on a SPARC-1 workstation. More examples than those below can be found in [DB0]. The notation in the tables is as follows: "smoother" indicates which smoother has been used for either point or block multigrid (GS, C, and ILU are the Gauss–Seidel, cyclic Gauss–Seidel, and incomplete LU smoothers), "levels" is the number of levels, "prol" is the type of prolongation used, "rate" indicates the observed average rate of decrease of the residual in the sup norm, ||res|| is the sup norm of the residual vector, "pre-post" indicates the number of pre- and postsmoothings, and NITS is the number of iterations needed to achieve a required value for the residual. The values of prol = 0, 1, 2 indicate the prolongation type: 0 is the linear prolongation, 1 is the piecewise constant, and 2 is the mixed one (used only for the block multigrid). In all cases, the initial guesses have been taken as identically 1, a maximum of 25 iterations was performed, and the iteration stopped when $\|\text{res}\| < 10^{-5}$. All problems have been discretized by (3.1), with $N_1 = N_2 = 80$. We have also tested the trivial injection as a restriction operator, but only report on the results obtained with the linear restriction, since they were consistently better. Likewise, the damped Jacobi iteration as a smoother always performed worse than the other smoothers, so we do not report on it.

(i) *Point Multigrid.* Our experience on the one-dimensional example $f(\theta)u_\theta + c(\theta)u = g(\theta)$, for several $f(\theta), c(\theta)$, entirely confirmed the theoretical results. In particular, when $f(\theta)$ did not vanish, we found that the number of levels had a very minor impact on the rate of convergence, the choice of linear prolongation could severely deteriorate the convergence, 1 pre- and 0 postsmoothing steps was the best choice, and the magnitude of η in (2.2) had a minor impact on convergence as well. The case in which $f(\theta)$ has a zero is exemplified by this case: $f(\theta) = -\sin(\theta + a)$, $c(\theta) = \eta$, and $g(\theta) = \sin^\eta(\theta + a)$, $a = .04$. This has exact solution

$$u(\theta) = -2^\eta \tan^\eta\left(\frac{\theta + a}{2}\right) \int_{\pi/2}^{\frac{\theta+a}{2}} \frac{\cos^{2\eta-1} t}{\sin t} dt,$$

and it is not in C^1 when $\eta = 1, 1/3$. In this case, we noticed that the cyclic Gauss–Seidel and ILU smoothers are always good choices, and that, especially if smooth upwind is performed, the linear interpolant prolongation is also performed well.

In Tables 1(a)–(b) we report on results for the following problem:

(Example 1) $$f_1(\theta_1, \theta_2)u_{\theta_1} + f_2(\theta_1, \theta_2)u_{\theta_2} + C(\theta_1, \theta_2)u = g(\theta_1, \theta_2),$$

and the usual periodic boundary conditions. Consider the following functions:

(a) f_1, $f_2 = \pm 1$, $C(\theta_1, \theta_2) = \eta$, $g(\theta_1, \theta_2) = \cos(\theta_1 + \theta_2) + \eta \sin\theta_1 \cos\theta_2$, and

(b) $f_1 = -\sin(\theta_1 + a)$, $f_2 = \begin{cases} -\cos(\theta_2 + a) \\ 1 \end{cases}$, $C = \eta$, $g = \sin^\eta(\theta_1 + a)$, $a = .04$

with solution independent of θ_2, the same as the one-dimensional problem above.

It is apparent that our theoretical insight is nicely confirmed, in particular as far as the prolongation operator is concerned. Observe the often dramatic improvement with respect to the results of [DB1].

TABLE 1(a)
Example 1(a), $\epsilon_1 = \epsilon_2 = 0$, *pre-post* = 1-0.

f_1	f_2	η	Smoother	Levels	Prol	NITS	$\|res\|$	Rate
1	1	1	GS	3, 5	1	4	10^{-6}	10^{-2}
"	"	"	"	"	0	20	$8 * 10^{-6}$	.4
"	"	"	C, ILU	3, 5	1	5	10^{-8}	10^{-2}
"	"	"	"	"	0	14	$8 * 10^{-6}$	.32
"	"	.1	GS, C, ILU	5	1	11	$7 * 10^{-6}$	.3
"	"	"	"	"	0	18	$8 * 10^{-6}$	.45
"	"	10^{-6}	GS	5	0	25	$8 * 10^{-3}$	.76
"	"	"	"	"	1	17	$6 * 10^{-6}$	.46
"	"	"	C, ILU	"	0	19	$8 * 10^{-6}$	.48
"	"	"	"	"	1	18	$6 * 10^{-6}$	.47
± 1	∓ 1	1	C, ILU	5	1	12	$8 * 10^{-6}$	.25
"	"	"	"	"	0	15	$6 * 10^{-6}$	.3

TABLE 1(b)
Example 1(b), $\eta = 1$, *levels* = 5.

f_2	(ϵ_1, ϵ_2)	Smoother	Prol	Pre-Post	NITS	$\|res\|$	Rate
1	(0,0)	C, ILU	1,0	1-0	13	10^{-5}	.33
"	"	"	1,0	2-1	5	10^{-5}	.1
"	(.1,0)	C, ILU	1	1-0	11	10^{-5}	.26
$-\cos(\theta_2 + a)$	(0,0)	C, ILU	1	"	11	$6 * 10^{-5}$	.28
"	"	"	0	"	13	$2 * 10^{-5}$	.34
"	(.1, .1)	"	1	"	10	$4 * 10^{-5}$	.23
"	"	C	0	"	12	10^{-5}	.31
"	"	ILU	"	"	13	10^{-5}	.33

The second model problem is (2.4) with $p = q = 2$:

(Example 2)
$$f_1(\boldsymbol{\theta})\begin{pmatrix} u \\ v \end{pmatrix}_{\theta_1} + f_2(\boldsymbol{\theta})\begin{pmatrix} u \\ v \end{pmatrix}_{\theta_2} + C(\boldsymbol{\theta})\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} g_1(\boldsymbol{\theta}) \\ g_2(\boldsymbol{\theta}) \end{pmatrix}$$

and the usual periodicity requirements in u and v. We consider the following functions:

(a) $$f_1 = f_2 = -1, \quad C = \begin{pmatrix} \alpha & 1 \\ -1 & \beta \end{pmatrix}, \quad \begin{pmatrix} g_1 \\ g_2 \end{pmatrix} = \begin{pmatrix} \alpha(\sin\theta_1 + \cos\theta_2) \\ \beta(\cos\theta_1 - \sin\theta_2) \end{pmatrix}, \quad \alpha, \beta > 0,$$

with exact solution $u = \sin\theta_1 + \cos\theta_2$, $v = \cos\theta_1 - \sin\theta_2$;

(b) $$f_1 = f_2 = 1, \qquad C = \begin{pmatrix} \alpha & -1 \\ 1 & \beta \end{pmatrix},$$

α, $\beta > 0$, g as in (a), with same solution as (a);

(c) $$f_1 = -\sin(\theta_1 + a),$$
$$f_2 = \begin{cases} 1 \\ -1 \\ -\cos(\theta_2 + \alpha) \end{cases}, \qquad C = \begin{pmatrix} 1 & \frac{1}{2}\cos(\theta_1 + \theta_2) \\ \frac{1}{2}\sin(\theta_1 + \theta_2) & 2 \end{pmatrix},$$

$a = .04$, $g_1 = \sin(\theta_1 + a) + \frac{1}{2}\cos(\theta_1 + \theta_2)v^*$, $g_2 = \sin^2(\theta_1 + a) + \frac{1}{2}\sin(\theta_1 + \theta_2)u^*$ with exact solution independent of θ_2:

$$u^* = -2\tan\frac{\theta_1 + a}{2}\ln\left|\sin\frac{\theta_1 + a}{2}\right|, \quad v^* = -4\tan^2\frac{\theta_1 + a}{2}\ln\left|\sin\frac{\theta_1 + a}{2}\right| - 2\sin^2\frac{\theta_1 + a}{2}.$$

For Examples 2(a)–(b), (2.2) is verified with $\eta = \min(\alpha, \beta)$, while for Example 2(c), η is arbitrarily close to 0. The notation of Tables 2(a)–(c) is as before, but now ILU refers to the implementation called ILU2 in the ILU section of [DB1]. It is interesting to observe the improvement over the results in [DB1] for the case of weakly attractive tori. However (see Tables 2(a)–(b) of [DB1]), the multigrid iteration might be marginally slower than the simple iterations for reasonably attractive tori.

TABLE 2(a)
Example 2(a), $\epsilon_1 = \epsilon_2 = 0$.

α	β	Smoother	Levels	Prol	Pre-Post	NITS	‖res‖	Rate
1	1	GS	5	1	1-0	25	.16	.84
"	"	C, ILU	5, 3	"	"	12	$9 * 10^{-6}$	.3
"	"	"	"	"	2-1	3	$9 * 10^{-8}$	.005
"	"	C	5	0	1-0	17	$6 * 10^{-6}$	.45
.1	.1	GS	"	1	"	25	.37	.88
"	"	C	"	"	"	16	$7 * 10^{-6}$	.37
"	"	ILU	"	"	"	14	$9 * 10^{-6}$	.36

TABLE 2(b)
Example 2(b), $\epsilon_1 = \epsilon_2 = 0$, *levels* = 5.

α	β	Smoother	Prol	Pre-Post	NITS	‖res‖	Rate
1	1	GS, C, ILU	1	1-0	5	10^{-7}	.1
1	.01	GS, C, ILU	"	"	6	$3 * 10^{-6}$	.07
"	"	GS, C, ILU	1	2-1	3	10^{-7}	.01
"	"	C	0	1-0	15	10^{-6}	.35

TABLE 2(c)
Example 2(c), $f_2 = -\cos(\theta_2 + a)$, *levels* = 5.

(ϵ_1, ϵ_2)	Smoother	Prol	Pre-Post	NITS	‖res‖	Rate
(.1,.1)	GS	1	1-0	25	$6 * 10^{-3}$	.77
"	"	0	"	25	$8 * 10^{-3}$	.8
"	C	1	"	25	$6 * 10^{-3}$	.77
"	"	0	"	13	$6 * 10^{-6}$	.3
(0,0)	C, ILU	1	"	12	$6 * 10^{-6}$	.3
"	"	0	"	15	$3 * 10^{-6}$	.37
"	"	"	2-1	5	$9 * 10^{-6}$	.09

(ii) *Block Multigrid.* Below are some results for the block multigrid iterations. The results of Table 3(a) agree with (4.33). Tables 3(b) and 4 show that even for functions changing sign the techniques perform rather well. Usually, the classical upwind discretization leads to better convergence, and the piecewise constant prolongation is to be preferred.

Final Remarks. As we hoped, the multigrid iteration gives a more homogeneous convergence behavior than the simpler iterations by themselves. This is always the case even though the simpler iterations might be occasionally faster. But for weakly attractive tori, the multigrid schemes are superior. Our results show that it usually does not pay to perform a

TABLE 3(a)
Example 1(a), $\epsilon_1 = \epsilon_2 = 0$, *levels* = 5.

f_1	f_2	η	Smoother	Prol	Pre-Post	NITS	$\|res\|$	Rate
1	1	.1	GS, C	1, 2	1-0	9	10^{-6}	.2
"	"	"	"	"	2-1	4	10^{-6}	.017
"	"	"	"	0	1-0	12	$6 * 10^{-6}$	.25
"	"	10^{-6}	GS	0	"	10	$5 * 10^{-6}$	.28
"	"	"	GS, C	1, 2	"	16	$5.8 * 10^{-6}$	.46
"	"	"	C	0	"	12	$5.2 * 10^{-6}$	.34

TABLE 3(b)
Example 1(b), $f_2 = -\cos(\theta_2 + a)$, $a = .04$, $\eta = 1$, *levels* = 5, *pre-post* = 1-0.

(ϵ_1, ϵ_2)	Smoother	Prol	NITS	$\|res\|$	Rate
(0,0)	GS	0, 1, 2	25	—	.8
"	C	1	5	10^{-6}	.02
"	"	0	10	10^{-6}	.1
"	"	2	15	$7 * 10^{-6}$	.4
(.1, .1)	GS	1, 0	25	10^{-4}	.7
"	C	1	6	$6 * 10^{-6}$	.1
"	"	0	9	$6 * 10^{-6}$	.16

TABLE 4
Example 2(c), *levels* = 5, *pre-post* = 1-0.

f_2	(ϵ_1, ϵ_2)	Smoother	Prol	NITS	$\|res\|$	Rate
-1	(0,0)	C	1	3	$5 * 10^{-6}$	.006
"	"	"	0	10	$7 * 10^{-6}$	.23
$-\cos(\theta_2 + a)$	(0,0)	C	1	5	$4 * 10^{-6}$	.03
"	"	"	0	10	$7 * 10^{-6}$	.23
"	(.1, .1)	C	1	7	10^{-6}	.1
"	"	"	0	10	$4 * 10^{-6}$	.22

block multigrid iteration, as the simpler point multigrid schemes are often as effective and, of course, less costly. In the absence of any further insight on the problem, our recommendations are: classical upwind discretization, cyclic Gauss–Seidel as smoother, one pre- and no postsmoothing steps, piecewise constant prolongation as prolongation operator, and linear restriction as restriction operator.

5. Two nonlinear problems. Here, we consider two physically relevant nonlinear problems. We use (3.3) to discretize the problem, perform a discrete Newton's iteration, and solve the resulting linear systems by the multigrid techniques of §4.

Josephson Junction [L], [vV1]. Consider the computation of the invariant torus for the following nonlinear forced oscillator:

$$\beta\ddot{\phi} + (1 + \gamma\cos\phi)\dot{\phi} + \sin\phi = p(t), \tag{5.1}$$

where $\beta, \gamma > 0$, and $p(t)$ is any function periodic in t of period 2π. Very careful computation of the invariant torus of (5.1) has been recently carried out by van Veldhuizen [vV1]. We present results obtained with our approach, to show how efficiently this type of problem can be solved with our method.

First, we rewrite (5.1) as the first-order system

$$\begin{aligned} \dot{\phi} &= \frac{1}{\beta}(u - \phi - \gamma \sin\phi), \\ \dot{u} &= -\sin\phi + p(t), \end{aligned} \tag{5.2}$$

and then ask for u to be $u = u(\phi, t)$, which leads to the scalar PDE

$$\frac{1}{\beta}(u - \phi - \gamma \sin\phi)u_\phi + u_t = -\sin\phi + p(t). \tag{5.3}$$

With respect to (2.6) we thus have

$$f_1(\phi, t, u) = \frac{1}{\beta}(u - \phi - \gamma \sin\phi), \quad f_2(\phi, t, u) = 1, \quad g(\phi, t, u) = -\sin\phi + p(t). \tag{5.4}$$

Observe that the problem is not periodic in ϕ in the usual sense, but only in the $\bmod 2\pi$ sense. In fact, because of the term $\phi + \gamma \sin\phi$, we can only expect u to be periodic in ϕ ($\bmod 2\pi$). In other words, both u and ϕ are angular coordinates. However, in the implementation, there is no need to explicitly enforce on u the $\bmod 2\pi$ periodicity in ϕ. First, observe that from (5.3), as $\beta \to 0$, u becomes a function of ϕ only as

$$\lim_{\beta \to 0} u = \bar{u}_0 := \phi + \gamma \sin\phi, \tag{5.5}$$

from which it is clear that $\bar{u}_0(0) = \bar{u}_0(2\pi) - 2\pi$; by continuity arguments, $u(\phi, t) = u(\phi + 2\pi, t) - 2\pi$ (also for $\beta > 0$). This is a linear boundary condition which we can explicitly enforce. Thus, the boundary conditions associated with (5.3) are

$$u(0, t) = u(2\pi, t) - 2\pi, \qquad u(\phi, 0) = u(\phi, 2\pi). \tag{5.3a}$$

The feasibility of the parametrization $u = u(\phi, t)$ and the solvability of the PDE (5.3)–(5.3a) must still be justified. The analytical results of [L] can be used to guarantee feasibility of the approach, because of the following theorem.

THEOREM 5.1. [L] *For all $p(t)$, 2π-periodic in t, and β, γ such that*

$$0 \le \gamma < 1, \qquad 0 < \beta < \frac{1}{4}(1 - \gamma)^2, \tag{5.6}$$

the Poincaré map for (5.2) *has a globally attracting invariant circle parametrizable as $u = F(\phi)$, where $F(\phi)$ is increasing in ϕ.* □

Remarks. (1) A consequence of Theorem 5.1 is the existence of an invariant torus $u(\phi, t)$ satisfying (5.3).

(2) The fact that $F(\phi)$ is increasing, hence invertible, has been used in [vV1] to rewrite the invariant manifold for ϕ in terms of u. However, we keep the $u(\phi, t)$ parametrization. In [vV1], the author also shows that the region of validity for Theorem 5.1 is larger than the region characterized by (5.6).

Next, consider a formal linearization of (5.3) around the solution. Since $g_u = 0$, and $f_{1,u} = \frac{1}{\beta}$, $f_{1,t} = f_{2,u} = f_{2,t} = 0$, we obtain a linear system as in (2.4), with $c(\phi, t) = \frac{1}{\beta}u_\phi$. Because of Theorem 5.1, we have $u_\phi > 0$, and therefore we have $c(\phi, t) > 0$ when (5.6) holds. Thus, the upwind discretization of (2.1) would give us a standard M-matrix. This

is also true for the linear systems that arise when doing a (discrete) Newton iteration on the nonlinear upwind scheme. A simple justification of this fact comes by looking at the diagonal elements of the discretization matrix (recall (3.5)), which can be written as

$$c_{ij} = \frac{\sigma_1}{h_1} + \frac{\sigma_2}{h_2} + \frac{1}{2h\beta}[(u^0_{i+1,j} - u^0_{i,j})(1 - \sigma_1') + (u^0_{i,j} - u^0_{i-1,j})(1 + \sigma_1')],$$

and so the claim follows at once as long as u^0 is increasing (as a grid-function), regardless of changes of sign for f_1, and of the values of h_1, h_2 (recall that $\sigma_1' \in [-1, 1]$). We can easily guarantee that u^0 is increasing when we start the approximation, because we choose $\bar{u}_0$ in (5.5) as the first guess. Afterwards, we check whether the computed approximations verify this monotonicity property. In our calculations, with $p(t) = \cos(t)$, we have fixed $\gamma = 0.5$ and let β increase up to 1, and the monotonicity was preserved, except for $\beta = 1$. Preservation of this increasing behavior of the solution is very important if we want to approximate the rotation number (see [vV2]).

We do not know, a priori, whether the function f_1 changes sign, as ϕ and t vary. Hence, we have used a smooth upwind discretization with respect to f_1 ($\epsilon_1 > 0$); in fact, we have found that f_1 does change sign. In our implementation, we have set $\epsilon_1 = 10^{-4}$. We can, instead, safely use a classical upwind discretization with respect to f_2 ($\epsilon_2 = 0$). We have performed a simple continuation procedure in β, fixing $\gamma = 0.5$. The analytical validity of the procedure is expected up to $\beta \approx \frac{11}{20}$; see [vV1]. However, we proceed without any apparent difficulty up to $\beta = 1$, after 80 continuation steps and about 1 hour of CPU time. The automatic continuation procedure is standard: we start with $\beta = 10^{-4}$, with a small increment of $\delta\beta = 10^{-4}$, which is then halved or doubled as we proceed, depending on the convergence behavior of the Newton iterates and of the iterative solution algorithm for the linear systems. We use an equispaced mesh of (80, 80) points and stop the Newton iterates when the sup norm of the correction is less than 10^{-4}, while, for safety reasons, we require the sup norm of the residual vector to be less than 10^{-5} for convergence of the iterative method when solving the linear systems. The CPU time quoted above refers to a five-level point multigrid algorithm, with one presmoothing and no postsmoothing, piecewise constant prolongation, linear restriction, and either cyclic Gauss–Seidel or ILU as smoothers (performance was nearly identical). The reduction in the residuals' norm for each step of the multigrid iteration deteriorates as β approaches 1: for example, for $\beta = 10^{-4}$ it is about 10^{-3}, while when $\beta = 1$ it is about $\frac{1}{4}$. For this problem, as we progressively "weaken" the M-matrix structure as β nears 1, the simpler iterations are not as effective. In Figs. 1 and 2 we show the computed invariant tori, for the parameter values indicated (Fig. 2 is an enlargement of Fig. 1). The mod 2π periodicity, and the lack of monotonicity in the solution for $\beta = 1$ are apparent.

Coupled Oscillator [ADO], [DLR]. Consider the system of two linearly coupled nonlinear oscillators:

$$\begin{aligned}
\dot{x}_1 &= x_1 + \beta y_1 - (x_1^2 + y_1^2)x_1 - \delta(x_1 + y_1 - x_2 - y_2),\\
\dot{y}_1 &= -\beta x_1 + y_1 - (x_1^2 + y_1^2)y_1 - \delta(x_1 + y_1 - x_2 - y_2),\\
\dot{x}_2 &= x_2 + \beta y_2 - (x_2^2 + y_2^2)x_2 + \delta(x_1 + y_1 - x_2 - y_2),\\
\dot{y}_2 &= -\beta x_2 + y_2 - (x_2^2 + y_2^2)y_2 + \delta(x_1 + y_1 - x_2 - y_2);
\end{aligned} \tag{5.7}$$

we fix $\beta = 0.55$, and the coupling parameter δ is positive. For $\delta = 0$, the two oscillators both have periodic, attracting, limit cycles $x_i^2 + y_i^2 = 1$, and thus the system has an attracting invariant torus given by the Cartesian product of these two circles. For $\delta > 0$, we perform a numerical continuation procedure in an attempt to follow the evolution of this torus.

As in [DLR], we first rewrite (x_i, y_i) in the polar plane $(r_i, -\theta_i)$, and then let $r_i = 1 + \delta u_i$, $i = 1, 2$. So doing, we obtain a new system for $\boldsymbol{\theta} = (\theta_1, \theta_2)$, $\mathbf{u} = (u_1, u_2)$ given as

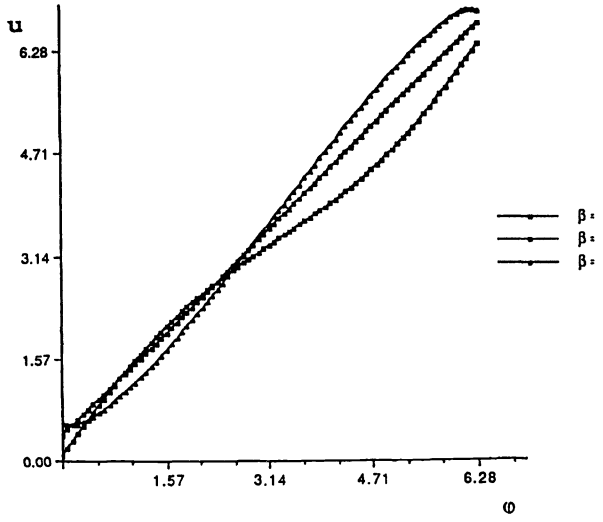


FIG. 1. *Josephson junction.*

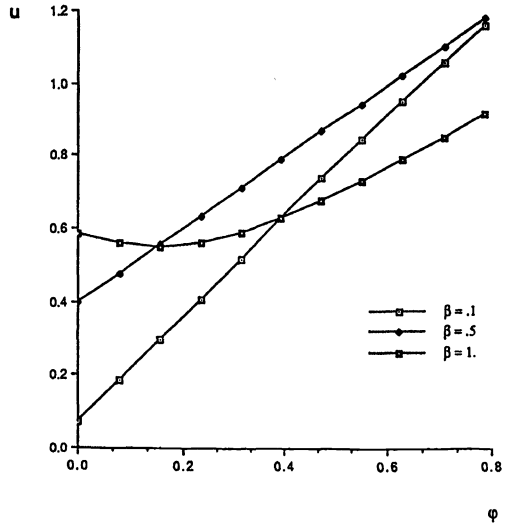


FIG. 2. *Josephson junction: Blow-up of Fig.* 1.

$$\begin{aligned} \dot{\boldsymbol{\theta}} &= \mathbf{f}(\boldsymbol{\theta}, \mathbf{u}), \\ \dot{\mathbf{u}} &= \mathbf{g}(\boldsymbol{\theta}, \mathbf{u}), \end{aligned} \tag{5.8}$$

where

$$\mathbf{f}(\boldsymbol{\theta}, \mathbf{u}) := \begin{pmatrix} \beta + \delta(\cos 2\theta_1 - \frac{1+\delta u_2}{1+\delta u_1}(\sin(\theta_1 - \theta_2) + \cos(\theta_1 + \theta_2))) \\ \beta + \delta(\cos 2\theta_2 - \frac{1+\delta u_1}{1+\delta u_2}(\sin(\theta_2 - \theta_1) + \cos(\theta_1 + \theta_2))) \end{pmatrix},$$

$$\mathbf{g}(\boldsymbol{\theta}, \mathbf{u}) := \begin{pmatrix} -2u_1 - \delta u_1^2(\delta u_1 + 3) - (1 + \delta u_1)(1 - \sin 2\theta_1) - a \\ -2u_2 - \delta u_2^2(\delta u_2 + 3) - (1 + \delta u_2)(1 - \sin 2\theta_2) - b \end{pmatrix}, \tag{5.9}$$

where $a = (1+\delta u_2)(\sin(\theta_1+\theta_2)-\cos(\theta_1-\theta_2))$ and $b = (1+\delta u_1)(\sin(\theta_1+\theta_2)-\cos(\theta_1-\theta_2))$. The usual invariance condition $\mathbf{u} = \mathbf{u}(\boldsymbol{\theta})$ gives the PDE system

$$
\begin{aligned}
& f_1(\boldsymbol{\theta}, \mathbf{u})\mathbf{u}_{\theta_1} + f_2(\boldsymbol{\theta}, \mathbf{u})\mathbf{u}_{\theta_2} = \mathbf{g}(\boldsymbol{\theta}, \mathbf{u}), \\
& \mathbf{u}(0, \theta_2) = \mathbf{u}(2\pi, \theta_2), \qquad \mathbf{u}(\theta_1, 0) = \mathbf{u}(\theta_1, 2\pi).
\end{aligned} \tag{5.10}
$$

The solution of this PDE can be found, for δ small, by carrying out an expansion in powers of δ (see [DB0]), but this is not needed here.

From the discretization viewpoint, since the gradients ∇f_1 and ∇f_2 are $O(\delta^2)$ quantities, and the Jacobian of $\mathbf{g}$ is the symmetric matrix

$$
\frac{\partial \mathbf{g}}{\partial \mathbf{u}} = \begin{pmatrix} -2 - \delta(1 - \sin 2\theta_1 + 6u_1 + 3\delta u_1^2) & -\delta(\sin(\theta_1+\theta_2) - \cos(\theta_1-\theta_2)) \\ -\delta(\sin(\theta_1+\theta_2) - \cos(\theta_1-\theta_2)) & -2 - \delta(1 - \sin 2\theta_2 + 6u_2 + 3\delta u_2^2) \end{pmatrix},
$$

then the matrix in (2.8) is certainly positive definite for δ small enough, and the discrete system inherits a block M-matrix structure. In fact, since the function $\mathbf{f}$ is of the nice form contemplated in the discussion following (3.7), for the discretized problem it is possible to show that a Newton–Kantorovich result holds for $\delta > 0$ small enough. In practice, we have checked whether the (computed) discrete problem maintained a block M-matrix structure.

Clearly, for δ small, the functions f_1 and f_2 are positive (recall that we have fixed $\beta = 0.55$). As it turns out, they remain positive as continuation in δ is performed (for the values of δ considered), and we can thus safely use a classical upwind discretization with respect to both f_1 and f_2. We have performed several computations for this problem, with different choices for solving the linear systems and different values of the smoothing parameters. In all cases, we have discretized on a (160, 160) grid, and performed a simple continuation in δ in the same way as described for the Josephson junction problem. We have then repeated the computations on a (320, 320) grid, with similar outcome (with an increase in CPU time by a factor of 8). Newton's method was observed to converge quadratically all the way.

The following is a description of two typical computations. In the first, we have done a smooth upwind discretization for both f_1 and f_2 with smoothing value 10^{-2}, and carried out a cyclic Gauss–Seidel iteration all along to solve the linear systems. In the second computation, we have used a classical upwind discretization ($\epsilon_{1,2} = 0$), and performed a six-level point multigrid strategy, with one presmoothing and no postsmoothing steps, using (forward) Gauss–Seidel as a smoother, linear restriction, and piecewise constant prolongation. For the first run, we successfully continued up to $\delta = 0.2609375$, with 60 continuation steps, while in the second case we were successful up to $\delta = 0.2627734375$, after 15 continuation steps; in both cases, the continuation step decreased below 10^{-5} after these indicated values, and the procedure halted. Both runs took about four hours of CPU time. In both cases, the rate of decrease for the iterative solution of the linear systems deteriorates as δ increases. For example, for the simple cyclic Gauss–Seidel iteration it is $\frac{1}{10}$ when $\delta = 0$, $\frac{1}{2}$ when $\delta = 0.25$, and $\frac{3}{5}$ at the end; for the multigrid strategy, the reduction is 10^{-3} when $\delta = 0$, $\frac{1}{10}$ for $\delta = 0.25$, and about $\frac{1}{4}$ at the end. (These are average rates of decrease.) The decrease in speed of convergence is due to the apparent weakening of attractivity of the invariant tori. All runs have then been repeated on a (320, 320) grid, obtaining similar outcome. In Figs. 3 and 4 we show the cross-sections of the invariant tori r_1 and r_2 obtained by holding $\theta_1 = 0$; these figures are a plot of the data obtained with the second computation on the (320, 320) grid, for selected values of δ up to 0.262104; the dot in the middle of the pictures is the origin. (Notice that for this problem one actually has $r_1(\theta_1, \theta_2) = r_2(\theta_2, \theta_1)$.) Plots of the data obtained with the first computation are basically identical for the common values of δ. A noticeable cusp seems to develop. These figures are qualitatively different from those we obtained in [DLR]; apart from a silly plotting

mishap in [DLR] (Figs. 17–20 there have been plotted for $-\theta$ rather than θ), the pictures in [DLR] show a bulge that is not present in the figures here. It is likely that the second-order scheme used in [DLR] develops some numerical instability. (The first-order discretization adopted here has very strong shape-preserving properties.) The value of $\delta \approx 0.263$ that we reached here is very close to the value for which a bifurcation takes place (see [ADO], Fig. 40).

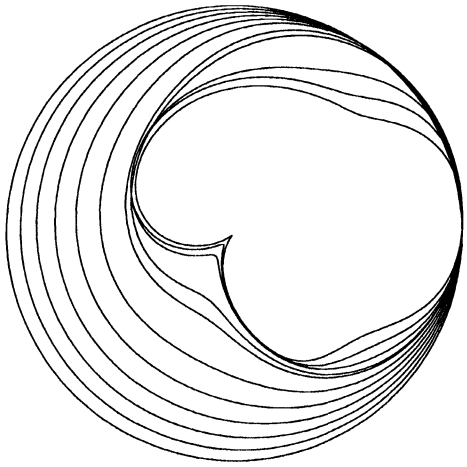


FIG. 3. *Coupled oscillator*: $r_1(0, \theta_2)$, $\delta \leq 0.262104$. *Classic upwind, multigrid iteration*, $N_1 = N_2 = 320$.

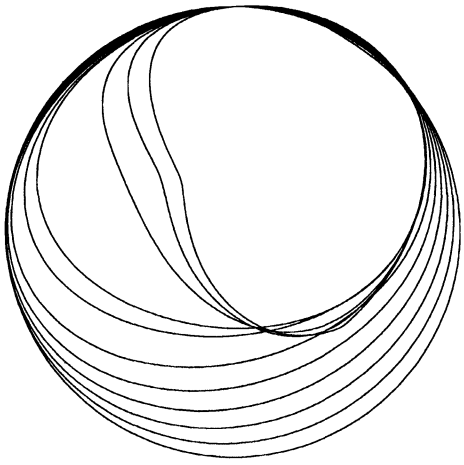


FIG. 4. *Coupled oscillator*: $r_2(0, \theta_2)$, $\delta \leq 0.262104$. *Classic upwind, multigrid iteration*, $N_1 = N_2 = 320$.

6. Conclusions. In this paper, we have continued (see [DB1]) the study of methods of solutions' strategies for the linear systems resulting after discretization of the periodic hyperbolic PDEs associated with invariant tori. Whereas in [DB1] we had considered some of the standard iterative methods, here we have considered multigrid approaches for the linear systems. We considered some new and some old choices for the restriction and prolongation operators, and also considered "point" and block smoothings. We proved convergence results on some model problems. Extensive implementation and testing has been done. We have

also given results for two important nonlinear problems: the forced Josephson junction and a system of coupled oscillators. The resulting linear systems have been solved by the multigrid methods, at a saving of several orders of magnitude over previously published results.

There are many aspects that still need to be addressed. It will be important to obtain more general convergence results for multigrid schemes on hyperbolic PDEs of our type as well as for similar hyperbolic PDEs. From the approximation viewpoint, work is needed to carry out a complete approximation theory for systems of nonlinear PDEs of the type we considered.

REFERENCES

[ADO] D. G. ARONSON, E. J. DOEDEL, AND H. G. OTHMER, *An analytical and numerical study of the bifurcations in a system of linearly coupled oscillators*, Phys. D, 25 (1987), pp. 20–104.

[BY] A. BRANDT AND J. YAVNEH, *Accelerated multigrid convergence and high Reynolds recirculating flows*, SIAM J. Sci. Comput., 14 (1993), pp. 607–626.

[DB0] L. DIECI AND G. BADER, *On approximating invariant tori: Block iterations and multigrid methods for the associated systems*, Preprint 658, Jan. 1992, SFB 123, Universität Heidelberg, Heidelberg, Germany, 1992.

[DB1] ———, *Solution of the systems associated with invariant tori approximation. I: Block iterations and compactification*, manuscript.

[DLR] L. DIECI, J. LORENZ, AND R. D. RUSSELL, *Numerical calculation of invariant tori*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 607–647.

[G] W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[Hac] W. HACKBUSH, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[Hal] J. K. HALE, *Ordinary Differential Equations*, 2nd Ed., R. Krieger, Melbourne, Fl, 1980.

[HS] P. HEMKER AND S. P. SPEKREIJSE, *Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations*, Appl. Numer. Math., 2 (1986), pp. 475–493.

[K] E. KAMKE, *Differentialgleichungen lösungsmethoden und lösungen*, Akademische Verlag, Leipzig, 1950.

[L] M. LEVI, *Nonchaotic behavior in the Josephson junction*, Phys. Rev. A, 37 (1988), pp. 927–931.

[OR] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solutions of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[S] R. SACKER, *A new approach to the perturbation theory of invariant surfaces*, Comm. Pure Appl. Math., 18 (1965), pp. 717–732.

[vV1] M. VAN VELDHUIZEN, *Approximation of the invariant manifold in the Josephson equation*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 666–675.

[vV2] ———, *On the numerical approximation of the rotation number*, J. Comp. Appl. Math., (1988), pp. 203–212.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1401–1412, November 1994

008

PERFORMANCE OF SEVERAL OPTIMIZATION METHODS ON ROBOT TRAJECTORY PLANNING PROBLEMS*

JOSEPH G. ECKER†, MICHAEL KUPFERSCHMID†‡, AND SAMUEL P. MARIN§

Abstract. This paper provides a comparison and analysis of the performance of a special purpose algorithm and several specific available nonlinear programming codes applied to a robot trajectory problem.

Key words. nonlinear programming, algorithm evaluation, robot modeling

AMS subject classification. 90

1. Introduction. Industrial robots track computed trajectories to coordinate the robot's axes to perform a desired task. We consider the planning of such trajectories "off-line" in a computer environment separate from the robot controller and actual execution of the motion, suitable for applications in which the trajectory is to remain fixed over a large number of cycles.

Off-line robot trajectory planning for manufacturing applications can frequently be accomplished by geometric path planning to satisfy spatial requirements of the task, followed by time parameterization of the path to meet robot dynamic limitations and to optimize a given performance objective. In this paper we focus on computational aspects of the time parameterization of a fixed path and use nonlinear programming algorithms to minimize the travel time of the robot on the given path while enforcing upper bound constraints on the absolute values of the axis velocity, acceleration, and jerk. Each axis starts and ends with zero velocity and acceleration.

A related problem involving constraints on manipulator joint torques is discussed in [11], where a special-purpose algorithm, CCD, based on cyclic coordinate descent, is developed. The approach used to formulate the nonlinear programming model in [11] can be modified to deal with the velocity, acceleration, and jerk constraints in this paper. In §2 we give a brief formulation of the current problem and refer the reader to [3], [10], and [11] for more details on the approach.

The main purpose of this paper is a comparison and analysis of the performance of a special-purpose algorithm CCD and several specific available nonlinear programming codes applied to an initial coarse-grid version of the problem.

2. Problem formulation. As discussed in detail in [11], the variables in the model are the coefficients x_j used in a cubic B-spline approximation of the derivative of an unknown strictly increasing function h, where $t = h(\tau)$ gives the time required to reach a position on the fixed parametric path corresponding to $\tau \epsilon [0, 1]$. In particular, if $g(\tau) = h'(\tau)$ is the derivative of h with respect to τ, then g is approximated by

$$g_N = \sum_{j=1}^{N+2} x_j \phi_j, \tag{1}$$

*Received by the editors December 5, 1988; accepted for publication (in revised form) September 21, 1993.

†Rensselaer Polytechnic Institute, Department of Mathematical Sciences, Troy, New York 12181-3590 (eckerj@rpi.edu).

‡Rensselaer Polytechnic Institute, Alan M. Voorhees Computing Center, Troy, New York 12180-3590 (mikek@ganef.its.rpi.edu).

§GM Research Staff, Mathematics Department, Warren, Michigan 48090-9055.

where the functions $\phi_1, \ldots, \phi_{N+2}$ are the B-splines spanning the space of piecewise cubic polynomials with N breakpoints in [0, 1]; see [2]. From

$$t = h(\tau) = \int_0^\tau g(z)dz$$

it follows that total travel time is

$$h(1) = \int_0^1 g(\tau)d\tau,$$

which we approximate as

$$\int_0^1 g_N(\tau)d\tau = \sum_{j=1}^{N+2} x_j \left[\int_0^1 \phi_j(\tau)d\tau\right] = b^T x,$$

where

$$b_j = \int_0^1 \phi_j(\tau)d\tau \quad \text{for } j = 1, \ldots, N+2 \quad \text{and} \quad x = (x_1, \ldots, x_{N+2})^T.$$

We consider a problem with three robot axes so the fixed path is defined by $[\theta_1(\tau), \theta_2(\tau), \theta_3(\tau)]$, where $\theta_s(\tau)$ gives the position of the sth robot axis in terms of τ. The constraints on the velocity, acceleration, and jerk in each axis are bounds on the absolute values of the first three derivatives (with respect to time t) of each $\theta_s(\tau)$, namely,

$$\left|\frac{d^r}{dt^r}\theta_s(\tau)\right| \le C_{sr}, \quad s = 1, 2, 3, \quad \text{and } r = 1, 2, 3. \tag{2}$$

By using $\tau = h^{-1}(t)$, applying the chain rule, and substituting $g(\tau) = h'(\tau)$, these constraints can be expressed in terms of g and derivatives (with respect to τ) of g and the θ_s's. For example, in the velocity constraint

$$\frac{d}{dt}\theta_s(\tau) = \frac{d}{dt}\theta_s(h^{-1}(t)) = \frac{\theta_s'(\tau)}{h'(\tau)} = \frac{\theta_s'(\tau)}{g(\tau)}.$$

Throughout, prime denotes differentiation with respect to τ. Replacing the absolute value constraint (2) by two inequalities, the velocity constraint for the sth axis becomes

$$-g(\tau)C_{s1} \le \theta_s'(\tau) \le g(\tau)C_{s1}.$$

Substituting the approximation g_N of (1) for g gives

$$-C_{s1}\sum_{j=1}^{N+2} x_j\phi_j(\tau) \le \theta_s'(\tau) \le C_{s1}\sum_{j=1}^{N+2} x_j\phi_j(\tau).$$

Since $\tau \in [0, 1]$ we can discretize this constraint by selecting M distinct points τ_p in [0, 1] and requiring that the preceding pair of inequalities holds at each τ_p. Thus, for the sth axis the velocity constraint $|d\theta_s/dt(\tau)| \le C_{s1}$ can be approximated by $2M$ linear constraints in the variables x so a total of $6M$ linear constraints are needed for all three axes.

Similarly, the acceleration and jerk constraints can be expressed using an additional $12M$ constraints. Because the second and third derivatives of $\theta_s(\tau)$ in (2) are nonlinear functions of the variables x_j (see [11]), these constraints are nonlinear as shown below in (3). In addition to

the bounds C_{sr}, all of the remaining data necessary to precisely define the constraints are the values of the functions θ_s and ϕ_j and their derivatives evaluated at the points τ_p. Specifically, let the column

$$\phi^{(r)}(\tau) = \left(\phi_1^{(r)}(\tau), \phi_2^{(r)}(\tau), \ldots, \phi_{N+2}^{(r)}(\tau)\right)^T, \qquad r = 0, 1, 2,$$

where $\phi_i^{(0)} = \phi_i$ and $\phi_i^{(1)}$ and $\phi_i^{(2)}$ denote the first and second derivatives of ϕ_i with respect to τ. Similarly, for $r = 1, 2, 3$ let $\theta_s^{(r)}(\tau)$ denote the rth derivative of θ_s with respect to τ.

Using this notation the resulting nonlinear programming problem is

(3)
$$\begin{aligned}
&\min b^T x,\\
&\text{subject to}\\
&-C_{s1}\phi(\tau_p)^T x \pm \theta_s^{(1)}(\tau_p) \le 0,\\
&-C_{s2}\left[\phi(\tau_p)^T x\right]^3 \pm \left[\theta_s^{(2)}(\tau_p)\phi(\tau_p) - \theta_s^{(1)}(\tau_p)\phi^{(1)}(\tau_p)\right]^T x \le 0,\\
&-C_{s3}\left[\phi(\tau_p)^T x\right]^5 \pm \Big[\theta_s^{(3)}(\tau_p)\left(\phi(\tau_p)^T x\right)^2 - 3\theta_s^{(2)}(\tau_p)\left(\phi(\tau_p)^T x\right)\left(\phi^{(1)}(\tau_p)^T x\right)\\
&\qquad\qquad +3\theta_s^{(1)}(\tau_p)\left(\phi^{(1)}(\tau_p)^T x\right)^2 + \theta_s^{(1)}(\tau_p)\left(\phi(\tau_p)^T x\right)\left(\phi^{(2)}(\tau_p)^T x\right)\Big] \le 0,\\
&s = 1, 2, 3, \quad \text{and} \quad p = 1, \ldots, M,
\end{aligned}$$

where the use of "±" indicates two constraints, one including the "+" and another including the "−".

Three coarse-grid realizations of this problem called Robot A, Robot B, and Robot C are studied in the next section. Each of these coarse-grid problems uses $N = 9$ equally spaced breakpoints and $M = 41$ equally spaced discrete values τ_p and thus has $n = N + 2 = 11$ variables x_j and $m = 18M = 738$ constraints. Each problem has the general form

$$\min f_0(x) \quad \text{subject to } f_i(x) \le 0, \quad i = 1, \ldots, m, \tag{4}$$

and uses the following functions θ_s for $s = 1, 2$, and 3 corresponding to the shoulder, elbow, and wrist axes, respectively.

$$\begin{aligned}
\theta_1(\tau) &= 1.5 f(\tau),\\
\theta_2(\tau) &= -0.5 \sin(4.7 f(\tau)),\\
\theta_3(\tau) &= -1.3 f(\tau) + 2.6,
\end{aligned}$$

where $f(\tau) = \tau^3(6\tau^2 - 15\tau + 10)$, $\tau \in [0, 1]$. The derivative bounds C_{sr}, $s = 1, 2, 3$ and $r = 1, 2, 3$ used for Robots A, B, and C are given in Table 1.

The bounds for Robot A were chosen to be most restrictive at the shoulder axes so that, at optimality, constraints involving θ_1 will most frequently be active. Robot B has identical bounds for each axis so that at optimality all axes will be involved in the active constraints. Robot C differs from Robot A only in the choice of third derivative bounds. In Robot C these are intentionally made small to insure that some of the highly nonlinear third derivative constraints are active at optimality.

A feasible starting point $x^0 = (a, a, \ldots, a)^T$ is easily computed [11] for each of the problems. For Robot A, $a = 1.491400623321533$; for Robot B, $a = 4.377146244049071$; and for Robot C, $a = 1.920426464080810$.

3. Computational experiments for coarse grid problems. In this section, we compare the performance of the CCD algorithm with four general-purpose algorithms when used to solve the three problems Robot A, Robot B, and Robot C.

TABLE 1
Derivative bounds C_{sr}.

	Robot A	Robot B	Robot C
C_{11}	2	1	2
C_{12}	8	3	8
C_{13}	250	100	25
C_{21}	3	1	3
C_{22}	18	3	18
C_{23}	650	100	65
C_{31}	4	1	4
C_{32}	50	3	50
C_{33}	1000	100	100

The CCD algorithm is the standard cyclic coordinate descent method [1] modified to deal with constraints. The algorithm starts at a feasible point x^0. At each iteration k the algorithm generates the next iterate $x^k = x^{k-1} + \alpha u_j$, where u_j is a coordinate direction and α is chosen to minimize the objective function in the direction u_j subject to the constraints. Thus, from x^{k-1}, CCD does a line search in the direction u_j until a minimum of f_0 is obtained or a boundary of the feasible set is encountered. The search directions used are $u_1, u_2, \ldots, u_n, u_1, u_2, \ldots$, where n is the number of variables. The algorithm terminates when a point x^k is obtained from which no coordinate direction is a direction of descent.

The use of a CCD algorithm is motivated by the fact that the coefficients b_j of the linear objective function in problem (3) are by construction strictly positive, so that reductions in the objective function can be obtained by successively reducing the values x_j. Because of the algorithm's simplicity and low computational cost per iteration, it is easy to apply to larger and larger problems. Unfortunately, the method can converge to a point far from the constrained optimum even if the constraint set is convex. For example, consider the problem $\min x_1 + 2x_2$ subject to $x_1 + x_2 \geq 1$, $x_1 \leq 1$, $x_2 \leq 1$. If $x^0 = (\frac{3}{4}, \frac{3}{4})^T$ is the starting point, the algorithm converges to $x^1 = (\frac{1}{4}, \frac{3}{4})^T$. Thus, although CCD can be expected to reduce the objective value somewhat before it stops, it is in general useful only for finding approximate solutions, as discussed further in §4.

The four general-purpose codes are listed below in order of increasing sophistication.

EA3: The ellipsoid algorithm as implemented by Kupferschmid and Ecker [8].

GRG2: The generalized reduced gradient algorithm as implemented by Lasdon et al., [9].

SQP: A sequential quadratic programming algorithm, as implemented in subroutine VMCWD by Powell [12], and using the quadratic programming algorithm of Goldfarb and Idnani, [6], as implemented by Powell. At each major iteration of SQP a search direction is found by solving a quadratic programming subproblem that involves a quasi-Newton approximation of the Hessian of the Lagrangian and linearization of all the problem constraints.

IMSL: The sequential quadratic programming algorithm of Schittkowski [13], also with the quadratic programming algorithm of Goldfarb and Idnani [6], as implemented in the IMSL Library [7] subroutine DN2ONG. Schittkowski's algorithm uses an active-set strategy to identify that subset of constraints believed to be active at optimality. At each major iteration, the quadratic programming subproblem uses a linearization of only this subset of constraints and is probably easier to solve than if all the constraints were included.

No theoretical limit is imposed on problem size by any of the algorithms, but it was necessary to increase the computer memory allocated by GRG2, SQP, and IMSL to use them for solving problems with 738 constraints.

To compare the performance of the five algorithms we use error-versus-effort curves as in [4]. The cumulative time required for the calculations of each algorithm is recorded as PSCPU time (problem state CPU time) after each iteration along with the current x^k and its objective function value. For each problem the same feasible starting point x^0 given above is used for all five algorithms.

The performance evaluation technique of plotting solution error versus PSCPU time avoids the issue of termination criteria (which differ from one algorithm to another) by comparing the results of the algorithms under study over a wide range of error levels and throughout their convergence trajectories. In running our computational experiments, we turn off all termination testing, by setting convergence tolerances to zero, so as to permit each method to achieve the most accurate solution of which it is capable. When this is done, the algorithms under test typically stop either because identical iterates are eventually generated or because some benign numerical condition occurs to prevent the calculations from continuing (such as, for example, the ellipsoid matrix becoming computationally nonpositive-definite in the ellipsoid algorithm, or an uphill search direction being produced in a sequential quadratic programming algorithm). Thus, the lowest errors shown in Figs. 1–3 reflect the best accuracy attainable by the various methods, and the convergence trajectories illustrated would stop sooner if nonzero convergence tolerances were used.

All of the experiments reported here were performed on an IBM 3090S computer under the Michigan Terminal System, and all of the programs are written in Fortran and were compiled without optimization. The floating-point calculations were done using IBM double precision, which provides 56 fraction bits. Recall that each of the three problems Robots A, B, and C has the general form given in (4). Machine precision is around 10^{-16}.

Extensive numerical experiments provided a highly precise estimate of the optimal x^* and the optimal Lagrange multipliers λ_i^*. This information allowed us to calculate the error measures described below at every iteration.

As in [4], our error-versus-effort curves use a measure of solution error that is based on function values and scaled by the Lagrange multipliers for the problems. First, we calculate the combined error measure

$$e_k = |f_0(x^k) - f_0(x^*)| + \sum_{i=1}^{m} \lambda_i^* |f_i(x^k)|,$$

which takes into account both objective function value and constraint satisfaction. The error e_k approaches 0 as x^k approaches x^*. We scale this error measure by the starting value e_0 to obtain the combined measure

$$E_k = \frac{e_k}{e_0}$$

and plot $\log_{10} E_k$ in our error-versus-effort curves.

To guard against the possibility of assigning a low error value to a point infeasible for constraints that are inactive at optimality, we omit from the error-versus-effort curves any points that do not strictly satisfy all of the constraints that are slack at optimality.

The resulting graphs, given in Figs. 1–3 for the three robot problems, clearly display the convergence behavior of all five algorithms and show how quickly and by how much each method reduces the solution error from its initial value. Thus, these pictures make it easy to compare the performance of the algorithms on these problems.

In the review process for this paper other error measures were suggested including

$$d_k = \|x^k - x^*\|_2$$

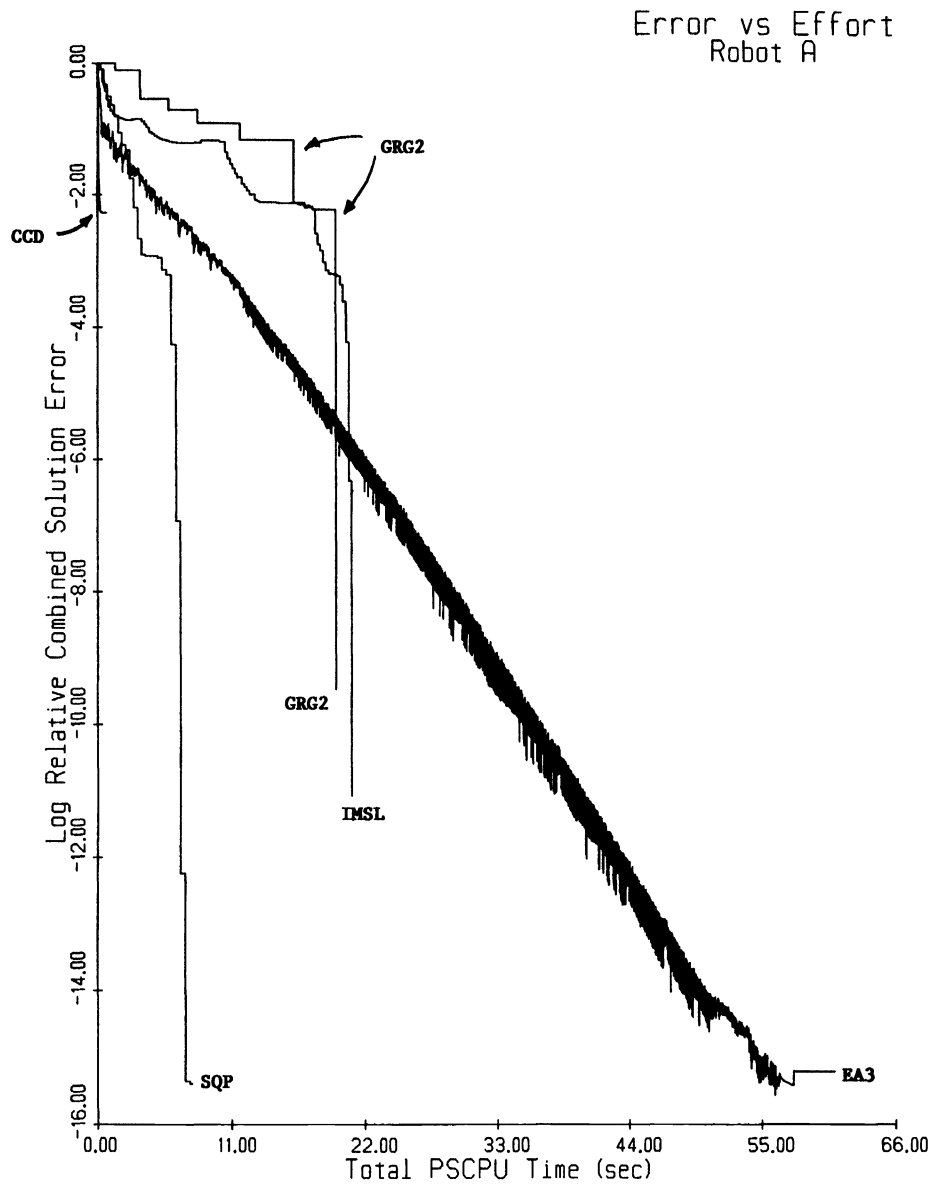


FIG. 1.

and

$$m_k = \max\{0, f_0(x^k) - f_0(x^*)\} + \sum_{i=1}^{m} \lambda_i^* \max\{0, -f_i(x^k)\}.$$

We tried these error measures in place of e_k to produce error-versus-effort curves for the three robot problems and found that the resulting curves were very similar to those in Figs. 1–3.

On Robots A, B, and C, CCD achieves errors of about 10^{-2}, 10^{-3}, and 10^{-1}, respectively, in less than 0.3 seconds. On each of the problems SQP is the next fastest algorithm in reaching the same error level as CCD but SQP takes 3.0 seconds, 5.7 seconds, and 0.8 seconds on

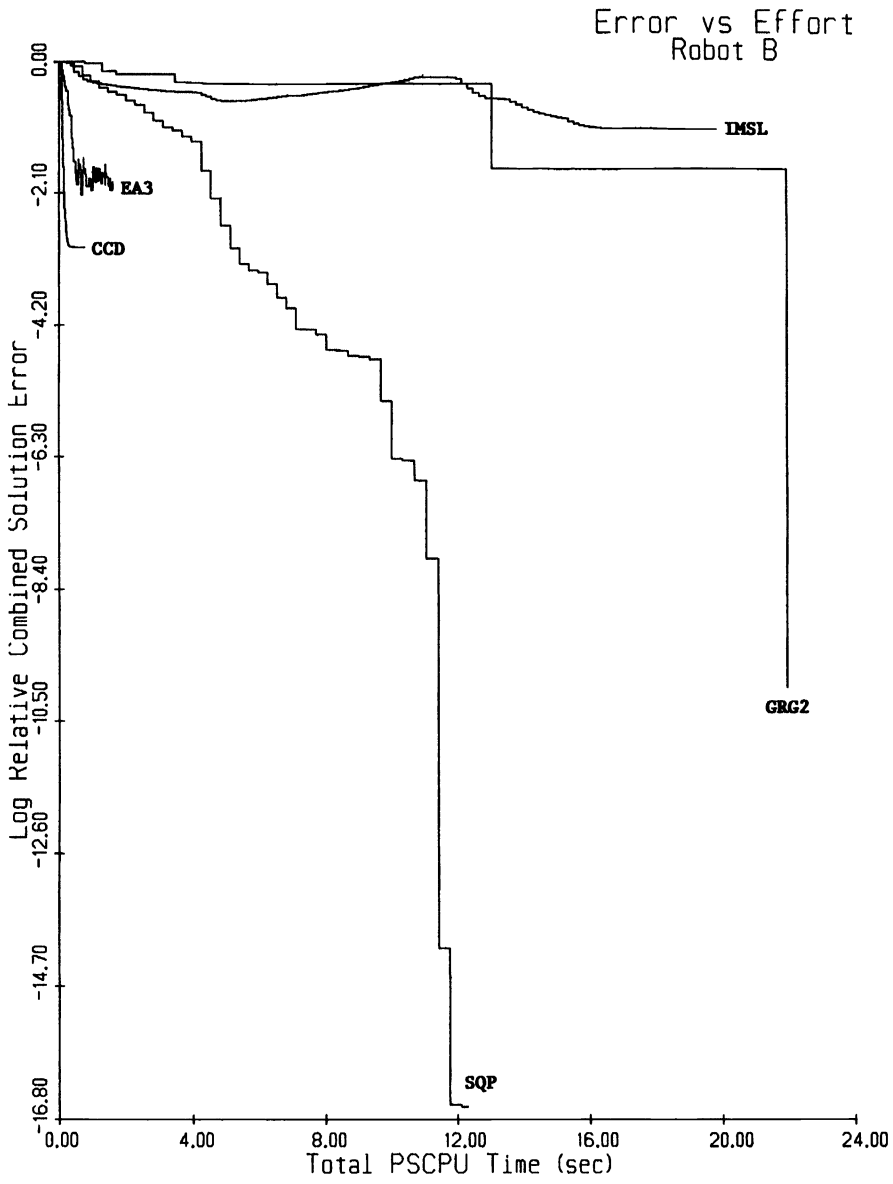


FIG. 2.

Robots A, B, and C, respectively, to reach this error level. From Fig. 3, it is not clear whether SQP or IMSL reaches the 10^{-1} error first, but the data file shows that SQP is first.

As illustrated in the discussion of CCD above, the algorithm can converge to a point far from the minimum even if the feasible region is convex, although it typically makes some progress in the direction of the minimum. The three robot problems have nonconvex feasible sets, so it is to be expected that CCD will probably converge to points that are better than the starting points but not as close to the optimal point as the solutions found by more sophisticated (and computationally expensive) algorithms.

If we were interested in finding a very accurate solution to each of these problems, say, with an error of 10^{-8}, then Figs. 1–3 show that SQP would be the fastest algorithm.

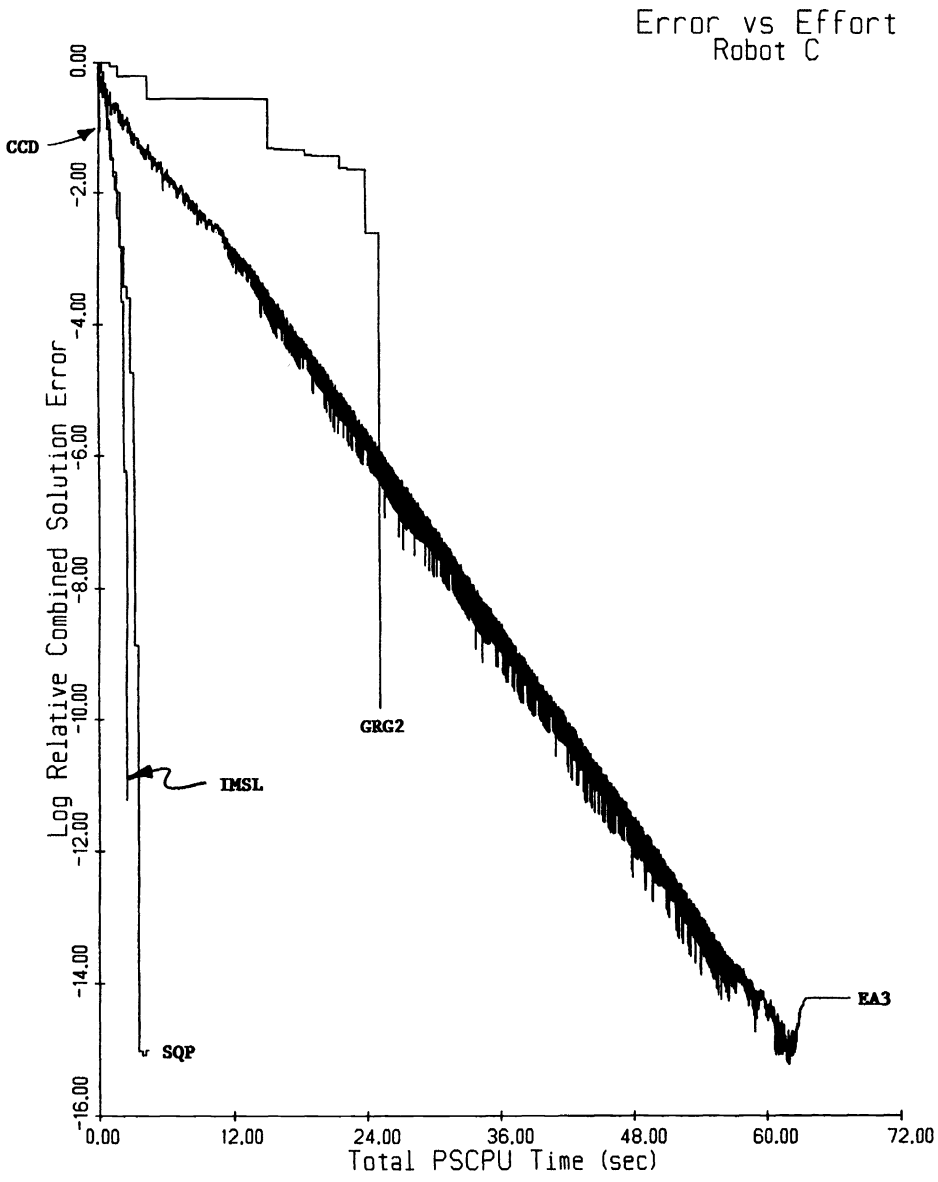


FIG. 3.

EA3 is successful in obtaining very accurate solutions to Robots A and C. Figures 1 and 3 show EA3's typically linear convergence. On Robot B, a quick reduction in error by EA3 is followed by a rather erratic trajectory. The ellipsoid algorithm can converge to a nonstationary point for a nonconvex problem (see [Problem 9.55 and pp. 315–322, 5]), and on this problem the nonconvex constraints lead EA3 to generate successive iterates that move away from the optimal point x^*.

GRG2 and SQP are successful in solving all three problems, as is clear from their error-versus-effort curves in Figs. 1–3. However, to reach the 10^{-8} error level on Robots A and B, GRG2 takes about twice the time required by SQP. On Robot C, GRG2 takes about five times as long as SQP in reaching the 10^{-8} error level.

To gain some insight into the reasons for the significantly longer times required by GRG2 to reach low error levels, we performed timing tests to see how much of the algorithm time is being expended in various parts of the algorithms. In Table 2, we give the results of our timing experiments for GRG2 and SQP on Robot A.

TABLE 2
Robot A times for GRG2 *and* SQP.

GRG2	Time in seconds	Percent of total
Function evaluations	13.384	67.55
Gradient evaluations	5.418	27.35
Line searching	.359	1.81
Computing the basis inverse	.109	.55
Other algorithm code	.543	2.74
	Total 19.812	
SQP		
Function evaluations	.612	8.39
Gradient evaluations	3.998	54.77
QP solver	2.493	34.15
Other algorithm code	.197	2.69
	Total 7.299	

The timing profiles for GRG2 and SQP on Robots B and C are very close to those reported in Table 2 in terms of the percent of total time spent in their various activities. These profiles changed very little throughout the iterations of the algorithms. For example, if we stopped both algorithms when they reached the 10^{-8} error level, the percent of total time spent in the various parts of the code was essentially the same as in Table 2.

From Table 2, we see that GRG2 spends almost 95% of its time evaluating functions and gradients versus about 63% for SQP. The 34% of its time that SQP spends on the QP solver to obtain a new search direction thus seems to be very well spent. Because Robot A has 738 constraints but only 12 that are active at optimality, the solution times required by both algorithms could be significantly decreased with an appropriate active-set strategy that did not require evaluation of all the constraints and gradients at every major iteration. SQP does not employ an active-set strategy, and GRG2 evaluates all constraints and gradients at every call to its function and gradient routines.

We have not included a table showing how EA3 spends its time, but on Robot A, EA3 spends about 91% of its time in function and gradient evaluations and 9% in the updates and other algorithm time.

The main reason for including the IMSL code in this comparison was that it is a sequential quadratic programming algorithm that uses an active-set strategy, and also uses the same QP algorithm used in SQP. Figure 1 shows that IMSL takes about three times as long as SQP to reach most error levels on Robot A. Figures 2 and 3 show that IMSL fails to solve Robot B but has slightly superior performance to SQP on Robot C.

The reason that the IMSL code takes a significantly longer time than SQP to solve Robot A is apparent from comparing the timing profiles for IMSL in Table 3 with those for SQP in Table 2.

In terms of percent of time spent in the various activities, the timing profiles for IMSL on Robots B and C are about the same as the times reported in Table 3.

TABLE 3
Robot A times for IMSL.

IMSL	Time in seconds	Percent of total
Function evaluations	2.554	12.11
Gradient evaluations	.350	1.66
QP solver	16.149	76.53
Other algorithm code	2.047	9.70
	Total 21.101	

From Table 3 it appears that the active-set strategy helps IMSL to significantly reduce the fraction of the time that it spends in function and gradient evaluations. IMSL spends about 14% of its time evaluating functions and gradients versus 63% for SQP. The reason that the total number of seconds required by IMSL is about three times that required by SQP is because IMSL spends 16.1 seconds (77%) of its time in the QP solver compared to the 2.5 seconds (34%) required by SQP.

As reported in [7], the IMSL code uses a revision of Powell's implementation of the Goldfarb–Idnani QP solver. Recall that Powell's implementation is also used in SQP. Although both codes solve exactly one QP subproblem at each major iteration, IMSL and SQP do different numbers of major iterations to reach a given error level. For example, to get to the 10^{-6} error level IMSL requires 104 major iterations and takes 21.73 seconds, but SQP only takes 21 major iterations and 6.48 seconds. Thus, the relatively better performance of SQP may indicate that it makes more effective use of the solutions that it finds to its larger QP subproblems.

To provide further insight into how much the active-set strategy helps IMSL, we performed another experiment. We allowed IMSL to use its active-set strategy, but we made IMSL waste time by evaluating all the constraints and gradients at each iteration. In this case IMSL took about twice as long to solve the problems.

To further investigate the difference in performance between IMSL and SQP on these problems, we studied the effectiveness of the IMSL active-set strategy in determining which of the 738 constraints of Robot A are active at optimality. To do this, we printed out, at each major iteration of the algorithm, the indices of the constraints thought to be active. Then we compared the list at each major iteration to the true active set and counted the listed constraints that are in the true active set (the number "right") and the number of constraints listed that are not in the true active set (the number "wrong"). These counts are plotted versus iteration number in Fig. 4.

From Fig. 4 it is clear that IMSL eventually identifies the active set, but only in its final major iterations. After 21 iterations (the number required by SQP to obtain its best result) IMSL's current estimate of the active set includes only 6 of the 12 active constraints but also includes 2 constraints that are not active. After 80 major iterations, IMSL's current estimate of the active set is even worse, still including only 6 of the 12 active constraints (not the same 6 as at iteration 21) but now also including 4 that are not active. Thus, for this problem, the main reason IMSL is not as fast as SQP is that its active-set strategy incorrectly identifies the active set and thus hurts rather than helps the overall performance of the algorithm.

4. Conclusions. Even using a coarse-grid, the robot problems of interest are fairly large. If only a low-accuracy solution is needed, then the special-purpose algorithm is faster than any of the general-purpose codes in solving these problems. If more accuracy is needed, then of the two sequential quadratic programming methods, SQP clearly outperforms IMSL on these

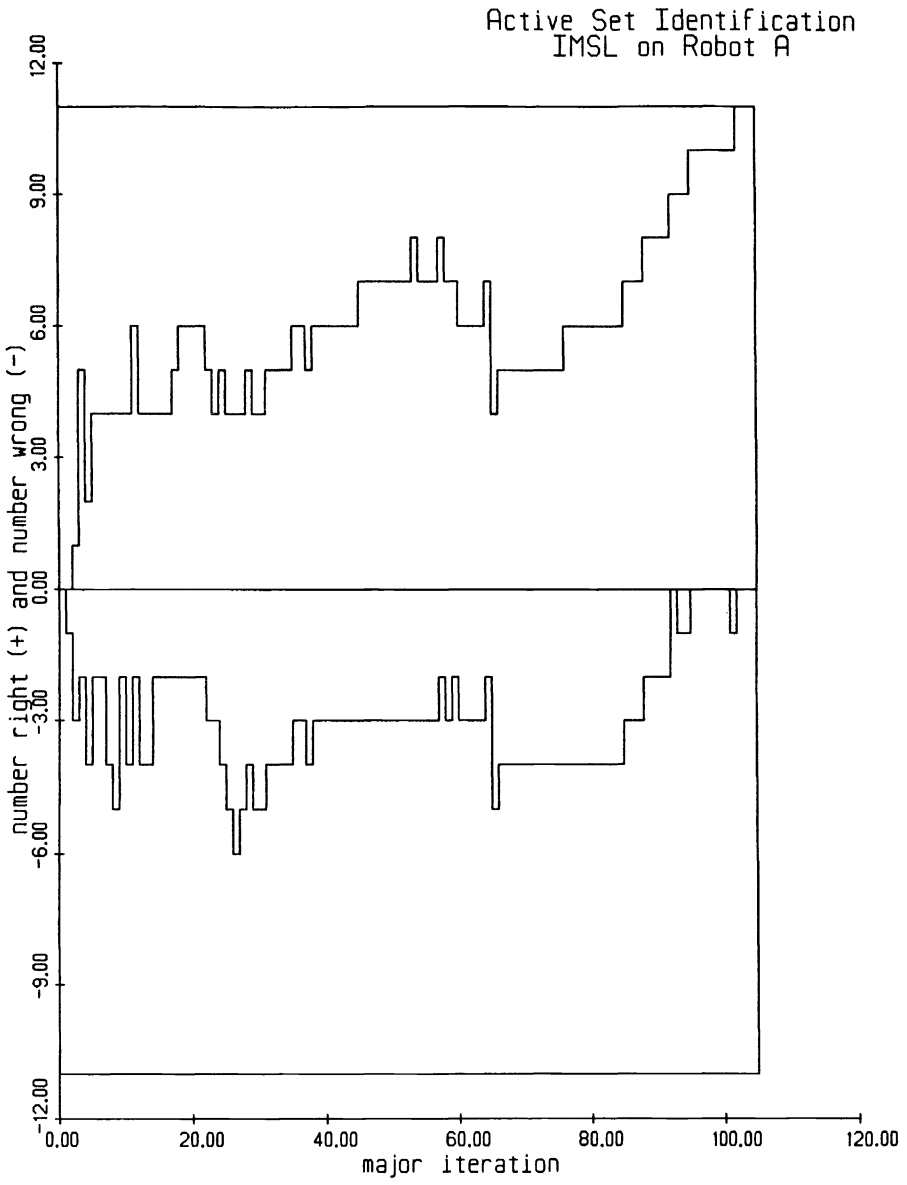


FIG. 4.

problems. Also, while GRG2 is successful in solving the problems, it is not competitive with SQP. The code EA3 fails to solve one of the problems and is dramatically outperformed by SQP (but not by GRG2) on the other two.

Acknowledgments. The authors acknowledge the assistance of Ahmed Ech-Cherif, who was at General Motors Research Laboratories when this work was begun, for constructing some of the computer programs used to define the robot problems in this study.

We also would like to acknowledge the Associate Editor, Margaret Wright, for her guidance in the revision of this paper.

REFERENCES

[1] M. S. BAZARAA AND C. M. SHETTY, *Nonlinear Programming*, Wiley, New York, 1979.

[2] C. DEBOOR, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.

[3] A. ECH-CHERIF, J. G. ECKER, M. KUPFERSCHMID, AND S. P. MARIN, *Robot Trajectory Planning by Nonlinear Programming*, General Motors Research Publication GMR-6460, General Motors Research Laboratories, Warren, MI, December 1988.

[4] J. G. ECKER AND M. KUPFERSCHMID, *A computational comparison of the ellipsoid algorithm with several nonlinear programming algorithms*, SIAM J. Control Optim., 23 (1985), pp. 657–674.

[5] ———, *Introduction to Operations Research*, Wiley, New York, 1988.

[6] D. GOLDFARB AND A. IDNANI, *A numerically stable dual method for solving strictly convex quadratic programs*, Math. Programming, 27 (1983), pp. 1–33.

[7] IMSL, Inc., *Users Manual Math/Library*, International Mathematical and Statistical Libraries, Sugarland, TX, April 1987.

[8] M. KUPFERSCHMID AND J. G. ECKER, EA3: *A Practical Implementation of an Ellipsoid Algorithm for Nonlinear Programming*, Dept. Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY, 1984.

[9] L. S. LASDON, A. WAREN, A. JAIN, AND M. W. RATNER, *Design and testing of a generalized reduced gradient code for nonlinear programming*, Trans. Math. Software, 4 (1978), pp. 34–50.

[10] S. P. MARIN, *Optimal Parametrization of Curves in R^N*, General Motors Research Publication GMR-4878, General Motors Research Laboratories, Warren, MI, August 1985.

[11] ———, *Optimal parametrization of curves for robot trajectory design*, IEEE Trans. Automat. Control, 33 (1988), pp. 14–31.

[12] M. J. D. POWELL, VMCWD: *A Fortran Subroutine for Constrained Optimization*, Paper DAMTP 1982/NA4, Dept. of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge, England, 1982.

[13] K. SCHITTKOWSKI, NLPQL: *A Fortran subroutine solving constrained nonlinear programming problems*, Ann. Oper. Res., 5 (1986), pp. 485–500.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1413–1439, November 1994

009

AN EFFICIENT BLOCK-ORIENTED APPROACH TO PARALLEL SPARSE CHOLESKY FACTORIZATION*

EDWARD ROTHBERG† AND ANOOP GUPTA‡

Abstract. This paper explores the use of a subblock decomposition strategy for parallel sparse Cholesky factorization in which the sparse matrix is decomposed into rectangular blocks. Such a strategy has enormous theoretical scalability advantages over more traditional column-oriented and panel-oriented decompositions. However, little progress has been made in producing a practical subblock method. This paper describes and evaluates an approach that is simple to implement, provides slightly higher performance than column (and panel) methods on small parallel machines, and has the potential to provide much higher performance on large parallel machines.

Key words. sparse Cholesky factorization, systems of linear equations, parallel computing, supernodes, scalability

AMS subject classifications. 65f05, 65f50

1. Introduction.

The Cholesky factorization of sparse symmetric positive definite systems is an extremely important computation, arising in a variety of scientific and engineering applications. Sparse Cholesky factorization is unfortunately also extremely time-consuming, and is frequently the computational bottleneck in these applications. Consequently, there is significant interest in performing the computation on large parallel machines. Several different approaches to parallel sparse Cholesky factorization have been proposed. While great success has been achieved for small parallel machines, success has unfortunately been quite limited for larger machines.

Virtually all parallel approaches to sparse Cholesky factorization [4], [12], [20] perform a one-dimensional (1-D) decomposition of the sparse matrix. That is, they distribute either rows or columns of the matrix among processors. Such a decomposition has two major limitations. The first is that it produces enormous volumes of interprocessor communication. Communication grows linearly in the number of processors [14], resulting in communication volumes that are difficult to sustain on all but the smallest parallel machines. The second limitation is that a 1-D decomposition produces extremely long critical paths. Since the critical path represents a lower bound on parallel runtime, parallel speedups are severely limited. As a result, few processors can be used effectively on the problem.

Both of these limitations can be overcome (in theory) by moving to a subblock or two-dimensional (2-D) decomposition. Such a decomposition has been shown to be extremely effective for parallel dense factorization [21], [27]. It is not clear, however, whether a similar decomposition would be practical for sparse problems. A few investigations [2], [26], [28] have been performed, but these contained little or no exploration of practical algorithms. This paper provides a detailed analysis of a new block-oriented algorithm, including performance results from an efficient implementation.

This paper focuses on two practical and important issues related to a 2-D decomposition approach. The first is implementation complexity. The fact that most sparse factorization methods use a 1-D decomposition indicates that this decomposition is more natural. A block approach might significantly complicate the implementation. The second issue is the efficiency

*Received by the editors July 20, 1992; accepted for publication (in revised form) September 22, 1993. This research is supported under Defense Advanced Research Projects Agency contract N00039-91-C-0138.

†Department of Computer Science, Stanford University, Stanford, California 94305 (rothberg@cs.stanford.edu).

‡Department of Computer Science, Stanford University, Stanford, California 94305 (ag@pepper.stanford.edu). The work of this author was supported by a National Science Foundation Presidential Young Investigator Award.

of a parallel block-oriented method for practical machine sizes. While parallel scalability arguments can be used to show that a block approach would give better performance than a column approach for extremely large parallel machines, these arguments have little to say about how well a block approach performs on smaller machines.

Regarding complexity, we find that a block approach need not be much more complicated than a column approach. We describe a simple strategy for performing a block decomposition and a simple parallel algorithm for performing the sparse Cholesky computation in terms of these blocks. The approach retains the theoretical scalability advantages of block methods. We term this block algorithm the *block fan-out method*, since it bears a great deal of similarity to the parallel column fan-out method [12].

Regarding efficiency, we explore this issue in two parts. We first consider a sequential block factorization code and compare its performance to that of a true sequential program to determine how much efficiency is lost in moving to a block representation. The losses turn out be quite minor with the block approach producing roughly 80% of the performance of an efficient sequential method. We then consider parallel block factorization, looking at the issues that potentially limit its performance. The parallel block method is found to give high performance on a range of parallel machine sizes. For larger machines, performance is good but not excellent, primarily due to load balance problems. We quantify the load imbalances and investigate the causes.

This paper is organized as follows. We begin in §2 with some background on sparse Cholesky factorization. Section 3 then discusses our experimental environment, including a description of the sparse matrices we use as benchmarks and the machines we use to study the parallel block factorization approach. Section 4 describes our strategy for decomposing a sparse matrix into rectangular blocks. Section 5 describes a parallel method that performs the factorization in terms of these blocks. Section 6 then evaluates the parallel method, both in terms of communication volume and achieved parallel performance. Section 7 gives a brief discussion, and finally conclusions are presented in §8.

2. Sparse Cholesky factorization. The goal of the sparse Cholesky computation is to factor a sparse symmetric positive definite n-by-n matrix A into the form $A = LL^T$, where L is lower triangular. The computation is typically performed as a series of three steps. The first step, *heuristic reordering*, reorders the rows and columns of A to reduce *fill* in the factor matrix L. The second step, *symbolic factorization*, performs the factorization symbolically to determine the nonzero structure of L given a particular reordering. Storage is allocated for L in this step. The third step is the *numerical factorization*, where the actual nonzero values in L are computed. This step is by far the most time-consuming, and it is the focus of this paper. We refer the reader to [13] for more information on these steps.

The following pseudo-code performs the numerical factorization step:

```
1. for k = 1 to n do
2.     for i = k to n do
3.         L_ik := L_ik / sqrt(L_kk)
4.     for j = k + 1 to n do
5.         for i = j to n do
6.             L_ij := L_ij - L_ik L_jk
```

The computation is typically expressed in terms of columns of the sparse matrix. Within a column-oriented framework, steps 2 and 3 are typically thought of as a single operation, called a column division or cdiv(k) operation. Similarly, steps 5 and 6 form a column modification, or cmod(j, k), operation. The computation then looks like

```
1. for k = 1 to n do
2.     cdiv(k)
3.     for j = k + 1 to n do
4.         cmod(j, k)
```

Only the nonzero entries in the sparse matrix are stored, and the computation performs operations only on nonzeros.

This column-oriented formulation of the sparse factorization has formed the basis of several parallel sparse factorization algorithms, including the fan-out method [12], the fan-in method [4], and the distributed multifrontal method [20]. The details of these various methods are not relevant to our discussion, so we refer the reader to the relevant papers for more information. We simply note that for each of these methods, communication volume can be shown to grow linearly in the number of processors [14]. Since available communication bandwidth in a multiprocessor typically grows much more slowly, this communication growth represents a severe scalability limitation.

Recent research in parallel sparse Cholesky factorization [3] has shown that the communication needs of column-oriented sparse factorization can be greatly reduced. Through limited replication of data and careful assignment of tasks to processors, communication can be made to grow as the square root of the number of processors, thus improving scalability. Communication volume is not the only thing that limits scalability in column-oriented approaches, however. A column formulation also leads to very long critical paths, thus placing a large lower bound on parallel runtime. For a dense n-by-n matrix, the sequential computation requires $O(n^3)$ operations while the length of the critical path and thus the best case parallel runtime is $O(n^2)$. Similar bounds apply for sparse problems.

An alternative formulation of the factorization problem divides the matrix into rectangular subblocks. The dense Cholesky computation, expressed in terms of subblocks, would look like

```
1. for K = 1 to N do
2.     L_KK := Factor(L_KK)
3.     for I = K + 1 to N do
4.         L_IK := L_IK L_KK^-1
5.     for J = K + 1 to N do
6.         for I = J to N do
7.             L_IJ := L_IJ - L_IK L_JK^T
```

In this pseudo-code, I, J, and K iterate over rows and columns of subblocks. The Factor() routine performs dense Cholesky factorization of a diagonal subblock. This formulation leads to greatly reduced communication volumes and exposes significantly more concurrency. Specifically, communication volumes can be shown to grow as the square root of the number of processors, and the critical path can be shown to grow as $O(n)$ [27]. It is an open question whether this formulation can be applied efficiently to parallel sparse factorization, and this is the question we address here.

Before we begin our discussion of a block decomposition of the sparse matrix, we first discuss two important concepts in sparse factorization that will be relevant to our presentation. The first is the concept of a *supernode* [6]. A supernode is a set of adjacent columns in the factor matrix L whose nonzero structure consists of a dense lower-triangular block on the diagonal and an identical set of nonzeros for each column below the diagonal. Supernodes arise in any sparse factor, and they are typically quite large. By formulating the sparse factorization

computation as a series of supernode-supernode modifications, rather than column-column modifications as described before, the computation can make substantial use of dense matrix operations. The result is substantially higher performance on vector supercomputers and on machines with hierarchical memory systems. For more details on supernodal factorization, see [1], [6], [22], [23]. The regularity in the sparse matrix captured by this supernodal structure will prove useful in this paper for producing an effective decomposition of the sparse matrix into rectangular blocks. We will return to this issue shortly.

One thing we should note is that it is possible to improve the performance of parallel sparse column-oriented methods by grouping sets of adjacent columns from within the same supernode into *panels* and distributing these panels among the processors [22], [23]. We use the term column-oriented in this paper to refer to methods that treat columns as indivisible entities. Thus, panel methods fit this description. When we compare the performance of our parallel block-oriented method to that of a parallel column-oriented method, we will actually compare against the higher-performance parallel panel method.

Another important notion in sparse factorization is that of the *elimination tree* of the sparse matrix [19], [25]. This structure concisely captures important dependency information. If each column of the sparse matrix is thought of as a node in a graph, then the elimination tree is defined by the following parent relationship:

$$parent(j) = \min\{i | l_{ij} \neq 0, i > j\}.$$

It can be shown that a column is modified only by descendent columns in the elimination tree, and equivalently that a column modifies only ancestors [19]. The most important property captured in this tree for parallel factorization is the property that disjoint subtrees are independent, and consequently can be processed in parallel. This fact will be relevant later in this paper.

3. Experimental environment. Since one of our interests in this paper is to consider practical performance issues for block methods, we will present several performance numbers for realistic sparse matrices factored on real machines. This section briefly describes both the matrices that we use as benchmarks and the machine on which we perform the factorizations.

3.1. Benchmark matrices. The benchmark matrices we consider in this paper are drawn from the Harwell/Boeing sparse matrix test set [7]. Since our interest is on factorization on large machines, we have chosen some of the largest sparse matrices in the collection. Table 1 gives brief descriptions of the matrices. Table 2 gives information about the factors of these matrices. The first column of numbers shows a count of the number of floating-point operations required for the factorization. The second column gives the number of nonzeros in L. The third gives the number of supernodes in the factor matrices. Note that all matrices except the two grid problems are preordered using the multiple minimum degree ordering heuristic [18] before being factored. A simple nested dissection ordering is used for the grid problems.

3.2. Target machine. Although many of the results presented in this paper will be machine-independent, the paper will also include some performance numbers from real parallel machines. We now briefly describe the parallel machines that are considered.

3.2.1. Moderately parallel machines. Performance numbers for sequential and moderately parallel machines are obtained from a Silicon Graphics 4D/380 multiprocessor. The 4D/380 contains eight high-performance RISC processors, each consisting of an MIPS R3000 integer unit and an R3010 floating-point coprocessor. The processors execute at 33 MHz, and are rated at 27 MIPS and 4.9 double-precision LINPACK MFLOPS. They are connected with a bus having a peak throughout of approximately 67 MBytes per second.

TABLE 1
Benchmarks.

	Name	Description	Equations	Nonzeros
1.	GRID100	5-point discretization of 2D region	10,000	39,600
2.	GRID200	Larger instance of 1	40,000	159,200
3.	BCSSTK15	Module of an offshore platform	3,948	113,868
4.	BCSSTK16	Corps of Engineers dam	4,884	285,494
5.	BCSSTK17	Elevated pressure vessel	10,974	417,676
6.	BCSSTK18	R.E. Ginna nuclear power station	11,948	137,142
7.	BCSSTK29	Boeing 767 rear bulkhead	13,992	605,496

TABLE 2
Factor matrix statistics.

	Name	FP ops	Nonzeros in L	Supernodes
1.	GRID100	15,707,205	250,835	6,672
2.	GRID200	137,480,183	1,280,743	26,669
3.	BCSSTK15	165,039,042	647,274	1,295
4.	BCSSTK16	149,105,832	736,294	691
5.	BCSSTK17	144,280,005	994,885	2,595
6.	BCSSTK18	140.919.771	650,777	7,438
7.	BCSSTK29	393,059,150	1,680,804	3,231

Each processor in the 4D/380 has a 64-KByte instruction cache, a 64-KByte first-level data cache, and a 256-KByte second-level data cache. The caches play a crucial role in determining performance. Memory references that hit in the first-level cache are serviced in a single cycle. References that miss in both levels of the cache require roughly 50 cycles to service, and they bring 4 16-byte cache lines into the cache. In contrast, a floating-point multiply requires 5 cycles and a floating-point add requires 2 cycles.

We also provide performance numbers from the Stanford DASH machine, a 48-processor distributed-shared-memory machine [17]. The DASH machine is built out of a network of 4-processor SGI 4D/340 nodes. Each 4D/340 node contains some portion of the global shared memory. In the DASH machine, a processor memory reference that misses in both levels of its cache is serviced in roughly 30 cycles if the requested location is held in the memory local to that processor. It brings a 16-byte line into the cache. A cache miss to a memory location held in a nonlocal memory requires roughly 100 cycles and again brings in a 16-byte cache line. Our factorization implementation for the Stanford DASH machine explicitly places matrix data in the memory local to the processor that owns that portion of the matrix.

3.2.2. Parallel machine simulator. To provide a more detailed understanding of the performance of parallel machines on this computation, this paper also makes use of multiprocessor simulation. To keep simulation costs manageable, we perform this simulation in terms of high-level factorization tasks. A single task might represent a block modification operation or the transmission of a large message from one processor to another. The costs of the individual high-level tasks are obtained through a simple performance model. The parallel simulation is performed as a discrete-event simulation of these tasks.

The three most important costs that are modeled in this simulation are the costs of performing floating-point operations, fetching data from the local memory of a processor, and communicating data between processors. We now describe our model for each in more detail.

An important cost in a parallel factorization will clearly be the cost of executing the machine instructions that perform the required floating-point operations. For the sake of

normalization, we assume that one floating-point operation requires one time unit. Note that this machine instruction term is meant to capture not only the cost of the single instruction that actually performs the floating-point operation, but also the costs of all instructions that are required to support a floating-point operation (such as index computations and memory load instructions).

Another, potentially even more important, component of performance is the cost of moving data between a processor's memory and its cache. Our assumption is that the cost per double-precision word of fetching data from memory into the cache is roughly 5 times the cost of a floating-point operation. This number is quite accurate for the SGI 4D/380 and is a reasonable estimate for a wide range of current generation hierarchical-memory machines as well. The cache is assumed to be large enough to hold three 32-by-32 blocks of data. To simplify our simulation, we further assume that all reuse of cached data occurs within block operations. That is, we assume that each block operation begins with an empty cache.

While this computational cost model may appear too simplistic to capture the intricacies of current and future microprocessors, which might include such complex features as heavily pipelined floating-point units and nonblocking caches, we believe this model will in fact provide relatively accurate estimates. No matter what the internal structure of a processor, we believe that its performance for this computation can be understood using the answers to two questions. First, what performance is achieved when virtually no memory references miss in cache? And second, what performance is achieved when all references miss in cache? While heavier pipelining will allow the processor to hide some of the latencies of memory accesses, at the same time memory latencies will continue to increase so there will be more latencies to hide. As a result, we believe that the processor will have to pay some cost for each cache miss.

The third cost that is modeled is the cost of interprocessor communication. Our model assumes that all communication takes place in the form of interprocessor messages. It also assumes that messages are handled by a message coprocessor and therefore cost the sending and receiving processors nothing to process. The true cost of a message is the time it spends in the interconnection network. Our model assumes that this time is determined by the length of the message, the available communication bandwidth between the sending and receiving processors, and the message start-up cost. Communication bandwidth is assumed to be one-tenth of computation bandwidth. That is, a processor can perform ten floating-point operations in the time required to send one floating-point value. This number is roughly average for today's parallel machines. The Intel Touchstone DELTA machine, for example, has a roughly 10 to 1 computation to communication bandwidth ratio (ignoring contention issues). The Intel Paragon machine will have a roughly 2 to 1 ratio. On the other hand, the Thinking Machines CM-5 has a roughly 50 to 1 ratio. Communication start-up is assumed to require 500 times as long as a floating-point operation. This number is somewhat aggressive, but it is within the range of what can be observed on current machines.

For an example of how this performance model would be applied, consider the following three example operations: (1) send a 32-by-32 block of data from one processor to another; (2) multiply the received block by another 32-by-32 block; (3) add the result into another 32-by-32 block. Operation 1 would send 1024 double-precision values and thus would require 10240 time units. Operation 2 would load 3 blocks into the processor cache, requiring 15360 time units, and would perform 65536 floating-point operations, requiring 65536 time units. Finally, operation 3 would load two blocks into the cache, requiring 10240 time units, and would perform 1024 floating-point operations, requiring 1024 time units.

We believe our model captures the most important aspects of parallel machine performance for modern multiprocessors with one exception. Our model does not capture the effect of

message contention in the processor interconnection network. Since the effect of contention is extremely difficult to model accurately, we instead discuss the issue in a qualitative way in sections where the performance model is used.

4. Block formulation. Having described our evaluation environment, we now move on to the question of how to structure the sparse Cholesky computation in terms of blocks. Our first step in describing a block-oriented approach is to propose a strategy for decomposing the sparse matrix into blocks. Our goal in this decomposition is to retain as much of the efficiency of a sequential factorization computation as possible. Thus, we will keep a careful eye on the amount of computational overhead that is introduced.

4.1. Block decomposition. When dividing a matrix into blocks, we believe the three most important issues are (1) producing blocks with simple internal nonzero structures, so that block operations can be performed efficiently; (2) producing blocks that interact with other blocks in simple ways, so that bookkeeping overheads are minimized; and (3) producing blocks that are as dense as possible, so that per-block computation and storage overheads are minimized. With these goals in mind, the approach we take to decomposing the sparse matrix into blocks is to perform a global partitioning on the matrix, guided by the supernodal structure. More precisely, we divide the columns of the matrix $(1 \ldots n)$ into contiguous sets $(\{1 \ldots p_2 - 1\}, \{p_2 \ldots p_3 - 1\}, \ldots, \{p_N \ldots n\}$, where N is the number of partitions and p_i is the first column in partition i). All columns within a particular partition must be members of the same supernode (although a partition will frequently be a subset of a supernode). An identical partitioning is performed on the rows. A simple example is shown in Fig. 1. A block L_{IJ} (we refer to partitions using capital letters) is then the set of nonzeros that fall simultaneously in rows $\{p_I \ldots p_{I+1} - 1\}$ and columns $\{p_J \ldots p_{J+1} - 1\}$.

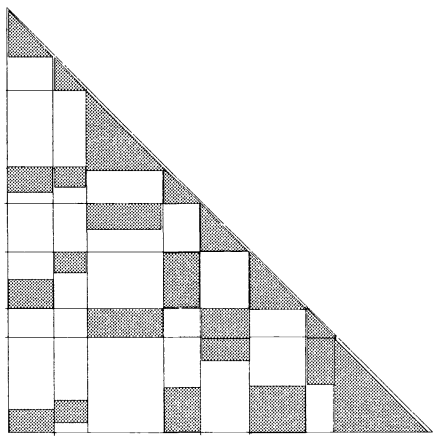


FIG. 1. *Example of globally partitioned matrix.*

This global partitioning approach addresses the above-mentioned issues quite well. Each block has a very simple nonzero structure. Since the block is a portion of a supernode, all rows in the block are dense. The blocks also share common boundaries. As a result, block interactions are extremely regular. As we will soon demonstrate, this decomposition leads to a computation structure where a block interacts with a block above it to produce a modification to a block to its right. Without these common boundaries, block modifications would be quite complicated with portions of blocks modifying portions of other blocks (see Fig. 2).

One issue that this distribution scheme does not address is the block density issue. The global nature of the partitions does not allow the blocks to be tailored to match the local sparsity

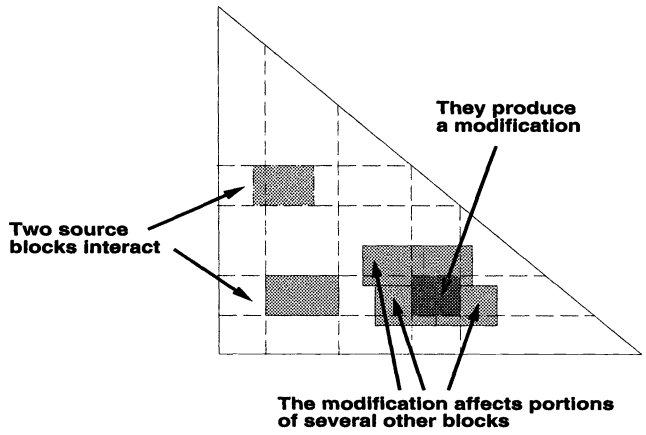


FIG. 2. *Example of irregular block interaction. Dashed lines indicate boundaries of affected areas.*

structure of the matrix. We will see in the next section that this is not actually a significant problem. While blocks will often not be completely dense, this sparsity has little effect on the efficiency of the overall computation.

Before proceeding, we note that Ashcraft [2] proposed a similar decomposition strategy independently.

4.2. Structure of the block factorization computation. One important goal we had in choosing this block decomposition was to retain as much efficiency as possible in the block factorization computation. We now describe a sequential algorithm for performing the factorization in terms of these blocks and evaluate that algorithm's efficiency. The parallelization of the sequential approach that we derive here will be described later.

At one level, the factorization algorithm expressed in terms of blocks is quite obvious. The following pseudo-code, a simple analogue of dense block Cholesky factorization, performs the factorization. Note that I, J, and K iterate over the partitions in the sparse matrix.

```
1. for K = 1 to N do
2.     L_KK := Factor(L_KK)
3.     for I = K + 1 to N with L_IK ≠ 0 do
4.         L_IK := L_IK L_KK^{-1}
5.     for J = K + 1 to N with L_JK ≠ 0 do
6.         for I = J to N with L_IK ≠ 0 do
7.             L_IJ := L_IJ − L_IK L_JK^T
```

The above pseudo-code works with a column of blocks at a time. Steps 2 through 4 divide block column K by the Cholesky factor of the diagonal block. Steps 5 through 7 compute block modifications from all pairs of blocks in column K. We store the blocks by columns, so that all blocks in a column can be easily located. One thing to note is that step 7 modifies some destination block L_{IJ} whose location in memory cannot easily be determined from the locations of the source blocks. To make this step efficient, we keep a hash table of all blocks (hashing on I and J).

Now consider the implementation of the individual operations in the pseudo-code. The block factorization in step 2 is quite straightforward to implement. Diagonal blocks are guaranteed to be dense, so this step is simply a dense Cholesky factorization. The multiplication by the inverse of the diagonal block in step 4 is also quite straightforward. This step does not actually compute the inverse of L_{KK}. Instead, it solves a series of triangular systems. While

the block L_{IK} is not necessarily dense, the computation can be performed without consulting the nonzero structure of the block.

The remaining step in the above pseudo-code, step 7, is both the most important and the most difficult to implement. It is the most important because it sits within a doubly nested loop and thus performs the vast majority of the actual computation. It is the most difficult because it works with blocks with potentially different nonzero structures and it must somehow reconcile these structures. More precisely, recall that a single block in L consists of some set of dense rows from among the rows that the block spans (see the example in Fig. 1). When a modification is performed in step 7 above, the structure of L_{IK} determines the set of rows in L_{IJ} that are affected. Similarly, the structure of L_{JK} determines the set of *columns* in L_{IJ} that are affected.

Block modification is most conveniently viewed as a two stage process. A set of modification values is computed in the first stage, and these values are subtracted from the appropriate entries in the destination block in the second, or scatter stage. The first stage can be performed as a dense matrix-matrix multiplication. The nonzero structures of the source blocks L_{IK} and L_{JK} are ignored temporarily; the two blocks are simply multiplied to produce a modification.

During the second stage, the resulting modification must be subtracted from the destination. The most simple case occurs when the modification has the same nonzero structure as the destination block. We have coded our dense matrix-matrix multiplication routine as a multiply-subtract (i.e., $C = C - AB^T$), rather than a multiply-add, so the destination block can be used as the destination directly without the need for a second scatter stage.

Consider the more difficult case where the nonzero structures differ. The first step in this case is to compute a set of *relative indices* [25]. These indices indicate the corresponding position in the destination for each row in the source. Two sets of relative indices are necessary to scatter a single block modification; rel_i, the affected set of rows and rel_j, the affected set of columns. The next step is to use these relative indices to scatter the computed modification into the destination block.

The computation of relative indices is quite expensive in general, since it requires a search through the destination to find the row corresponding to a given source row. Fortunately, such a search is only rarely necessary due to an important special case. When the destination block has dense structure, the relative indices bear a trivial relationship to the source indices. Note that the rel_j indices always fall into this category, since the destination block always has dense column structure. We will be more precise shortly about exactly how often relative index computations are necessary.

The scatter operation is also somewhat expensive, and it is much more prevalent than relative index computation. The frequency with which relative index computations and scatters must be performed will be considered shortly.

In summary, the efficiency of a block modification operation depends heavily on the nonzero structures of the involved blocks.

- The best case occurs when the modification has the same structure as the destination. In this case, the $C = C - AB^T$ operation can use the destination block as its destination.
- The next best case occurs when the destination block is dense. The modification must be scattered, but the relative indices can be computed inexpensively.
- The worst case occurs when the modification has different structure from the destination and the destination block is sparse. The modification must be scattered, and relative indices are relatively expensive to compute.

4.3. Performance of block factorization. We now look at the performance obtained with a sequential program that uses a block decomposition and block implementation. Since

our goal is to create an efficient *parallel* approach, performance is studied for the case where the matrix is divided into relatively small blocks. The blocks should not be too small, however, because of the overheads that will be associated with block operations. We consider 16-by-16, 24-by-24, and 32-by-32 block sizes. To produce blocks of the desired size B, we form partitions that contain as close to B rows/columns as possible. For example, with a block size of $B = 16$, a supernode of width 51 would be split into three partitions of size 17. Since partitions represent subsets of supernodes, some partitions will naturally be much smaller than B.

The performance obtained with the sequential block approach on a single processor of the SGI 4D/380 is shown in Fig. 3. This performance is expressed as a fraction of the performance obtained with an efficient sequential code (a supernode-supernode left-looking method; among the most efficient sequential approaches [23]). From the figure, it is clear that the block approach is reasonably efficient. Typical efficiencies are roughly 65% for a block size of 16 and roughly 75% for a block size of 32. We will discuss shortly the reasons why three of the matrices, GRID100, GRID200, and BCSSTK18, achieve significantly lower performance.

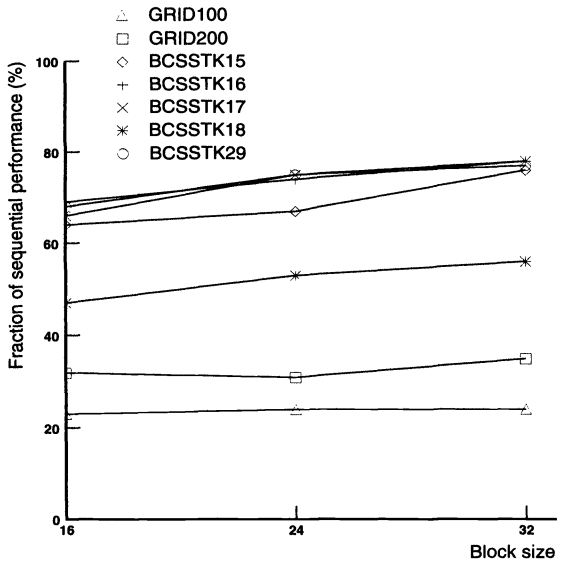


FIG. 3. *Performance of a sequential block approach, relative to a sequential left-looking supernode-supernode approach.*

Our earlier discussion indicated that the performance of the block method might suffer because of the need for relative index calculations and scattering. To gauge the effect of these two issues on overall performance, Table 3 relates the number of distinct scatters performed and the number of relative indices computed (for $B = 16$) to the number of floating-point operations performed in the factorization. The numbers are quite similar for the other block size choices. The first column compares the number of distinct relative indices computed against the number of floating-point operations. The second column compares distinct element scatters against floating-point operations. The table shows that even if relative index computations and scatters are much more expensive than floating-point operations, the related costs will still be small. Clearly, the vast majority of block modification operations produce a modification with the same structure as the destination block.

TABLE 3
Frequency of relative index computations and scatters for block method, compared with floating-point operations ($B = 16$).

Problem	Relative indices (relative to FP ops)	Scatters (relative to FP ops)
GRID100	0.37%	4.0%
GRID200	0.18%	2.4%
BCSSTK15	0.04%	1.6%
BCSSTK16	0.02%	1.4%
BCSSTK17	0.04%	1.8%
BCSSTK18	0.11%	2.6%
BCSSTK29	0.01%	1.0%

It is also interesting to compare relative indices and scatters to those performed by a true sequential method. Table 4 gives the relevant numbers. In this case, the comparison is with a sequential multifrontal method, where notions of relative indices and scatters are easily quantified. The comparison is relevant for the left-looking supernode-supernode method as well, since the two methods perform similar computations. Note that in most cases the block method performs fewer relative index computations and scatters.

TABLE 4
Frequency of relative index computations and scatters for block method, compared with sequential multifrontal method ($B = 16$).

Problem	Relative indices (relative to seq MF)	Scatters (relative to seq MF)
GRID100	78%	72%
GRID200	80%	69%
BCSSTK15	109%	105%
BCSSTK16	50%	88%
BCSSTK17	61%	90%
BCSSTK18	163%	91%
BCSSTK29	32%	40%

Ashcraft [2] has described methods for improving block structure and thus decreasing the need for scattering. It is our belief that a very simple block decomposition is more than adequate for keeping such costs in check.

4.4. Improving performance. It is clear from the previous section that the block method is generally quite efficient. Recall, however, that the method was much less efficient than a true sequential method for several problems. Data on relative index and scatter frequency showed that these were not the source of the losses. The losses are actually due to overheads in the block operations.

Consider a single block modification operation. It must find the appropriate destination block through a hash table, determine whether the source and destination blocks have the same structure, and then pay the loop start-up costs for the dense matrix multiplication to compute the modification. While these costs are trivial when all involved matrices are 32-by-32, in fact many blocks in the sparse matrix are quite small. In the case of matrix GRID100, for example, the average block operation when $B = 32$ performs only 96 floating-point operations, as compared with the 65536 operations that would be performed with 32-by-32 full blocks. The average number of floating-point operations per block operation across the whole benchmark

set is shown in Fig. 4. Note that this figure quite accurately predicts the performance numbers seen in the previous figure.

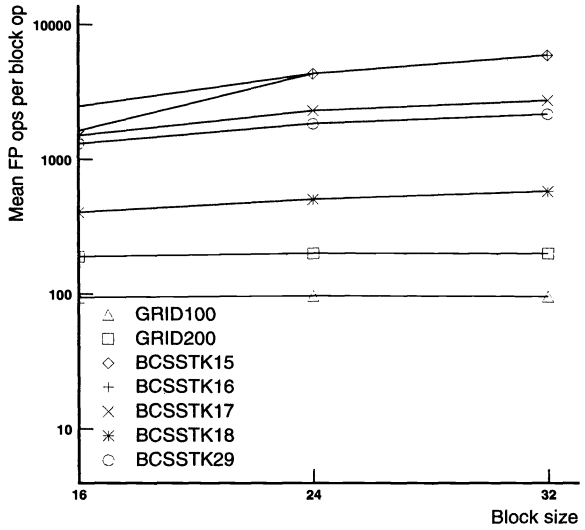


FIG. 4. *Average floating-point operations per block operation.*

The primary cause of small blocks in the block decomposition is the presence of small supernodes and thus small partitions. To increase the size of these partitions, we now consider the use of *supernode amalgamation* [5], [8] techniques. The basic goal of supernode amalgamation is to find pairs of supernodes that are nearly identical in nonzero structure. By relaxing the restriction that the sparse matrix only store nonzeros, some zeros can be introduced into the sparse matrix to make the sparsity structures of two supernodes the same. These supernodes can then be merged into one larger supernode. We refer the reader to [5] for more details on supernode amalgamation.

In Fig. 5 we show the average block operation sizes after amalgamation. It is clear that amalgamation significantly increases the block operation grain size.

Figure 6 shows relative performance levels after amalgamation. To be fair, we have computed these numbers relative to post-amalgamation sequential performance, since amalgamation improves sequential performance as well (by between 0% and 15%). The results indicate that amalgamation is quite effective at reducing overheads. Performance roughly doubles for GRID100, where the average task grain size for $B = 32$ increases from 96 floating-point operations to 597. Performance increases for the other matrices as well. With only two exceptions, block method performance for $B = 32$ is roughly 80% of that of a true sequential method. Performance falls off somewhat when $B = 24$, and it decreases further when $B = 16$, but the resulting efficiencies are still roughly 70%.

Note that our chosen range of blocks sizes, 16 to 32, is meant to span the range of reasonable choices. Blocks that are smaller than 16-by-16 would be expected to lead to large overheads. Indeed, performance was observed to fall off quite quickly for block sizes of less than 16. The marginal benefit of increasing the block size beyond 32-by-32 would be expected to be small. This expectation was also confirmed by the empirical results.

4.5. Block decomposition summary. This section has described a simple means of decomposing a sparse matrix into a set of rectangular blocks. The performance of a method based

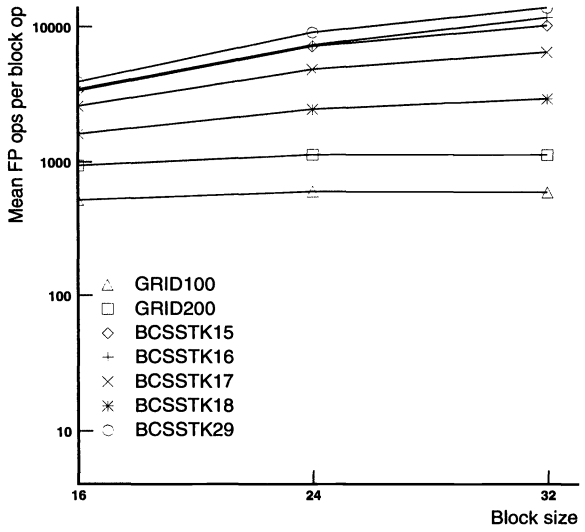


FIG. 5. *Average floating-point operations per block operation after supernode amalgamation.*

on such blocks on a sequential machine is nearly equal the performance of a true sequential method. Of course, our goal here is not an efficient sequential method, but, instead, an efficient parallel method. The next section will consider several issues related to the parallelization of the above approach.

5. Parallel block method. The question of how to parallelize the sequential block approach described so far can be divided into two different questions. First, how will processors cooperate to perform the work assigned to them? And second, what method will be used to assign this work to processors? This section will address these two questions in turn.

5.1. Parallel factorization organization. We begin our description of the parallel computation by assuming that each block will have some specific owner processor. In our approach, the owner of a block L_{IK} performs all block modification operations with L_{IK} as their destination. With this choice in mind, we present the parallel block fan-out algorithm in Fig. 7. The rest of this section will be devoted to an explanation of the algorithm.

The most important notion for the block fan-out method is that once a block L_{IK} is *complete*, meaning that it has received all block modifications and has been multiplied by the inverse of the diagonal block, then L_{IK} is sent to all processors that could own blocks modified by it. Blocks that could be modified by L_{IK} fall in block-row I or block-column I of L. When a block L_{IK} is received by a processor p (step 2 in Fig. 7), processor p performs all related modifications to blocks it owns. The block L_{IK} only produces block modifications when it is paired with blocks in the same column K. Thus, processor p considers all pairings of the received block L_{IK} with completed blocks it has already received in column K (these blocks are held in set $Rec_{K,p}$) to determine whether the corresponding destination block is owned by p (steps 10 and 11). If the destination L_{IJ} is owned by p ($map[L_{IJ}] = p$), then the corresponding modification operation is performed (steps 12 and 13). Each processor maintains a hash table of all blocks assigned to it, and the destination block is located through this hash table.

A count is kept with each block ($nmod[L_{IJ}]$), indicating the number of block modifications that still must be done to that block. When the count reaches zero, then block L_{IJ} is

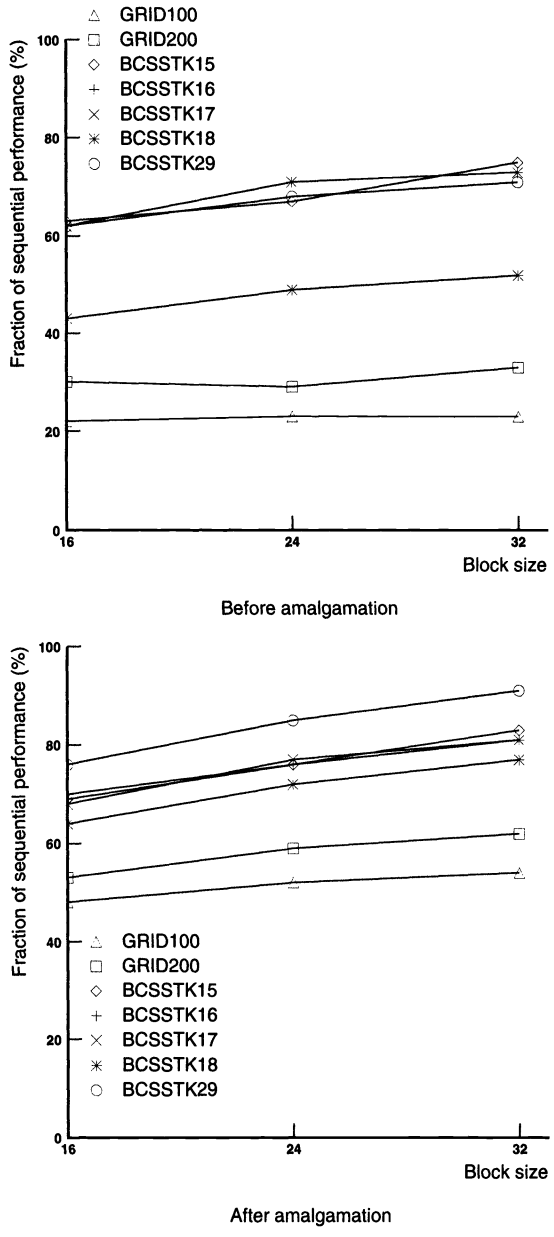


FIG. 6. *Performance of a sequential block approach, before and after supernode amalgamation, relative to a sequential left-looking supernode-supernode approach.*

ready to be multiplied by the inverse of L_{JJ} (step 20 if L_{JJ} has already arrived at p; step 6 otherwise). A diagonal block L_{JJ} is kept in $Diag_{J,p}$, and any blocks waiting to be modified by the diagonal block are kept in $Wait_{J,p}$. The sets $Diag$, $Wait$, and Rec can be kept as simple linked lists of blocks.

One issue that is not addressed in the above pseudo-code is that of block disposal. As described above, the parallel algorithm would retain a received block for the duration of the factorization. To determine when a block can be thrown out, we keep a count $ToRec_{K,p}$ of

```
1.  while some L_IJ with map[L_IJ] = MyID is not complete do
2.      receive some L_IK
3.      if I = K /* diagonal block */
4.          Diag_{K,MyID} := L_KK
5.          foreach L_JK ∈ Wait_{K,MyID} do
6.              L_JK := L_JK L_KK^{-1}
7.              send L_JK to all P that could own blocks in
                        row J or column J
8.      else
9.          Rec_{K,MyID} := Rec_{K,MyID} ∪ {L_IK}
10.         foreach L_JK ∈ Rec_{K,MyID} do
11.             if map[L_IJ] = MyID then
12.                 Find L_IJ
13.                 L_IJ := L_IJ - L_IK L_JK^T
14.                 nmod[L_IJ] := nmod[L_IJ] - 1
15.                 if (nmod[L_IJ] = 0) then
16.                     if I = J then /* diagonal block */
17.                         L_JJ := Factor(L_JJ)
18.                         send L_JJ to all P that could own blocks in
                                    column J
19.                     else if (Diag_{J,MyID} ≠ ∅) then
20.                         L_IJ := L_IJ L_JJ^{-1}
21.                         send L_IJ to all P that could own blocks in
                                    row I or column I
22.                     else
23.                         Wait_{J,MyID} := Wait_{J,MyID} ∪ {L_IJ}
```

FIG. 7. *Parallel block fan-out algorithm.*

the number of blocks in a column K that will be received by a processor p. Once $|Rec_{K,p}| = ToRec_{K,p}$, then the storage associated with blocks in column K is reclaimed.

We note that a small simplification has been made in steps 11 through 14 above. For all blocks L_{IJ}, I must be greater than J, a condition that is not necessarily true in the pseudo-code. The reader should assume that I is actually the larger of I and J, and similarly that J is the smaller of the two.

5.2. Block mapping for reduced communication. We now consider the issue of mapping blocks to processors. Our general approach is to restrict the set of processors that can own blocks modified by a particular block L_{IK} and thus decrease the number of processors the block must be sent to. The actual restriction is done by performing a *scatter decomposition* [9] of the blocks in the sparse matrix.

More precisely, assume that P processors are used for the factorization, and assume for the sake of simplicity that P is a perfect square ($P = s \times s$). Furthermore, assume that the processors are arranged in a 2-D grid configuration with the bottom left processor labeled $p_{0,0}$ and the upper right processor labeled $p_{s-1,s-1}$. To limit communication, a row of blocks is mapped to a row of processors. Similarly, a column of blocks is mapped to a column of processors. We choose round-robin distributions for both the rows and columns, where

$$map[L_{IJ}] = p_{I \bmod s, J \bmod s}.$$

Other distributions could be used. By performing the block mapping in this way, a block L_{IK} in the sparse factorization need only be sent to the row of processors that could own blocks in row I and the column of processors that could own blocks in column I. Every block in the matrix would thus be sent to a total of $2s = 2\sqrt{P}$ processors. Note that communication volume is independent of the block size with this mapping; every block in the matrix is simply sent to $2\sqrt{P}$ processors.

The scatter decomposition is appealing not only because it reduces communication volume, but also because it produces an extremely simple and regular communication pattern. All communication is done through multicasts along rows and columns of processors. This pattern is simple enough that one might reasonably expect parallel machines with 2-D grid interconnection networks to provide hardware multicast support for it eventually. In the absence of hardware support, an efficient software multicast scheme can be used. We will return to this issue later in this paper.

5.3. Enhancement: Domains. Before presenting performance results for the block fan-out approach, we first note that the method as described above produces more interprocessor communication than competing column approaches for small parallel machines. To understand the reason, consider a simple 2-D k-by-k grid problem. The corresponding factor matrix contains $O(k^2 \log k)$ nonzeros, and the parallel factorization of this matrix using a column approach generates $O(k^2 P)$ communication volume [14]. In the block approach, every nonzero in the matrix is sent to $O(\sqrt{P})$ processors, so the total communication volume grows as $O(\sqrt{P}k^2 \log k)$. The communication in the block approach grows less quickly in P, but it grows more quickly in k. The k term is more important for small P.

An important technique for reducing communication in column methods involves the use of *domains* [2], [4]. Domains are large sets of columns in the sparse matrix that are assigned en masse to a single processor. They are perhaps most easily understood in terms of the elimination tree of L. Recall that disjoint subtrees in the elimination tree are computationally independent, and consequently can be processed concurrently. By assigning the columns of an entire subtree (a domain) to a single processor, the communication that would have resulted had these columns been distributed among processors is avoided.

More precisely, by localizing all columns in a domain to a single processor, all modifications to these columns can be performed without the need for interprocessor communication. Furthermore, the modifications from all columns within a domain to all other entries in the matrix can be computed and aggregated within the owner processor, again with no communication. That processor can then send the aggregate modifications to the appropriate destinations. In a column approach, the aggregate modification is sent out on a columnwise basis. We refer the reader to [4] for more details.

Ashcraft suggested [2] that domains can be incorporated into a block approach as well. The basic approach is as follows. The nonzeros within a domain are stored as they would be in a column-oriented method. The domain factorization is then performed using a column method. The aggregate domain modification is computed columnwise as well. We use an extremely efficient left-looking supernode-supernode method for both. Once the aggregate modification has been computed, it is sent out in a blockwise fashion to the appropriate destination blocks.

Of course, the domains must be carefully assigned to processors so that processors do not sit idle, waiting for other processors to complete local domain computations. Geist and Ng [10] described an algorithm for assigning a small set of domains to each processor so that the amount of domain work assigned to each processor is evenly balanced. All results from this point on use the algorithm of Geist and Ng to produce domains.

With the introduction of domains, the parallel computation thus becomes a three-phase process. In the first phase, the processors factor the domains assigned to them and compute

the modifications from these domains to blocks outside the domains. In the second phase, the modifications are sent to the processors that own the corresponding destination blocks and are added into their destinations. Finally, the third phase performs the block factorization, where blocks are exchanged between processors. Note that these are only logical phases; no global synchronizations is necessary between the phases.

Consider the effect of domains on communication volume in a block method for a 2-D grid problem. We first note that the number of nonzeros not belonging to domains in the sparse matrix can be shown to grow as $O(k^2 \log P)$ [16]. Total communication volume for these nonzeros using a block approach is thus $O(\sqrt{P}k^2 \log P)$. The other component of communication volume when using domains is the cost of sending domain modifications to their destinations. The total size of all such modifications is $O(k^2)$, independent of P, so domain modification communication represents a lower-order term. Total communication for a 2-D grid problem is thus $O(\sqrt{P}k^2 \log P)$.

Note that domains produce the added benefit of reducing the number of small blocks in the matrix, and thus reducing related overheads. Recall that small supernodes are the main source of small blocks. In a sparse problem, most small supernodes lie toward the leaves of the elimination tree, where they are likely to be contained within domains.

6. Evaluation. This section evaluates the parallel block fan-out approach proposed in the previous section. The approach is evaluated in three different contexts. First, we look at performance on a small-scale multiprocessor. Then, we consider performance on moderately parallel machines (up to 64 processors), using our multiprocessor simulation model and using the DASH machine. Finally, we consider issues for more massively parallel machines.

6.1. Small parallel machines. The first performance numbers we present come from the Silicon Graphics SGI 4D/380 multiprocessor. Parallel speedups are shown in Fig. 8 for 1 through 8 processors. All speedups are computed relative to a left-looking supernode-supernode sequential code. The figure shows that the block fan-out method is indeed quite efficient for small machines. In fact, performance is slightly higher than that of our highly efficient panel-based parallel code [24]. Speedups on 8 processors are roughly 5.5-fold, corresponding to absolute performance levels of 40 to 50 double-precision MFLOPS. Speedups are less than linear in the number of processors for two simple reasons. First, the block method is slightly less efficient than a column method. We believe this accounts for a roughly 15% performance reduction. Second, the load is unevenly distributed among the processors. A simple calculation reveals that processors spend roughly 15% of the computation on average sitting idle. These two factors combine to give a relatively accurate performance prediction.

6.2. Moderately parallel machines. We now perform an evaluation of parallel performance of the block fan-out approach on machines with up to 64 processors, using the multiprocessor simulation model described earlier. We also discuss issues of communication volume.

6.2.1. Simulated performance. Figure 9 shows simulated processor utilization levels for between 4 and 64 simulated processors, using a block size of 24. It is clear from the figure that the block approach exhibits less than ideal behavior as the machine size is increased. On 64 processors, for example, utilization levels drop to roughly 40%. Further investigation reveals that the primary cause of the drop in performance is a progressive decline in the quality of the load balance. Figure 10 compares simulated performance for matrix BCSSTK15 with the best performance that could possibly be obtained with the same block distribution. The load balance performance bound is obtained by computing the time that would be required if there were no dependencies between blocks and if interprocessor communication were free.

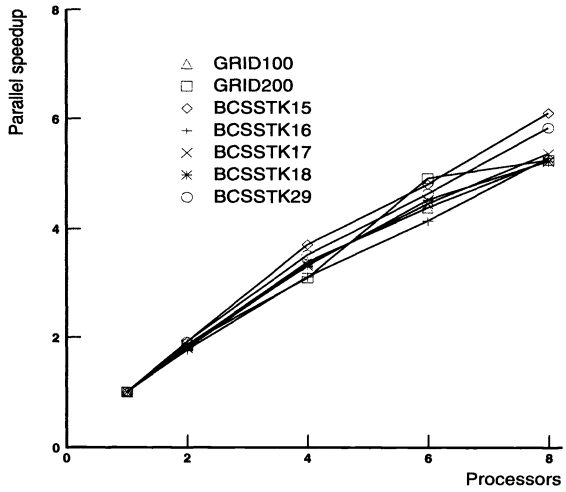


FIG. 8. *Parallel speedups for block fan-out method on* SGI 4D-280, $B = 24$.

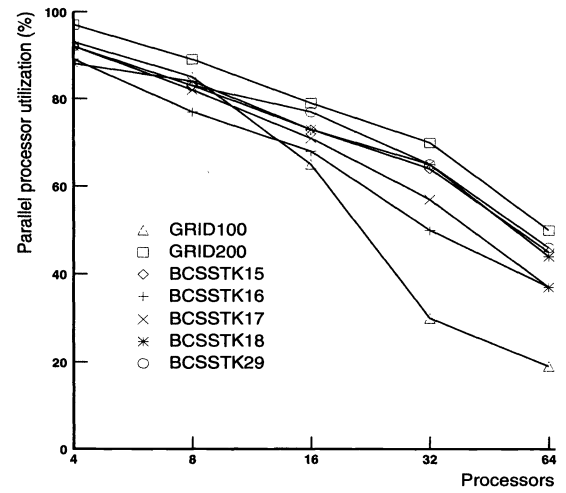


FIG. 9. *Simulated parallel efficiencies for block fan-out method,* $B = 24$.

The quality of the load distribution clearly depends on the method used to map blocks to processors. Recall that we use a very rigid mapping strategy, where block L_{IJ} is assigned to processor $p_{I \bmod s, J \bmod s}$. One possible explanation for the poor behavior of this strategy is that it does not adapt to the structure of the sparse matrix; it tries to impose a very regular structure on a matrix that is potentially comprised of a very irregular arrangement of nonzero blocks.

While the mismatch between the regular mapping and the irregular matrix structure certainly contributes to the poor load balance, it is our belief that a more important factor is the wide variability in task sizes. In particular, since a block is modified by some set of blocks to its left, blocks to the far right in the matrix generally require much more work than blocks

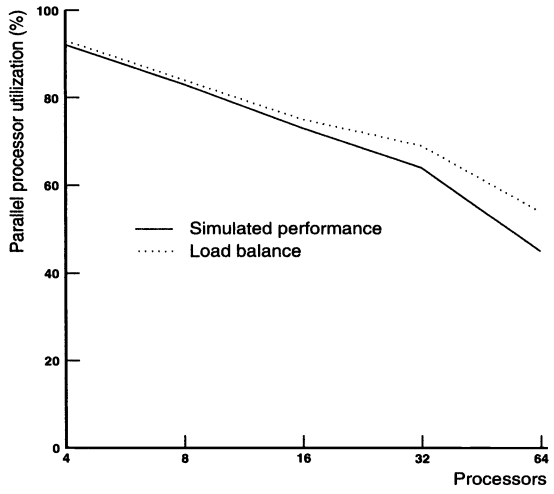


FIG. 10. *Simulated parallel performance for* BCSSTK15, *compared with load balance upper bound* ($B = 24$).

to the left (more accurately, blocks near the top of the elimination tree require more work than blocks near the leaves). Furthermore, since the matrix is lower-triangular, the number of blocks in a column decreases toward the right. The result is a small number of very important blocks in the bottom-right corner of the matrix.

To support our contention that the sparse structure of the matrix is less important than the more general task distribution problem, Fig. 11 compares the quality of the load balance obtained for matrix BCSSTK15 to the load balance obtained using the same mapping strategy for a dense matrix. The curves show the maximum obtainable processor utilization levels given the block mapping. The dense problem is chosen so as to perform roughly the same number of floating-point operations as the sparse problem.

Note that the load balance can be improved by moving to a smaller block size, thus creating more distributable blocks and making the block distribution problem easier. However, as discussed earlier, smaller blocks also increase block overheads. For the larger benchmark sparse matrices, decreasing the block size from $B = 24$ to $B = 16$ increases simulated parallel efficiencies for $P = 64$ from 40%–45% for $B = 24$ to 50%–55% for $B = 16$. A block size of less than 16 further improves the load balance, but achieves lower performance due to overhead issues.

The general conclusion to be drawn from these simulation results is simply that large machines require relatively large problems to achieve high processor utilization levels. In particular, the sparse matrices that we study here are too small to make good use of a 64 processor machine. Of course, it may be possible to significantly improve parallel load balance with a better mapping strategy. A more general function could be used to map columns of blocks to columns of processors, and to map rows of blocks to rows of processors. This matter will require further investigation.

6.2.2. Communication volume. So far, our analysis has assumed that parallel performance is governed by two costs: the costs of executing block operations on individual processors and the latencies of communicating blocks between processors. Another important, although less easily modeled, component of parallel performance is the total interprocessor communication volume. Communication volume will determine the amount of contention

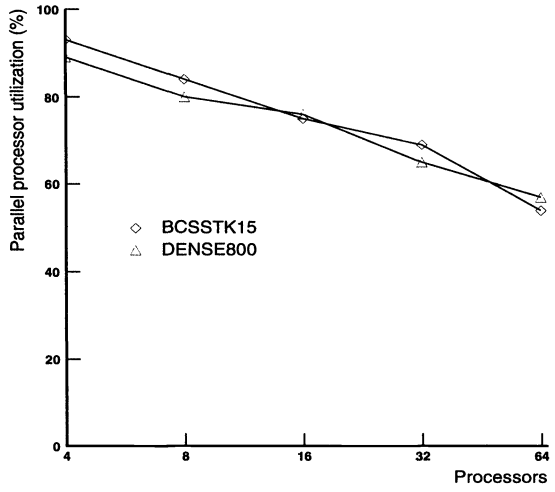


FIG. 11. *Parallel utilization upper bounds due to load balance for* BCSSTK15, *compared with load balance upper bounds for a dense problem* ($B = 24$). *Both problems perform roughly the same number of floating-point operations.*

that is seen on the interconnection network. Such contention can have severe performance consequences, and can in many cases govern the performance of the entire computation (see [26], for example).

Rather than trying to integrate these costs into our simple performance model, we instead look at interprocessor communication in a more qualitative way. To obtain a general idea of how much communication is performed, Fig. 12 compares total interprocessor communication volume with total floating-point operation counts for a variety of sparse matrices and machine sizes. This figure shows the average number of floating-point operations performed by a processor per floating-point value sent by that processor. Sustainable values will depend on the relative computation and communication bandwidths of the processor and the processor interconnect in the parallel machine. Current machines would most likely not have trouble supporting the roughly 40 to 1 ratio (corresponding to a 0.025 communication to computation ratio in the figure) seen for 16 processors on these matrices. The roughly 20 to 1 ratio (0.05 in the figure) on 64 processors, however, could prove troublesome.

To put these communication figures into better perspective, we now compare them to the communication volumes that would be seen with a column-oriented factorization method. Figure 13 shows relative communication volume, compared with a parallel column multifrontal method. Interestingly, the block approach does not always produce less communication than the column approach on 64 or fewer processors. While the growth rates, $O(P)$ for columns and $O(\sqrt{P}\log P)$ for blocks, favor the block approach, constants make these rates less relevant for small P. However, the trends clearly favor the block approach.

An interesting thing to note here is that relative communication is quite a bit higher for the two grid problems than for the other matrices. The reason is that the column multifrontal approach does very well communicationwise for sparse matrices whose elimination trees have few nodes toward the root and instead quickly branch out into several independent subtrees. The two grid problems have this property. The block approach derives no benefit from this property.

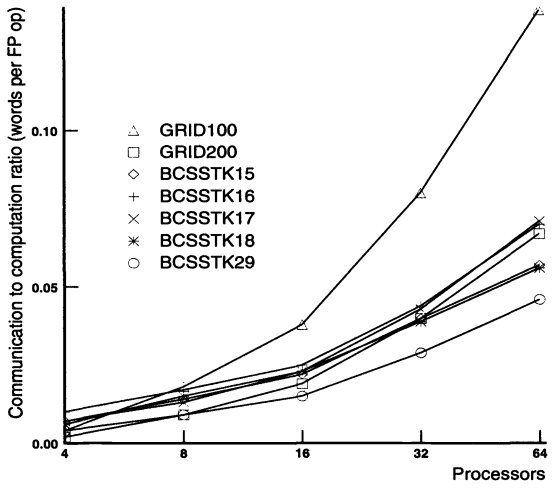


FIG. 12. *Communication versus computation for the block fan-out method.*

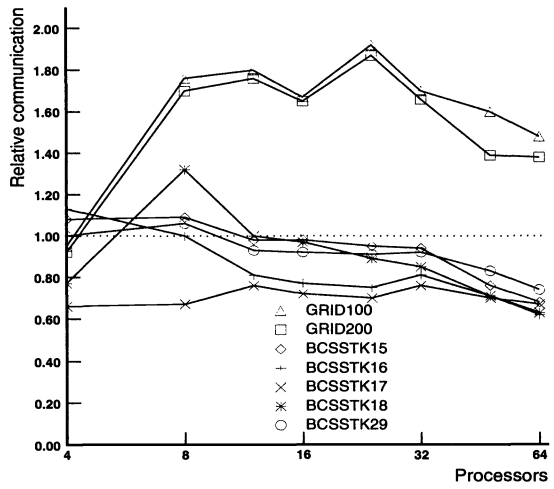


FIG. 13. *Communication volume of block approach, relative to a column-oriented parallel multifrontal approach.*

6.2.3. DASH performance. We now provide performance numbers from a block fan-out implementation on the Stanford DASH machine. Figure 14 shows achieved parallel speedups on 1 to 40 processors, again compared with a sequential left-looking supernode-supernode method. The sequential method obtains between 7 and 8 MFLOPS on these problems. The figure shows that speedups are relatively low, ranging from 12 to 18 on 40 processors. While these speedups are low, we should note two important items about the results. First, the absolute parallel performance levels of the DASH machine are still quite respectable. The 40 processor machine achieves roughly 100 double-precision MFLOPS. Second, we note that these performance numbers are roughly 10% to 40% higher than corresponding numbers from our panel-oriented parallel multifrontal implementation.

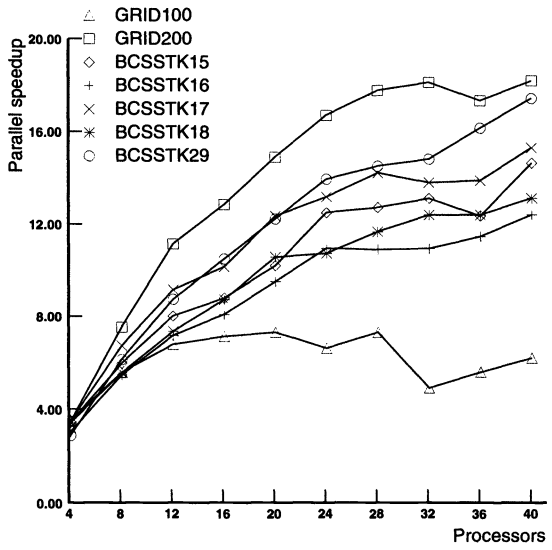


FIG. 14. *Parallel speedups for block fan-out method on the Stanford DASH machine, $B = 24$.*

One other thing to note about the speedups achieved on the DASH machine is that they are somewhat lower than those predicted by our simulation. We believe the main cause of this difference is our assumption in the simulation that communication latencies can be hidden from the processor. The DASH machine has limited ability to hide such latencies. Fortunately, current trends in parallel machines are toward machines that can hide latencies. We expect that these trends will lead to speedups on future parallel machines that are much closer to those predicted by our simulation.

6.2.4. Summary. To summarize this subsection, we note that our block fan-out approach provides good performance for moderately parallel machines, although parallel speedups are well below linear in the number of processors for the matrices we have considered. An important limiting factor is the relatively small size of the matrices and the relatively poor load balance that results from our rigid block distribution scheme. Regarding communication volumes, we find that the block approach produces comparable amounts of traffic to a column approach on 64 or fewer processors. Even so, we found that the block approach produces higher performance that a competing panel-oriented approach on the 8 processor SGI 4D/380 and the 40 processor Stanford DASH machine.

6.3. Massively parallel machines. Having concentrated on issues of efficiency on smaller machines in the first part of this section, we now turn our attention to three issues that will be important for very large parallel machines. First, we look at available concurrency in the problem. In other words, we look at how many processors can be productively used for a particular problem. Next we turn to the issue of per-processor storage requirements, and we consider how they grow as the number of processors and the problem size is increased. A common assumption for large parallel machines is that each processor will contain some constant amount of memory. Thus, it would be desirable for the amount of storage required per processor to remain constant. Finally, we consider interprocessor communication issues. Our discussions will use 2-D grid problems as examples. We should note that a similar analysis, using a different algorithm and different assumptions, appears in [15].

Before further discussing these issues, we should first explain our goals. The primary advantage of a block approach over a column approach for a massively parallel machine is that it allows more processors to cooperate for the same sparse problem. For a k-by-k 2-D grid problem, for example, the column approach allows $O(k)$ processors to participate (due to critical path constraints discussed earlier). By some measures, a block approach can use $O(k^2)$. Our goal is to determine whether the use of $O(k^2)$ processors is a realistic goal and to understand the difficulties that might be encountered in trying to reach this goal.

6.3.1. Concurrency. One important bound on the parallel performance of a computation is the length of the critical path. The length of this path for block-oriented sparse Cholesky factorization is proportional to the height of the elimination tree, assuming some constant block size. For a 2-D grid problem, the elimination tree has height $3k$. Thus, in the best case the $O(k^3)$ work of the entire factorization can be performed in $O(k)$ time. Consequently, at most $O(k^2)$ processors can be productively applied to this problem. This figure is consistent with our goals for the block approach.

6.3.2. Storage. We now look at the issue of how per-processor storage requirements grow as the size of the machine and the size of the problem is increased. We first note the obvious fact that the processor must store the portion of the matrix assigned to it. If the factorization is performed on P processors, and the problem being factored is a k-by-k grid problem, then each processor must store $O(k^2 \log k/P)$ nonzeros. Keeping per-processor storage requirements constant would thus require that the number of processors grow slightly faster than k^2. Since the critical path analysis showed that only $O(k^2)$ processors can be used productively for this problem, we must resign ourselves to a slow growth rate in per-processor storage.

Now consider the storage requirements of the auxiliary data structures that a processor must maintain. One important set of auxiliary data is the per-block information. An example is the count of how many times a block is modified. Another is the particular row and column of processors to which a particular block is sent when complete. This data adds a small constant to the size of each block, and consequently it represents a small constant factor increase in overall storage.

Another important set of auxiliary data is the columnwise data. One example is the arrival count information, which keeps track of how many blocks will arrive in a particular column. Since the number of columns in the matrix is k^2, this data structure would occupy $O(k^2)$ space per processor if every entry were kept. Fortunately, only $O(k^3/P)$ of these entries must be stored. The reason is as follows. If the factorization work is distributed evenly among the processors, then the work performed per processor is $O(k^3/P)$. Since a received block is only retained in a processor if it participates in some useful work, clearly the number of such retained blocks and thus the number of arrival counts that must be stored is also $O(k^3/P)$. We can keep a hash table, indexed by column number, of all nonzero arrival counts. When a block arrives, the corresponding arrival count is located and decremented. Note that not all blocks that arrive at a processor participate in a modification on that processor. If no arrival count is found for the block column of an arriving block, then the block is immediately discarded. Similar hash structures can be used for the other columnwise data structures.

Regarding per-processor storage growth rates, note that if P grows as k^2, then the per-processor matrix storage costs grow as $O(\log k)$ while the arrival count storage costs grows as $O(k^3/P) = O(k)$. Fortunately, the $O(k)$ term has a very small constant in front of it, so this term will not be particularly constraining for practical P. However, asymptotic per-processor storage requirements will grow with P.

6.3.3. Communication. A crucial determinant of performance on massively parallel machines is the bandwidth of the processor interconnection network. To obtain a rough feel for whether the bandwidth demands of the block fan-out method are sustainable as the machine size increases, we look at these demands in relation to two common upper bounds on available communication bandwidth, in a manner similar to that used by Schreiber in [26]. The two upper bounds are based on bisection bandwidth and total available point-to-point bandwidth in the multiprocessor. We consider a 2-D mesh machine organization, which is in some sense a worst case since it offers lower connectivity than most alternative organizations.

A bisection bandwidth bound is obtained by breaking some set of point-to-point interconnection links in the parallel machine to divide it into two halves. Clearly, all communication between processors in different halves must travel on one of the links that is split. The bisection bandwidth bound simply states that the parallel runtime is at least as large as the time that would be required for these bisection links to transmit all messages that cross the bisector.

In the case of the block fan-out method applied to a 2-D grid problem, recall that $O(k^2 \log P)$ messages are sent, and each is multicast to $O(\sqrt{P})$ processors (a row and column of processors). Figure 15 shows an example mesh of processors, an example bisector, and the communication pattern that can be used to multicast a message. For any simple bisector, a multicast to a row and column of processors crosses that bisector twice. Thus, total traffic across the bisector is $O(k^2 \log P)$. This traffic must travel on one of $O(\sqrt{P})$ communication links in the bisector, and this communication occurs in the $O(k^3/P)$ time required for the factorization. If we assume that communication is evenly distributed among the bisector links, then communication per bisector link per unit time is $O(k^2 \log P/\sqrt{P}(k^3/P)) = O(\sqrt{P} \log P/k)$. If P grows as k^2, communication per link per unit time is thus $O(\log P)$. Since the amount of data that can travel on a single link per unit time is constant, this growth rate represents a small problem. The number of processors P must grow slightly slower than k^2 to keep message volume per link constant.

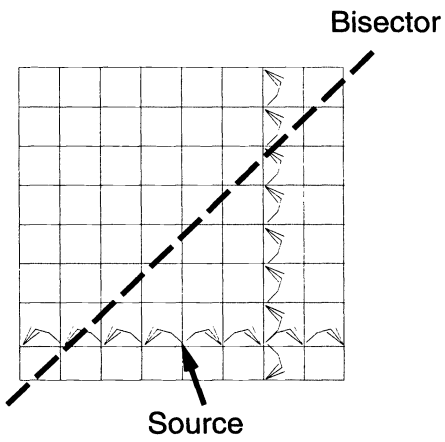


FIG. 15. *Communication pattern for row/column multicast.*

Another common communication-based bound on parallel performance is the total amount of traffic that appears on any link in the machine, expressed as a fraction of the total number of links in the machine. For our example, there are $O(k^2 \log P)$ multicasts, each of which traverses $O(\sqrt{P})$ links. The number of links in the machine is $O(P)$, and again this communication occurs in $O(k^3/P)$ time. Thus, global traffic per link per time unit is

$O(\sqrt{P}k^2 \log P/(k^3/P)(P))$ or $O(\sqrt{P} \log P/k)$. If P is $O(k^2)$, we obtain $O(\log P)$ traffic per link, which is identical to the bisector traffic.

We should note that the preceding arguments have said nothing about *achieved* performance. Demonstrating that certain performance levels can actually be achieved would require a detailed analysis of the structure of the sparse matrix, the way in which the factorization tasks are mapped to processors, and the order in which these tasks are handled by their owners. This would certainly be a daunting task. This discussion has simply shown that the approach is not constrained away from achieving high performance by any of the most common performance bounds.

6.4. Summary. To summarize our evaluation, we have found that the block fan-out method is quite appealing across a range of parallel machines. Overheads are low enough that the method is quite effective for small parallel machines. It is also effective for moderately parallel machines, although performance is somewhat limited by the quality of the computational load balance. For massively parallel machines, we found that the approach is not perfect. Per-processor storage requirements grow with the number of processors. Bisection bandwidth considerations also limit the number of processors to below ideal. However, these constraints are mild enough that the block fan-out approach appears to be quite practical even for very large P.

7. Future work. While this paper has explored several practical issues related to parallel block-oriented factorization, it also has brought up a number of questions that will require further investigation. Foremost among these is the question of whether the load balance could be significantly improved. We are currently investigating more flexible block mapping strategies.

Another interesting question concerns the choice of partitions for the 2-D decomposition. Recall that our partitions are chosen to contain sets of contiguous columns from within the same supernode. Ashcraft has shown [2] that by choosing columns that are not necessarily contiguous, it is often possible to divide the sparse matrix into fewer, denser blocks. While our results indicate that the simpler approach is quite adequate, we are currently looking into the question of how large the benefit of a more sophisticated approach may be.

We also hope to compare the block fan-out approach we have proposed here with the block multifrontal approach proposed by Ashcraft [2]. One thing we are certain of is that the block fan-out method is much less complex. So far, we have not discovered any significant advantages to a multifrontal approach, but the issue requires further study. We also hope to investigate a block analogue of the fan-in method.

Once a matrix A has been factored into the form $A = LL^T$, the next step is typically the solution of several triangular systems $Ly = b$, where b is given. An issue that we have left unaddressed in this paper is the efficiency of this triangular solve computation when L is represented as a set of blocks. Our belief is that this computation will be more efficient than the corresponding computation for a column representation, but further investigation will be required to fully answer this question.

8. Conclusions. It is becoming increasingly clear that column approaches are inappropriate for sparse Cholesky factorization on large parallel machines. One thing that has been much less clear is whether the alternative, a 2-D matrix decomposition, is truly practical. This paper has proposed and evaluated a parallel block algorithm that is quite practical. The primary virtues of our approach are: (1) it uses an extremely simple decomposition strategy in which the matrix is divided using global horizontal and vertical partitions; (2) it is straightforward to implement; (3) it provides good per-processor performance, since it performs the vast majority of its work within dense matrix-matrix multiplication operations; (4) it is efficient on

moderately parallel machines, providing performance that is comparable to that of efficient column (and panel) methods; and (5) it shows good promise for large parallel machines.

Acknowledgments. We would like to thank Rob Schreiber and Sid Chatterjee for their discussions on block-oriented factorization.

REFERENCES

[1] P. R. AMESTOY AND I. S. DUFF, *Vectorization of a multiprocessor multifrontal code*, Internat. J. Supercomputer Appl., 3 (1989), pp. 41–59.

[2] C. C. ASHCRAFT, *The Domain/Segment Partition for the Factorization of Sparse Symmetric Positive Definite Matrices*, Boeing Computer Services Tech. report ECA-TR-148, Nov. 1990.

[3] ———, *The fan-both family of column-based distributed Cholesky factorization algorithms*, in Graph Theory and Sparse Matrix Computation, IMA Volumes in Mathematics and its Applications 56, Springer-Verlag, New York, 1993.

[4] 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 Dept., Yale University, New Haven, CT, 1990.

[5] 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.

[6] 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.

[7] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[8] I. S. DUFF AND J. K. REID, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[9] G. FOX ET AL., *Solving Problems on Concurrent Processors: Volume 1—General Techniques and Regular Problems*, Prentice–Hall, Englewood Cliffs, NJ, 1988.

[10] G. A. GEIST AND E. NG, *Task scheduling for parallel sparse Cholesky factorization*, Internat. J. Parallel Programming, 18 (1989), pp. 291–314.

[11] A. GEORGE, M. HEATH, J. LIU, AND E. NG, *Sparse Cholesky factorization on a local-memory multiprocessor*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 327–340.

[12] ———, *Solution of sparse positive definite systems on a hypercube*, J. Comput. Appl. Math., 27 (1989), pp. 129–156.

[13] A. GEORGE AND J. LIU, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[14] A. GEORGE, J. LIU, AND E. NG, *Communication results for parallel sparse Cholesky factorization on a hypercube*, Parallel Comput., 10 (1989), pp. 287–298.

[15] J. GILBERT AND R. SCHREIBER, *Highly parallel sparse Cholesky factorization*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1151–1172.

[16] L. HURLBERT AND E. ZMIJEWSKI, *Limiting communication in parallel sparse Cholesky factorization*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 1184–1197.

[17] D. LENOSKI, J. LAUDON, K. GHARACHORLOO, W. D. WEBER, A. GUPTA, J. HENNESSY, M. HOROWITZ, AND M. LAM, *The Stanford DASH multiprocessor*, Computer, 23 (1992), pp. 63–79.

[18] J. LIU, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.

[19] ———, *The role of elimination trees in sparse factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 134–172.

[20] R. LUCAS, *Solving Planar Systems of Equations on Distributed-Memory Multiprocessors*, Ph.D. thesis, Stanford University, 1988.

[21] D. MOORE, *Dense Patch-Oriented Matrix Factorization on a Hypercube Multiprocessor*, Tech. report 87–809, Cornell University, Ithaca, NY, 1987.

[22] E. ROTHBERG, *Exploiting the Memory Hierarchy in Sequential and Parallel Sparse Cholesky Factorization*, Ph.D. thesis, Stanford University, Jan. 1993.

[23] E. ROTHBERG AND A. GUPTA, *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, 1991.

[24] ———, *Techniques for improving the performance of sparse matrix factorization on multiprocessor workstations*, Supercomputing '90, Nov. 1990, pp. 232–243.

[25] R. SCHREIBER, *A new implementation of sparse Gaussian elimination*, ACM Trans. Math. Software, 8 (1982), pp. 256–276.

[26] R. SCHREIBER, *Scalability of sparse direct solvers*, in Graph Theory and Sparse Matrix Computation, IMA Volumes in Mathematics and its Applications 56, Springer-Verlag, New York, 1993.

[27] R. VAN DE GEIJN, *Massively Parallel LINPACK Benchmark on the Intel Touchstone Delta and* iPSC/860 *Systems*, Tech. report CS-91-28, University of Texas at Austin, Aug. 1991.

[28] S. VENUGOPAL AND V. K. NAIK, *Effects of partitioning and scheduling sparse matrix factorization on communication and load balance*, Supercomputing '91, Nov. 1991.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1440–1451, November 1994

010

A TWO-PHASE ALGORITHM FOR THE CHEBYSHEV SOLUTION OF COMPLEX LINEAR EQUATIONS*

DIRK P. LAURIE† AND LUCAS M. VENTER‡

Abstract. An overdetermined system of complex linear equations is solved in the maximum norm by an iterative algorithm in which each iteration has two phases. Phase 1 concentrates on the combinatorial problem of identifying the active set of equations where the residual magnitude attains the maximum, and Phase 2 concentrates on the numerical problem of accurately solving the equations on the active set. Phase 1 is aimed at building up the set of active equations by means of an interior point method applied to the equivalent real semi-infinite linear program. Phase 2 uses the quadratically convergent Newton method to find the solution to the equality constrained problem on the current active set. A theoretical convergence proof has not been found, but numerical examples show rapid convergence.

Key words. complex linear equations, Chebyshev solution, complex approximation, semi-infinite linear programming

AMS subject classifications. 65D15, 41A50

1. Introduction.

Let $\Psi(\mathbf{A}, \mathbf{b})$ denote the Chebyshev distance between a complex m-vector $\mathbf{b}$ and the column space of a complex $m \times n$ matrix $\mathbf{A}$, i.e.,

$$\Psi(\mathbf{A}, \mathbf{b}) = \min_{\mathbf{z} \in \mathbb{C}^n} \|\mathbf{b} - \mathbf{A}\mathbf{z}\|_\infty, \tag{1}$$

where the Chebyshev norm of a complex n-vector $\mathbf{z}$ is defined as

$$\|\mathbf{z}\|_\infty = \max_{j=1,\ldots,n} |z_j|. \tag{2}$$

A vector $\mathbf{z}$ for which the minimum in (1) is attained is called a Chebyshev solution to the overdetermined system $\mathbf{A}\mathbf{z} = \mathbf{b}$.

An excellent survey of various approaches to the problem of finding the Chebyshev solution appears in Watson [7]. We will therefore refer only to those ideas that are relevant to the present paper.

It is standard practice to rewrite the problem as a convex program

$$\min_{\mathbf{z} \in \mathbb{C}^n} \varepsilon \quad \text{subject to } |r_k|^2 \le \varepsilon^2, \qquad k = 1, \ldots, m, \quad \text{where } \mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{z} \tag{3}$$

in the $2n + 1$ real variables ε, $\operatorname{Re} z_j$, $\operatorname{Im} z_j$, $j = 1, \ldots, n$. Watson's approach is to approximate the convex program by a sequence of quadratic programs. The objective function appearing in each quadratic program depends on the current approximation to the solution, an estimate of the optimal ε, and a plausible active set I. Watson's algorithm is quadratically convergent once I has been identified.

The convex programming approach leads to the following characterization theorem given by Watson [7].

THEOREM 1.1. *Let $\mathbf{z} \in \mathbb{C}^n$ and $\varepsilon = \|\mathbf{r}\|_\infty$, where $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{z}$. Then $\mathbf{z}$ is a Chebyshev solution to the overdetermined system $\mathbf{A}\mathbf{z} = \mathbf{b}$, and $\varepsilon = \Psi(\mathbf{A}, \mathbf{b})$, if and only if there exists a set I containing p indices where $p \le 2n + 1$, and a real m-vector $\mathbf{w}$ such that:*

*Received by the editors January 24, 1990; accepted for publication (in revised form) September 23, 1993.

†Department of Mathematics, Potchefstroom University for Christian Higher Education, P.O. Box 1174, Vanderbijlpark 1900, South Africa (WSKDPL@puknet.puk.ac.za).

‡Department of Mathematics, Potchefstroom University for Christian Higher Education, Potchefstroom 2520, South Africa (WSKLMV@puknet.puk.ac.za).

(a) $|r_k| = \varepsilon$, $k \in I$,

(b) $w_k > 0$ *for* $k \in I$ *while* $w_k = 0$ *for* $k \notin I$, *and*

(c) $\mathbf{A}^*\mathbf{W}\mathbf{r} = \mathbf{0}$ *with* $\mathbf{W} = \operatorname{diag}(\mathbf{w})$. (*We use* $\mathbf{A}^*$ *to denote the Hermitian transpose of* $\mathbf{A}$.) *The set* I *is called an* active set *and the vector* $\mathbf{w}$ *a* dual solution. *In addition, if all* $n \times n$ *submatrices of* $\mathbf{A}$ *are nonsingular (the discrete Haar condition), the Chebyshev solution* $\mathbf{z}$ *(but not necessarily the active set* I *or the dual solution* $\mathbf{w}$*) is unique. In that case,* $p \geq n+1$. *In this article we assume that the discrete Haar condition holds.*

The problem may also be written as a semi-infinite linear program (SILP)

$$\min_{\mathbf{z}\in\mathbb{C}^n} \varepsilon \quad \text{subject to } \operatorname{Re}(e^{-i\theta_k} r_k) \leq \varepsilon, \quad 0 \leq \theta_k < 2\pi, \quad k = 1, \ldots, m. \tag{4}$$

The problem can be approximated by a linear program (LP) [8] after discretizing the continuous variables θ_k. Streit [6] has shown that the special structure of the problem can be exploited when solving the sequence of LPs that arises when the discretization interval is progressively halved. He also shows that the solution obtained in this way produces a value of ε that underestimates $\Psi(\mathbf{A}, \mathbf{b})$ by a factor not exceeding $\sec \frac{\pi}{p}$, where p is the number of equispaced points used in the discretization. Thus his algorithm is only linearly convergent as p is increased (the error in ε is reduced by a factor of 4 at each halving), but it is nevertheless very efficient when moderate accuracy (two to three digits) is required.

In his concluding remarks, Watson [7] observes, "An alternative would be to work with a guessed active set (an approximation to I) at each step, and solve an equality constrained problem; however, this would require the incorporation of a satisfactory active set strategy, and so far this has not been tried." The algorithm to be described here is based on this approach.

At the start of each iteration, we assume only that a current approximate solution $\mathbf{z}$ is known. Each iteration consist of two phases.

Phase 1. Obtain a plausible active set (or working set) with the aid of an interior point search method applied to the SILP starting from the current approximate solution $\mathbf{z}$. In the process, the approximate solution is continually improved. The algorithm in Phase 1 is related to a steepest-descent method and therefore converges at best linearly.

Phase 2. Attempt to solve the equality constrained problem on the current working set; if successful, try to modify the working set to find a solution that satisfies the conditions of the characterization theorem. If an optimal solution is not obtained, return to Phase 1. The algorithm used to solve the equations is Newton's method using the current $\mathbf{z}$ as starting point, which converges quadratically (or not at all.)

Our algorithm at certain points makes use of line searches, i.e., given a point $\mathbf{z}$ and a direction $\mathbf{s}$, the best point on the line $\mathbf{z} + \alpha\mathbf{s}$ where α is a nonnegative real number, is defined as the point where $f(\alpha) = \|\mathbf{b} - \mathbf{A}(\mathbf{z} + \alpha\mathbf{s})\|_\infty^2$ is a minimum. The graph of f is a piecewise parabolic convex curve, and an exact line search is therefore easy to implement by moving from one intersection to the next.

The paper is organized as follows. We start with two sections giving the motivation and mathematical derivation of the two phases of the algorithm. Additional computational details of interest to the implementation of the algorithm are given in §4, and numerical examples are given in §6. We do not have a convergence proof; in §5 we discuss the gaps in the theoretical analysis of the algorithm. The algorithm nevertheless performs well in practice; the numerical examples include cases where Watson's implementation of his algorithm failed, as well as accurate solutions to large practical problems that were solved to three significant digits by Sherrill and Streit [3].

2. Finding a plausible working set. In Phase 1 of the algorithm we think of the problem as an SILP. Each constraint of the SILP (4) can be "numbered" by a pair (k, d) where k identifies

a row of the matrix and $d = e^{i\theta}$ is a number on the unit circle $\mathcal{K} \subset \mathbb{C}$. When the constraint is active, $d = \operatorname{sign} r_k$. The component of a vector $\mathbf{q} \in \mathbb{C}^m$ in the direction of the constraint (k, d) is defined as the number $\operatorname{Re}(\bar{d}q_k)$.

We use a modified form of an interior point search method for LPs developed by Snyman [4]. Starting from the current point $\mathbf{z}$ having residual $\mathbf{r} = \mathbf{b} - \mathbf{Az}$ and discrepancy $\varepsilon = \|\mathbf{r}\|_\infty$, we set $\mathbf{y}^0 = \mathbf{z}$ and perform an inner iteration during which new points $\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^p$ with $p \leq 2n$ are generated. At the same time, constraints (i_j, d_j), $j = 0, \ldots, p$ of the SILP are identified. The indices $i_0, i_1, \ldots, i_p$ will later define the working set to be used by Phase 2 of the algorithm. The first constraint (i_0, d_0) is chosen among the maximal components of the residual at the start of Phase 1. We define the jth inner residual by $\mathbf{q}^j = \mathbf{b} - \mathbf{Ay}^j$ and the jth discrepancy by $\varepsilon_j = \|\mathbf{q}^j\|_\infty$.

The crucial idea (adapted from Snyman) is that at each stage the inner residual $\mathbf{q}^l$ should have the same component δ_l in the direction of each previously identified constraint, i.e.,

$$\operatorname{Re}(\bar{d}_j q^l_{i_j}) = \delta_j, \qquad j = 0, \ldots, l. \tag{5}$$

Initially $\delta_0 = \varepsilon_0$.

The inner iteration may stop for one of three reasons:

1. $p = 2n$ has been reached. Note that the indices $i_0, i_1, \ldots, i_p$ need not all be different.
2. The inner iteration starts to diverge, i.e., $\varepsilon_{p+1} > \varepsilon_p$. In this case we replace $\mathbf{y}^p$ by the best value along the line $\mathbf{y}^p + \alpha(\mathbf{y}^{p+1} - \mathbf{y}^p)$.
3. A zero division is encountered while trying to calculate $\mathbf{y}^{p+1}$.

At termination, following Snyman [4], we replace $\mathbf{z}$ by the best value along the line $\mathbf{y}^0 + \alpha(\mathbf{y}^p - \mathbf{y}^0)$. Whatever the reason for termination, we say that *Phase* 1 *has been successful* if at least $n + 1$ distinct indices have been identified. If Phase 1 is unsuccessful repeat Phase 1, otherwise proceed to Phase 2.

In the inner iteration, each new point is formed by the formula

$$\mathbf{y}^l = \mathbf{y}^{l-1} + \sigma_l \mathbf{s}^l, \tag{6}$$

where in the words of Snyman the search direction $\mathbf{s}^l$ "points away equally" from each constraint already identified as potentially active, and σ_l is a suitably chosen step size. More precisely, we require that

$$\operatorname{Re} \bar{d}_j \mathbf{A}[i_j]\mathbf{s}^l = 1, \qquad j = 0, \ldots, l-1, \tag{7}$$

where $\mathbf{A}[i]$ denotes the ith row of $\mathbf{A}$. The system of equations (7) contains l real equations in the $2n$ real variables $\operatorname{Re} s_i$, $\operatorname{Im} s_i$, $i = 1, \ldots, n$. The overdetermination is resolved by using the pseudoinverse of the coefficient matrix, which gives the solution of minimal norm. In other words, $\mathbf{s}^l$ is the vector of steepest descent among all vectors that "point away equally" from each identified constraint.

Before choosing the step size σ_l, we determine the first constraint encountered in the direction $\mathbf{s}_l$. It is convenient to work with the "residual direction" $\mathbf{u} = \mathbf{As}^l$, in terms of which the residual $\mathbf{q}(\alpha)$ at the point $\mathbf{y}^{l-1} + \alpha\mathbf{s}^l$ can be written as

$$\mathbf{q}(\alpha) = \mathbf{q}^{l-1} - \alpha\mathbf{u},$$

where $\mathbf{q}^l = \mathbf{b} - \mathbf{Ay}^l$. For $k = 1, \ldots, m$ we find the smallest positive value β_k for which

$$|q^l_k(\beta_k)| = \varepsilon_{l-1}. \tag{8}$$

The next identified constraint (i_l, d_l) is formed such that β_{i_l} is smallest, and $d_l = \operatorname{sign}(q^l_{i_l})$.

We now choose the new point $\mathbf{y}^l$ so that (5) is satisfied. Substituting $\mathbf{q}^l = \mathbf{q}^{l-1} - \sigma_l \mathbf{u}$, and using (7), we find that cases $j = 0, \ldots, l-1$ of the equations (5) are satisfied when

$$\delta_l = \delta_{l-1} - \sigma_l. \tag{9}$$

The case $j = l$ of (5) can then be solved to give

$$\sigma_l = \frac{\delta_{l-1} - \operatorname{Re}(\overline{d}_l q^l_{k_l})}{1 - \operatorname{Re}(\overline{d}_l u_{k_l})}. \tag{10}$$

The cost of one complete Phase 1 iteration, when the direction $\mathbf{s}^l$ is computed as described in §4.2, is $\mathcal{O}(n^3)$.

3. Modifying the current working set. If the active set I is known, the problem becomes much simpler, since the characterization theorem tells us that indices outside I may be ignored. The problem can then be formulated as an equality constrained problem

$$\min_{\mathbf{z}\in\mathbb{C}^n, \mathbf{d}\in\mathcal{K}^p} \varepsilon \quad \text{subject to } \mathbf{A}[I]\mathbf{z} + \varepsilon\mathbf{d} = \mathbf{b}[I]. \tag{11}$$

Conditions (a) and (c) of the characterization theorem, together with a suitable normalization of the vector $\mathbf{w}$, yield the equations

$$\mathbf{e}^\top \mathbf{w} = 1, \tag{12}$$

$$\mathbf{A}[I]\mathbf{z} + \varepsilon\mathbf{d} = \mathbf{b}[I], \tag{13}$$

$$\mathbf{A}[I]^* \mathbf{W}\mathbf{d} = \mathbf{0}, \tag{14}$$

where $\mathbf{W} = \operatorname{diag}(\mathbf{w})$ and $\mathbf{e}$ is a vector of ones. Condition (b) of the characterization theorem is used to check whether a solution to (12)–(14) is indeed optimal.

On the assumption that a solution $\mathbf{z}$ to (12)–(14) with its accompanying dual solution $\mathbf{w}$ can be computed relatively easily (for instance, by Newton's method), one can devise an exchange algorithm along the following lines.

1. Given a current working set I^k with p indices where $n+1 \le p \le 2n+1$, obtain primal and dual solutions $\mathbf{z}^k$ and $\mathbf{w}^k$ with discrepancy ε_k to the equations (12)–(14) with $I = I^k$. Let $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{z}^k$.

2. Any residual $\mathbf{r}_j$ with $j \notin I^k$ whose modulus exceeds ε_k provides an index j that is a candidate for inclusion in the next working set I^{k+1}. Any negative element of the dual solution $\mathbf{w}^k$ provides an index that is a candidate for exclusion from the working set. If there are no candidates for inclusion or exclusion, the current solution is optimal. If not, the working set is modified by including or excluding indices (or doing both), and the computation is repeated using the next working set.

There are several reasons why this scenario might fail.

1. The set I_k might be such that equations (12)–(14) have no solution.

2. The initial values for the iterative solution might not be good enough for the numerical method to converge.

3. The solution to equations (12)–(14) may have a larger residual than the vector used as starting value.

4. The inclusion or exclusion of indices might lead to a working set with too few or too many elements.

5. The index set may cycle.

We have therefore not attempted to implement a strategy based only on exchanging indices in the working set. Instead, we make a small number of explorations around the working set supplied by Phase 1, and if an optimal solution is not found, we return to Phase 1.

The logic runs as follows. If the solution to equations (12)–(14) produces a $\mathbf{z}^k$ with lower discrepancy than that of the previous value, but that is not optimal, we first try to augment the working set by including one index (the index corresponding to the largest residual). If there are no candidates for inclusion, we reduce the working set by removing all the candidates for exclusion. If the modified working set still has between $n+1$ and $2n+1$ elements, we compute a new solution to equations (12)–(14) and repeat the process. The number of retries is limited. (In our numerical examples, we only allow three retries.)

If the solution to equations (12)–(14) produces a $\mathbf{z}^k$ with higher discrepancy than that of the previous value $\mathbf{z}^{k-1}$, we replace $\mathbf{z}^k$ by the best point on the line $\mathbf{z}^{k-1}+\alpha(\mathbf{z}^k-\mathbf{z}^{k-1})$ before returning to Phase 1.

4. Computational considerations. Most of the computations involve standard algorithms of linear algebra, e.g., solution of least-squares systems. In this section we point out a number of simplifications that reduce the amount of work considerably.

We found that the least-squares solution to the original problem provides a suitable initial value for $\mathbf{z}$.

We sometimes need a real vector $\mathbf{x}$ of length $2n$ formed from the real and imaginary parts of a complex vector $\mathbf{w}$ of length n. In some computer implementations, the formation of $\mathbf{x}$ involves only a pointer conversion with no actual movement of data. We use the notation $\mathbf{x}=\underline{\mathbf{w}}$ for the real vector formed in this way. Note that $\operatorname{Re}(\mathbf{v}^*\mathbf{w})=\underline{\mathbf{v}}^{\top}\underline{\mathbf{w}}$.

4.1. Repeated index. It may happen during Phase 1 that a certain constraint is encountered twice in the inner iteration, say at indices i_k and i_l with $k<l$. In this case, the numbers d_k and d_l may be nearly equal, which causes computational problems when calculating the new search direction $\mathbf{s}$. We therefore replace the lth equation in the system (7) by

$$\operatorname{Im}(\overline{d_k}+\overline{d_l})\mathbf{A}[i_l]\mathbf{s}^l=0, \tag{15}$$

which is equivalent to subtracting the lth equation from the kth equation in the system (7); since $|d_k|=|d_l|$ the vectors d_k-d_l and d_k+d_l in the complex plane are orthogonal, and hence

$$\operatorname{Re}(\overline{d_k}+\overline{d_k})\mathbf{A}[i_l]\mathbf{s}^l=0 \quad\Rightarrow\quad \operatorname{Im}(\overline{d_k}+\overline{d_l})\mathbf{A}[i_l]\mathbf{s}^l=0.$$

It may even happen that the constraint is encountered a third time. In that case, the new constraint is linearly dependent on the two previously found constraints and the inner iteration must be aborted.

4.2. Updating the search direction. Denote the system of (real) equations describing $\underline{\mathbf{s}}$ during stage j (i.e., equation (7)) by

$$\mathbf{P}\underline{\mathbf{s}}^j=\mathbf{t}. \tag{16}$$

Since we require the minimum norm solution, $\underline{\mathbf{s}}^j$ is given by $\underline{\mathbf{s}}=\mathbf{P}^+\mathbf{t}$, where $\mathbf{P}^+$ is the pseudoinverse of $\mathbf{P}$.

It is more economical, however, to update $\underline{\mathbf{s}}$ with the aid of an orthogonal factorization of $\mathbf{P}$. The next system (stage $j+1$) is of the form

$$\begin{bmatrix}\mathbf{P}\\ \mathbf{p}^{\top}\end{bmatrix}\underline{\mathbf{s}}^{j+1}=\begin{bmatrix}\mathbf{t}\\ \tau\end{bmatrix}. \tag{17}$$

Since the column space of $\mathbf{P}^+$ is the row space of $\mathbf{P}$, $\underline{\mathbf{s}}_j^{\top}$ is a linear combination of the rows of $\mathbf{P}$, and $\underline{\mathbf{s}}_{j+1}^{\top}$ is a linear combination of the rows of $\mathbf{P}$ and $\mathbf{p}$. If we set

$$\underline{\mathbf{s}}^{j+1}=\underline{\mathbf{s}}^j+\mathbf{h},$$

then $\mathbf{h}$ is also a linear combination of the rows of $\mathbf{P}$ and $\mathbf{p}$. Substituting into equation (17), we find that

$$\mathbf{P}\underline{\mathbf{s}}^j + \mathbf{P}\mathbf{h} = \mathbf{t}, \tag{18}$$

$$\mathbf{p}^\top \underline{\mathbf{s}}^j + \mathbf{p}^\top \mathbf{h} = \tau. \tag{19}$$

From equations (16) and (18) it follows that $\mathbf{P}\mathbf{h} = \mathbf{0}$, which implies that $\mathbf{h} \in \mathcal{R}(\mathbf{P}^\perp)$. Since $\mathbf{h}$ is at the same time a linear combination of the rows of $\mathbf{P}$ and $\mathbf{p}$, this means that $\mathbf{h}$ must be a multiple of the projection $\mathbf{q}$ of $\mathbf{p}$ on $\mathcal{R}(\mathbf{P}^\perp)$.

To find $\mathbf{q}$, we assume that $\mathbf{P} = \mathbf{L}\mathbf{Q}$ with $\mathbf{L}$ lower-triangular and $\mathbf{Q}$ orthogonal. We then find the LQ factorization $\mathbf{L}'\mathbf{Q}'$ of $\mathbf{P}'$ from the LQ factorization of $\mathbf{P}$ with one step of the modified Gram–Schmidt algorithm (see, e.g., [2], p. 151): $\mathbf{q}^\top$ is simply the last row of $\mathbf{Q}'$. Finally, from equation (19) we have that

$$\mathbf{h} = \frac{\tau - \mathbf{p}^\top \underline{\mathbf{s}}^j}{\mathbf{p}^\top \mathbf{q}} \mathbf{q}.$$

As a final simplification, we note that $\mathbf{p}^\top \mathbf{q} = \ell_{jj}$, the last element on the diagonal of $\mathbf{L}'$.

4.3. The Phase 2 iteration. The straightforward approach is to consider (12)–(14) as $2p + 2n + 1$ equations in the $2p + 2n + 1$ real variables ε, $\mathbf{w}$, $\boldsymbol{\theta}$, $\operatorname{Re}\mathbf{z}$, and $\operatorname{Im}\mathbf{z}$. It is possible to reduce the number of equations substantially by a suitable substitution, however. Define the vector $\mathbf{q}$ by

$$q_k = w_k d_k,$$

and let $\mathbf{U}$ be a basis for the left nullspace $\mathcal{N}(\mathbf{A}[I]^*)$ of $\mathbf{A}[I]$. By (14), $\mathbf{q} \in \mathcal{N}(\mathbf{A}[I]^*)$, and therefore there exists $\boldsymbol{\nu} \in \mathbb{C}^{p-n}$ such that $\mathbf{q} = \mathbf{U}\boldsymbol{\nu}$. We can write $\mathbf{w}$ in terms of $\boldsymbol{\nu}$ and $\mathbf{d}$ by putting

$$\mathbf{w} = \mathbf{D}^*\mathbf{Q}\boldsymbol{\nu}, \tag{20}$$

where $\mathbf{D} = \operatorname{diag}(d_k)$. Since $\mathbf{w}$ is real, the additional equation

$$\operatorname{Im}\mathbf{D}^*\mathbf{Q}\boldsymbol{\nu} = 0 \tag{21}$$

must be satisfied. On the other hand, one can eliminate $\mathbf{z}$ entirely by multiplying (13) by $\mathbf{U}^*$ to obtain

$$\varepsilon\mathbf{Q}^*\mathbf{d} - \mathbf{c} = 0, \tag{22}$$

where

$$\mathbf{c} = \mathbf{Q}^*\mathbf{b}.$$

We end up with $2(p-n) + p + 1$ real equations in the $2(p-n) + p + 1$ real unknowns ε, $\boldsymbol{\theta}$, $\operatorname{Re} s$, $\operatorname{Im} s$, namely

$$0 = f_1(\varepsilon, \boldsymbol{\theta}, \boldsymbol{\nu}) := \operatorname{Re}(\mathbf{d}^*\mathbf{Q}\boldsymbol{\nu}) - 1, \tag{23}$$

$$\mathbf{0} = \mathbf{f}_2(\varepsilon, \boldsymbol{\theta}, \boldsymbol{\nu}) := \operatorname{Im}\mathbf{D}^*\mathbf{Q}\boldsymbol{\nu}, \tag{24}$$

$$\mathbf{0} = \mathbf{f}_3(\varepsilon, \boldsymbol{\theta}, \boldsymbol{\nu}) := \varepsilon\mathbf{Q}^*\mathbf{d} - \mathbf{c}, \tag{25}$$

where (23) is obtained by substituting (20) and (22) into (12).

It is tempting to eliminate $\boldsymbol{\theta}$ also, since $w_k \exp(i\theta_k)$ is simply the polar form of q_k. This procedure is inadvisable since it is necessary to know whether w_k is positive or negative, and that information is lost when we only know the product $q_k = w_k \exp(i\theta_k)$. We actually tried this approach, and it is unreliable in practice as well as in theory.

To solve equations (23)–(25) by Newton's method, we need the Jacobian

$$\mathbf{J} = \frac{\partial[f_1, \mathbf{f}_2, \operatorname{Re}\mathbf{f}_3, \operatorname{Im}\mathbf{f}_3]}{\partial[\varepsilon, \boldsymbol{\theta}, \operatorname{Re}\boldsymbol{\nu}, \operatorname{Im}\boldsymbol{\nu}]}.$$

The required derivatives are easy to find when one remembers that

$$\mathbf{D}^*\mathbf{Q}\boldsymbol{\nu} = \operatorname{diag}(\mathbf{Q}\boldsymbol{\nu})\bar{\mathbf{d}},$$
$$\frac{\partial[\bar{\mathbf{d}}]}{\partial[\boldsymbol{\theta}]} = -i\bar{\mathbf{D}}.$$

We obtain

$$\mathbf{J} = \begin{bmatrix} 0 & \mathbf{0} & \operatorname{Re}(\mathbf{d}^*\mathbf{Q}) & -\operatorname{Im}(\mathbf{d}^*\mathbf{Q}) \\ \mathbf{0} & -\operatorname{diag}(\operatorname{Re}(\mathbf{D}^*\mathbf{Q}\boldsymbol{\nu})) & \operatorname{Im}(\mathbf{D}^*\mathbf{Q}) & \operatorname{Re}(\mathbf{D}^*\mathbf{Q}) \\ \operatorname{Re}(\mathbf{Q}^*\mathbf{d}) & \varepsilon\operatorname{Re}(i\mathbf{Q}^*\mathbf{D}) & \mathbf{0} & \mathbf{0} \\ \operatorname{Im}(\mathbf{Q}^*\mathbf{d}) & \varepsilon\operatorname{Im}(i\mathbf{Q}^*\mathbf{D}) & \mathbf{0} & \mathbf{0} \end{bmatrix}.$$

We can approximate $\mathbf{Q}^*\mathbf{d}$ by $\frac{\mathbf{c}}{\varepsilon}$, since the approximation becomes exact at the solution.

Normally in Newton's method one would solve for the increments $\delta\varepsilon$, $\delta\boldsymbol{\theta}$, and $\delta\boldsymbol{\nu}$, but in this case the right-hand side becomes simpler if we solve for $\hat{\boldsymbol{\nu}} = \boldsymbol{\nu} + \delta\boldsymbol{\nu}$ instead of $\delta\boldsymbol{\nu}$. The system of equations is

$$\begin{bmatrix} 0 & \mathbf{0} & \underline{\mathbf{c}}^\top \\ \mathbf{0} & -\mathbf{W} & \underline{\mathbf{E}}^\top \\ \frac{1}{\varepsilon}\underline{\mathbf{c}} & -\varepsilon\mathbf{E} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\varepsilon \\ \delta\boldsymbol{\theta} \\ \underline{\hat{\boldsymbol{\nu}}} \end{bmatrix} = \begin{bmatrix} 1 \\ \mathbf{0} \\ -\underline{\mathbf{f}}_3 \end{bmatrix}, \tag{26}$$

where

$$\mathbf{W} = \operatorname{diag}(\operatorname{Re}(\mathbf{D}^*\mathbf{Q}\boldsymbol{\nu})), \tag{27}$$

$$\mathbf{E} = \begin{bmatrix} \operatorname{Im}(\mathbf{Q}^*\mathbf{D}) \\ -\operatorname{Re}(\mathbf{Q}^*\mathbf{D}) \end{bmatrix}. \tag{28}$$

It is possible to scale rows and columns so that the matrix becomes symmetric, in which case one may use a method designed for indefinite symmetric systems [2], but the present matrix can be handled more simply. Exchange the first row with the row containing the largest element of $\mathbf{c}$ and perform one step of Gaussian elimination. Discard the first row and column and we are left with a system of the form

$$\begin{bmatrix} -\mathbf{W} & \mathbf{E}^\top \\ \mathbf{F} & \mathbf{M} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\theta} \\ \underline{\hat{\boldsymbol{\nu}}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \hat{\mathbf{f}} \end{bmatrix}, \tag{29}$$

where $\mathbf{M}$ has only one nonzero row. Putting $\mathbf{G} = \mathbf{W}^{-1}\mathbf{E}^\top$, a simple scaling of the rows of $\mathbf{E}$, we obtain

$$\underline{\hat{\boldsymbol{\nu}}} = (\mathbf{M} - \mathbf{F}\mathbf{G})^{-1}\hat{\mathbf{f}}, \tag{30}$$

$$\delta\boldsymbol{\theta} = -\mathbf{G}\underline{\hat{\boldsymbol{\nu}}}. \tag{31}$$

Instead of using the previously discarded equation to obtain $\delta\varepsilon$, we prefer to compute the new value $\hat{\epsilon}$ directly from (22). There are several ways to do so, all equivalent when equations (23)–(25) are satisfied, but they show quite different behaviour when the current approximate solution is still inaccurate. For example, the obvious equation $\varepsilon = \|\mathbf{c}\|/\|\mathbf{U}^*\mathbf{d}\|$ gives misleadingly small values for ε even when the solution is quite bad. We have obtained the best results with the formula

$$\varepsilon = \operatorname{Re} \frac{\mathbf{c}^*\mathbf{c}}{\mathbf{c}^*\mathbf{U}^*\mathbf{d}}, \tag{32}$$

obtained by premultiplying (22) by $\mathbf{c}^*$.

When Phase 2 is complete, we find the corresponding $\mathbf{z}$ by solving (13) for $\mathbf{z}$ and ε, taking $\mathbf{d}$ as known. The least-squares solution is used when $p > n + 1$, even though the overdetermined system is compatible when Phase 2 is successful.

The cost of one Phase 2 iteration is $\mathcal{O}((p-n)^3) + \mathcal{O}(pn)$. Since $n + 1 \le p \le 2n + 1$, the cost is no worse than $\mathcal{O}(n^3)$.

4.4. Easy special cases of Phase 2. The two extreme cases $p = n+1$ and $p = 2n+1$ can be handled easily under the assumption that all elements of $\mathbf{w}$ are positive. When $p = n + 1$, $\mathcal{N}(\mathbf{A}[I]^*)$ is one-dimensional. Therefore $\mathbf{q}$ is a multiple of the single column of $\mathbf{U}$, and we can read off $\mathbf{w}$ and $\mathbf{d}$ directly from the polar form of $\mathbf{U}$, normalizing $\mathbf{w}$ afterwards.

When $p = 2n + 1$, (13) may be considered as $4n + 2$ equations in the $4n + 2$ unknowns ε, $\operatorname{Re} z$, $\operatorname{Im} z$, θ. The system is not underdetermined and there is no need to bring in $\mathbf{w}$ at all. Multiply the system by $\mathbf{D}^*$ and extract the real part to obtain

$$\operatorname{Re}(\mathbf{D}^*(\mathbf{A}[I]\mathbf{z} - \mathbf{b}[I])) + \varepsilon\mathbf{e} = \mathbf{0}. \tag{33}$$

Newton's method for (13) reduces to the alternation of two steps:

1. Given $\mathbf{d}$, solve the square linear system (33) for ε, $\operatorname{Re}\mathbf{z}$, $\operatorname{Im}\mathbf{z}$.
2. Compute the new $\mathbf{d}$ from

$$\mathbf{d} = \operatorname{sign}(\mathbf{b}[I] - \mathbf{A}[I]\mathbf{z}). \tag{34}$$

4.5. Starting values for Phase 2. At the end of Phase 1, we have available a working set I and a current solution $\mathbf{z}$. From the current residual $\mathbf{r}[I] = \mathbf{b}[I] - \mathbf{A}[I]\mathbf{z}$ one can obtain starting values for ε and $\mathbf{d}$ by

$$\varepsilon = \frac{1}{p}\sum_{k=1}^{p} |r_{i_k}|, \tag{35}$$

$$\mathbf{d} = \operatorname{sign}\mathbf{r}[I]. \tag{36}$$

Alternatively, one could use $\mathbf{r} = \mathbf{U}\mathbf{c}$ (the least-squares residual) in (35)–(36). This choice has the desirable property that $\mathbf{U}\mathbf{c}$ is known to be in $\mathcal{N}(\mathbf{A}[I]^*)$, which is normally not the case for the residual at the end of Phase 1. On the other hand, no advantage is taken of the fact that the Phase 1 residual is likely to be closer to the optimal residual. Our numerical results indicate no clear preference for either choice of $\mathbf{r}$.

For the Phase 2 iteration, we need starting values for ν, and these are most easily obtained by using (12) and (14) to compute starting values for $\mathbf{w}$. We obtain the $2n + 1$ equations

$$\begin{bmatrix} \operatorname{Re}(\mathbf{A}[I]^*\mathbf{D}) \\ \operatorname{Im}(\mathbf{A}[I]^*\mathbf{D}) \\ \mathbf{e}^{\mathsf{T}} \end{bmatrix} \mathbf{w} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \end{bmatrix} \tag{37}$$

in the p unknowns $\mathbf{w}$. Since $p \le 2n + 1$, these equations should be solved in the least-squares sense.

5. Theoretical discussion. The algorithm described above contains several heuristics aimed at efficiency. The price paid is that a convergence proof cannot be obtained. In this section we discuss the theoretically inadequate aspects of the algorithm and indicate how they could be met at the cost of greatly increased computation time.

The convergence of the algorithm depends only on the convergence of Phase 1. If Phase 1 does not converge, there is no reason to suppose that the initial values supplied to Phase 2 will be good enough for convergence even if Phase 1 happens to provide the correct active set. Phase 2 is provided only to obtain rapid convergence when possible; the explorations around the current working set are not intended to exhaust the possibilities.

There are several reasons why Phase 1 may not converge. It may abort each time with too few indices in the working set, or it may repeatedly come up with the same working set. These reasons can be traced back to the decision to retain identified SILP constraints as long as possible. By the time that the pth index is added to the working set, the signs of the previously identified residuals may have changed to such an extent that equation (7) no longer guarantees that the direction $\mathbf{s}_l$ is a descent direction. This problem could be corrected by using the current signs in (7), but then the efficient computation of $\mathbf{s}_l$ as indicated in §4.2 would no longer be possible; a new system would need to be solved from scratch, thereby increasing the computing time for Phase 1 from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^4)$. We have tried this approach in practice (in fact, the Matlab coding is much shorter than that for the updating of $\mathbf{s}_l$) without observing any great benefit in terms of a reduction of the total number of Phase 1 iterations.

In view of our claim that our algorithm is an efficient way of calculating the Chebyshev solution to an overdetermined complex system to high accuracy, we are unwilling to incur a severe speed penalty for the sake of being able to provide a theoretical proof.

6. Numerical results. The algorithm was programmed in Matlab 3.5 on an IBM-compatible microcomputer, which uses a 54-bit mantissa (approximately 16 decimal digits). We present three sets of examples. First a tiny problem, small enough to be worked by hand, is shown in detail so that the workings of the algorithm may be understood. Then the performance of the algorithm on two examples given by Watson [7] is shown. Finally we demonstrate the effectiveness of the algorithm on two large examples from a practical problem given by Sherrill and Streit [3]. In each case, our algorithm terminated with primal and dual solutions $\mathbf{z}$ and $\mathbf{w}$ that satisfy the optimality conditions of the characterization theorem. The optimal values of ε quoted are believed to be correct to the number of digits given.

Example 6.1. *Approximate* w^2 *by* $z_1 w - z_2$ *in the three points* 1, $\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i$, *and* i. This gives

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{2}}i \\ 1 & i \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 1 \\ i \\ -1 \end{bmatrix}.$$

The least-squares solution is $z = (-0.8918i, 1.0765 + 1.0765i)$. We choose this as an initial value for z and calculate

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{z} = \begin{bmatrix} -0.0765 - 0.1847i \\ 0.3694i \\ 0.0765 - 0.1847i \end{bmatrix}.$$

The maximum value for $|\mathbf{r}_i|$ is $\varepsilon = 0.3694$, which occurs at r_2, and hence we set $I = \{2\}$. We are now ready to start the inner iteration of Phase 1.

Solving for $\mathbf{s}$ from the equation $\mathrm{Re}\,(\bar{d}_0 u_{k_0}) = 1$, we find that $\mathbf{s}^\top = [0.5i; 0.3536 + 0.3536i]$. This enables us to calculate $\beta = 0.1834$ from equation (8); the next active index is

identified as $\ell = 1$. From r_1 we find that $d_1 = \operatorname{sign} r_1 = -0.3827 - 0.9239i$. The final step is to calculate $\sigma = 0.0881$ from equation (10) and to update $\mathbf{z}$, $\mathbf{r}$, and δ to

$$\mathbf{z} = \begin{bmatrix} -0.8478i \\ 1.1077 + 1.1077i \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} -0.1077 - 0.2599i \\ 0.2813i \\ 0.1077 - 0.2599i \end{bmatrix}, \quad \delta = 0.2813.$$

This completes one inner iteration of Phase 1.

The next iteration identifies the third row of $\mathbf{A}$ as active, and Phase 1 terminates with $I = \{1, 2, 3\}$, $\mathbf{z}^\top = [-0.8478i;\ 1.1077 + 1.1077i]$, and $\varepsilon = 0.2813$. Phase 2 computes the correct solution in one step since I contains $n + 1$ indices.

At the start of Phase 2, we update ε to the current discrepancy; the updated value is $\varepsilon = 0.2813$. The solution to equations (23)–(25) is

$$\nu = 0.4027 - 0.4512i, \quad \delta\boldsymbol{\theta} = \begin{bmatrix} -0.2192E-15 \\ -0.1190E-15 \\ -0.2084E-15 \end{bmatrix}.$$

Notice that Phase 2 did not change the value of ε or even that of $\mathbf{z}$ significantly. However, we can now calculate $\mathbf{w}^\top = [0.2599;\ 0.4802;\ 0.2599]$, which shows that the solution is optimal by the characterisation theorem.

Example 6.2. *Approximate the function* $f(w) = 1/\sqrt{1 + (w+1)^2}$ *by* $\sum_{j=1}^{n} z_j(1 + w)^{1-j}$, *using* 25 *points on* $\{w : w = iy,\ -20 \le y \le 20\}$.

This example was considered by Ellacott and Williams in [1]. For $n = 7$, it proved troublesome to the algorithm of Watson [7], who reports that the library subroutine he uses to solve the quadratic programming subproblems failed in that case. Our algorithm experienced no great difficulty; the results up to $n = 14$ are given in Table 1.

Example 6.3. *Approximate the function* $f(w) = 1/\sqrt{1 + (w+1)^2}$ *by* $\sum_{j=1}^{n} z_j/(w + j)$, *using* 25 *points on* $\{w : w = iy,\ -20 \le y \le 20\}$.

This is the same as Example 6.2, with a different approximating function. The problem rapidly becomes ill conditioned for large values of n, but this did not influence the performance of the algorithm. The results are given in Table 1.

TABLE 1
Optimal discrepancy and number of outer iterations for Examples 6.2 *and* 6.3.

	Example 6.2			Example 6.3		
		Phase			Phase	
n	ε	1	2	ε	1	2
3	0.01627327178246	4	4	0.01738515249031	1	1
4	0.00955282778207	1	3	0.00633755462554	1	1
5	0.00041701172641	2	4	0.00198911364538	2	3
6	0.00033491131379	1	2	0.00058661398810	2	5
7	0.00000602297966	3	8	0.00037706725013	1	1
8	0.00000590606483	1	2	0.00036767407418	3	5
9	0.00000006261614	1	3	0.00029829945416	2	3
10	0.00000006017302	1	2	0.00026372633210	1	1
11	0.00000000049419	1	1	0.00015470487937	1	1
12	0.00000000038075	1	1	0.00013213127646	1	1
13	0.00000000000283	2	4	0.00007103828118	2	3
14	0.00000000000176	1	1	0.00005726328318	1	1

Example 6.4. *Let* $\mathbf{x}$ *be a given vector of length* $n+1$, *and* $\boldsymbol{\theta}$ *be* m *equally spaced real numbers with* $\theta_1 = K_0$ *and* $\theta_m = 4\pi - K_0$. *The vector* $\mathbf{b}$ *is defined by* $b_p = f(\theta_p, x_{n+1})$ *and the matrix* $\mathbf{A}$ *by* $a_{p,q} = b_p - f(\theta_p, x_q)$ *where* $f(\theta, x) = \exp(-i\theta x)$.

This example, together with its relevance to real-life applications, is given by Sherrill and Streit [3]. The number K_0 is given by the formula

$$K_0 = \frac{2}{D}\cos^{-1}\frac{2}{(r-s)^{\frac{1}{N-1}} + (r+s)^{\frac{1}{N-1}}},$$

where $r = t$ dB (where as usual t dB $= 10^{t/20}$), $s = \sqrt{(r^2-1)}$, and t, D, and N are design parameters. Three different cases are reported there. In each case $\mathbf{x} = \{D, 2D, 3D, \ldots, ND\}$ with some elements removed. The problems are solved in [3] by Streit's algorithm [5] described in §1 with 32 equispaced values in the discretization of the SILP (4). Actually, they also employ a strategy (thoroughly explained in [3]) that reduces computing time substantially at the cost of increasing the factor in the error bound to $\sec^2 \frac{\pi}{p}$, which for $p = 32$ has the value 1.00484.

1. $t = 30$, $N = 25$, $D = 0.5$, $m = 128$; $\mathbf{x}$ has elements 2 and 4 removed, i.e., $n = 22$. The optimal value given in [3] is -26.86 dB $= 0.0454$, which implies that the correct optimal value does not exceed 0.04561. Our algorithm found the optimal solution with $\varepsilon = 0.04552462718747$ using 8 Phase 1 iterations and 14 Phase 2 iterations. Four of the Phase 2 iterations were unsuccessful.

2. $t = 30$, $N = 50$, $D = 0.5$, $m = 128$; $\mathbf{x}$ has elements 7, 22, 40, 43, and 50 removed, i.e., $n = 44$. The solution in [3] is -25.51 dB $= 0.0530$, yielding $\varepsilon < 0.0533$; we obtained $\varepsilon = 0.05315420963977$ in 4 Phase 1 and 11 Phase 2 iterations. All Phase 2 iterations were successful.

3. $t = 30$, $N = 25$, $D = 0.5$, $m = 128$; $\mathbf{x}$ has elements 11 and 14 removed, i.e., $n = 22$. The numerical value of the solution is not quoted in [3], but a graph given there suggests ε near -19 dB. Our algorithm required 8 Phase 1 and 14 Phase 2 iterations and obtained $\varepsilon = 0.11449564056985$. Three of the Phase 2 iterations were unsuccessful.

7. Conclusion. The algorithm presented here is fast and accurate. In numerical examples with up to 128 equations in up to 44 unknowns, it never failed to produce an accurate solution in a reasonably small number of iterations. Its main drawback is that it is theoretically possible that the algorithm may fail to converge. The most obvious method of addressing this objection will increase the cost of the algorithm from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^4)$.

Acknowledgment. This paper owes a great deal to the comments of two anonymous referees. The authors felt at the time that the points raised would cause much extra work (which was true), but the increase in the readability of the paper and the efficiency of the algorithm made the effort well worthwhile.

REFERENCES

[1] S. ELLACOTT AND J. WILLIAMS, *Linear Chebyshev approximation in the complex plane using Lawson's algorithm*, Math. Comp., 30 (1976), pp. 35–44.

[2] G. H. GOLUB AND C. F. V. LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.

[3] M. S. SHERRILL AND R. L. STREIT, In situ *optimal reshading of arrays with failed elements*, IEEE J. Ocean. Eng., OE-12 (1987), pp. 155–162.

[4] J. A. SNYMAN, *An interior feasible direction method with constrained projections for linear programming*, Comput. Math. Appl., 20 (1990), pp. 43–54.

[5] R. L. STREIT, *An algorithm for the solution of complex linear equations in the* l_∞ *norm with constraints on the unknowns*, ACM Trans. Math. Software, 11 (1985), pp. 242–249.

[6] R. L. STREIT, *Solution of systems of complex linear equations in the l_∞ norm with constraints on the unknowns*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 132–149.

[7] G. A. WATSON, *A method for the Chebyshev solution of an overdetermined system of complex linear equations*, IMA J. Numer. Anal., 8 (1988), pp. 461–471.

[8] P. WOLFE, *An outline of nonlinear programming*, in Mathematics of the Decision Sciences, Part I, G. Dantzig and J. A. F. Veinott, eds., American Mathematical Society, Providence, RI, 1968.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1452–1466, November 1994

011

PRECONDITIONING FOR DOMAIN DECOMPOSITION THROUGH FUNCTION APPROXIMATION*

MO MU† AND JOHN R. RICE‡

Abstract. A new approach was presented in [M. Mu, *SIAM J. Sci. Comput.*, 16 (1995), to appear] for constructing preconditioners through a function approximation for the domain decomposition-based preconditioned conjugate gradient method. This work extends the approach to more general cases where grids may be nonuniform, elliptic operators may have variable coefficients (but are separable and self-adjoint), and geometric domains may be nonrectangular. The theory of expressing the Schur complement as a function of a simple interface matrix is established. The approximation to this complicated function by a simple function is discussed and the corresponding error bound is given. Preconditioning a nonrectangular domain problem is done by first reducing it to a rectangular domain problem, and then applying the theory developed here for the rectangular domain case. Accurate error bounds are given by using the results in [*SIAM J. Numer. Anal.*, 28 (1991), pp. 378–391] for typical domains, such as L-, T-, and C-shaped ones. Numerical results are also reported to illustrate the efficiency of this approach.

Key words. domain decomposition, preconditioners, preconditioned conjugate gradient methods, iterative methods, partial differential equations, parallel computation

AMS subject classifications. 65N55, 65F10, 65Y05

1. Introduction.

A new approach to constructing preconditioners for the domain decomposition-based preconditioned conjugate gradient (PCG) method through a function approximation is proposed in [11] by making the observation that the interface *capacitance matrix* S, or the *Schur complement*, can be viewed as a matrix function

$$S = f(T), \tag{1.1}$$

where $f(t)$ is a complicated function and T is a simple interface matrix. The approach is to find a simple approximation $r(t)$ to $f(t)$, such that, with $q(t) \equiv f(t)/r(t)$,

$$\text{(a)} \qquad R \equiv r(T) \text{ is easily invertible;}$$

$$\text{(b)} \qquad \frac{\max_i |q(t_i)|}{\min_i |q(t_i)|} \sim 1 \quad \text{or} \quad \{q(t_i)\} \text{ are clustered,} \tag{1.2}$$

where $\{t_i\} = \sigma(T)$ is the spectrum of T. The convergence rate of the PCG method is governed by the quantity

$$\kappa(R^{-1}S) \equiv \frac{\lambda_{\max}(R^{-1}S)}{\lambda_{\min}(R^{-1}S)} = \frac{\max_i |q(t_i)|}{\min_i |q(t_i)|}, \tag{1.3}$$

where $\lambda_{\max}$ is the maximum eigenvalue and $\lambda_{\min}$ is the minimum eigenvalue, or by the spectrum distribution of the preconditioned matrix $R^{-1}S$ given by

$$\sigma(R^{-1}S) = \{q(t_i)\}. \tag{1.4}$$

*Received by the editors June 1, 1993; accepted for publication (in revised form) October 5, 1993.

†Department of Mathematics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong (mamu@usthk.bitnet). This author's work was supported in part by National Science Foundation grant CCR-8619817 and Hong Kong RGC grant DAG93/94.SC10.

‡Computer Sciences Department, Purdue University, West Lafayette, Indiana 47907 (rice@cs.purdue.edu). This author's work was supported in part by Air Force Office of Scientific Research grants 88-0243 and F49620-92-J-0069 and by the Strategic Defense Initiative through Army Research Office contract DAAL03-86-K-0106.

It is easily seen that the conditions (1.2) imply that R is a good preconditioner for the PCG method.

We begin with a review of previous work and relate it to this approach. Relation (1.1) is essential to this approach. This relation is established in [2] for the standard model problem of a Poisson equation on a rectangle with Dirichlet condition discretized by the *five-point-star* stencil with a uniform square grid, where $f(t)$ is shown to be a rational function in terms of the Chebyshev polynomials and $T = 2I + K$ where K is the discrete one-dimensional Laplacian on the interface. An equivalent expression in terms of the eigendecomposition of K is obtained in [5]. An extension to the variable coefficient case in which the elliptic operator is separable and self-adjoint is implied in [1], where $f(t)$ is still a rational function but in terms of other orthogonal polynomials that play a role analogous to that of the Chebyshev polynomials in the constant coefficient case. These orthogonal polynomials are defined by the three-term recurrence relation in terms of the discrete one-dimensional operator in the direction perpendicular to the interface. The roots of such a polynomial are the eigenvalues of the corresponding tridiagonal matrix from the theory of orthogonal polynomials. An equivalent expression is also used in [14] for the reciprocal $1/f(t)$ and is expanded into a sum of partial fractions to approximate the inverse of the Schur complement for a nonrectangular domain, which is referred to as the *rational approximation to the Schur complement of a nonrectangular domain* because of the rational function $f(t)$. The rational expressions in terms of these orthogonal polynomials developed in [1] and [2] are also used in [9], by being expanded into a sum of partial fractions, to devise a fast direct solver in a parallel setting for the original two-dimensional discrete operator. All the above assume a uniform square grid.

The rational expression theory for the Schur complement is extended in [11] to the nonuniform grid case in which the grid is nonuniform on the interface and uniform in the other direction, and the elliptic operator has constant coefficients. In this case, (1.1) is modified to the form

$$S = \Theta^{1/2} f(\Theta^{-1/2} T \Theta^{-1/2}) \Theta^{1/2}, \tag{1.5}$$

where Θ is a diagonal scaling operator corresponding to the spacings of the nonuniform grid on the interface, $f(t)$ is, within a constant factor, the same rational function as in the uniform case, and T corresponds to a certain discrete one-dimensional operator on the interface.

For a nonrectangular domain Ω and mixed boundary value conditions, [14] shows a spectral equivalence in the sense that there exist constants c_1 and c_2, such that, for any vector $\mathbf{v}$ of a proper dimension,

$$c_1(S_0\mathbf{v}, \mathbf{v}) \leq (S\mathbf{v}, \mathbf{v}) \leq c_2(S_0\mathbf{v}, \mathbf{v}), \tag{1.6}$$

where S_0 is the Schur complement on the same interface Γ but corresponds to the Dirichlet condition for the rectangular region embedded in the original domain Ω by shifts of Γ up to $\partial\Omega$. Here $(\cdot, \cdot)$ is the inner product and ∂ is the boundary symbol. Similar results for Dirichlet boundary conditions are also obtained in [3], [4], and [6]. The relation (1.6) implies that S_0 can be taken as a preconditioner for S. This reduces, in principle, a non-Dirichlet problem on a nonrectangular region to a Dirichlet problem on a rectangular region. The efficiency of using S_0 depends on the constants c_1 and c_2 being close to 1.

For a rectangular region and the case of a constant coefficient and separable elliptic operator and a uniform grid on the interface, the Schur complement can be efficiently inverted using (1.1) and the fast Fourier transform (FFT) applied to the eigendecomposition of T. This makes the PCG method a direct solver in this case. Other well-known preconditioners can be related to (1.1) by being viewed as approximating $f(t)$ with square-root-like functions because $f(t)$ behaves like $t^{1/2}$ near the smallest eigenvalues of T. The matrix T is usually called K in

this literature and they define the $K^{1/2}$ family of preconditioners; see, for example, [3], [5], [8], and [10]. However, these preconditioners depend either on using the FFT for T or on using two-dimensional subdomain solvers, which makes their extension to general cases inefficient and ineffective. The approach proposed in [11] provides a general framework for constructing preconditions for S using (1.1) and a function approximation to $f(t)$. Various approximations yield different preconditioners. The $K^{1/2}$ family of preconditioners can, of course, fall into this category. But they are not generally efficient because of the appearance of a square root in the corresponding approximations. One of the basic principles in our approach is to have a simple form for the $r(t)$ such that the generated matrix R is easily invertible in terms of T. In [11] we illustrate how to construct such simple functions, such as a product of two first-degree interpolating rational functions, or a linear interpolation. By utilizing the special properties of $f(t)$ we can satisfy the conditions (1.2). Examples are given in [11] and [12] showing that this approach is very simple, effective, and efficient. Independently, a similar idea is used in [14] by constructing another m-term sum of partial fractions as an approximation to the n-term sum expression for $1/f(t)$. A theoretical analysis is given showing that under certain conditions on the eigenvalues of two one-dimensional discrete operators in the x and y directions, the approximation error is of the order of $O(1/n^{\tau})$, for any $\tau > 0$, if $m = O(\log n)$. However, for a real application it is not clear when those conditions can be satisfied, what number needs to be used for m, and so on. No numerical experiments are reported on the actual performance of the approach in [14].

The purpose of this paper is to extend our approach to general cases. Section 2 is devoted to establishing the relation (1.5) for a very general case on a rectangular region where the elliptic operator is separable and self-adjoint with variable coefficients, and the grid may be nonuniform in both directions. An expression for efficiently evaluating $f(t)$ is presented and we note in our approach that $f(t)$ only needs to be evaluated at a few interpolating points. The function approximation and preconditioner construction are discussed in §3 with numerical results showing the efficiency and effectiveness of the approach. Section 4 considers the extension to a nonrectangular domain. An accurate estimate for the convergence rate of the PCG method is given with the help of the results in [6] from the relationship between overlapping and nonoverlapping for domain decomposition and from the dependence of the convergence rate on geometry for the Schwarz overlapping method. Finally, we give conclusions in §5.

A typical application of this work is for solving an elliptic boundary value problem on a concave domain, such as L-shaped or T-shaped, where certain geometry-related singularities are usually present in the solution. A common practice in discretization to efficiently handle this type of singularity is to use a nonuniform grid, for instance, generated from an adaptive procedure, so that the grid is much finer near reentrant corners where the singularities are located and coarser for other parts of the region where the solution is smooth. As an example, Fig. 1.1 shows a T-shaped domain with two reentrant corners located at $(-1, 0)$ and $(1, 0)$. The solution of a boundary value problem with smooth data behaves like $\rho^{2/3} \sin \frac{2}{3}\theta$ around each of these corners, where (ρ, θ) are local polar coordinates. Also seen from the figure is a nonuniform grid adapted to singularities at the two corners.

2. Expression of the Schur complement as a matrix function. We consider the elliptic Dirichlet boundary value problem on a rectangular domain Ω with a separable and self-adjoint operator of the form $L = L_x + L_y$,

$$
\begin{aligned}
L_x &= \tfrac{-\partial}{\partial x}\left(a(x)\tfrac{\partial}{\partial x}\right) + c(x), \\
L_y &= \tfrac{-\partial}{\partial y}\left(b(y)\tfrac{\partial}{\partial y}\right) + d(y).
\end{aligned} \tag{2.1}
$$

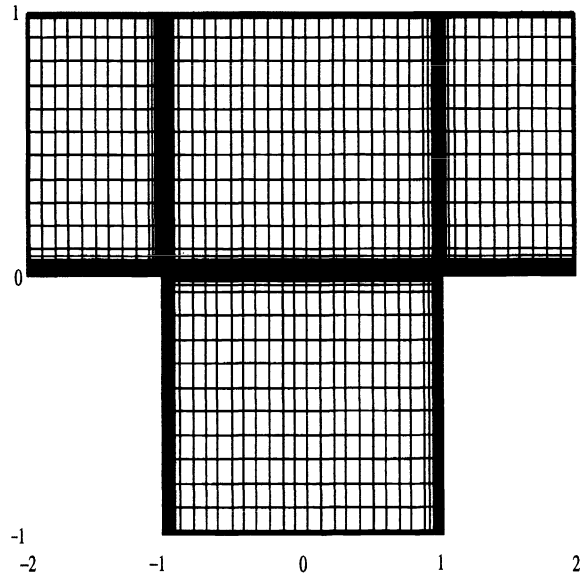


FIG. 1.1. *T-shaped domain with an adaptive nonuniform grid suitable for singularities in the solution at the reentrant corners* $(-1, 0)$ *and* $(1, 0)$.

Assume that Ω is discretized by a nonuniform tensor product grid with $\{h_x^i\}_{i=1,\ldots,n_x+1}$ and $\{h_y^i\}_{i=1,\ldots,n_y+1}$ being the spacings for the x and y directions. Using the standard finite differences, L_x is discretized by a tridiagonal matrix A_{FD}^x:

$$\left[-\frac{2a\left(\frac{x_{i-1}+x_i}{2}\right)}{h_x^i\left(h_x^i+h_x^{i+1}\right)},\ \frac{2\left(h_x^i a\left(\frac{x_i+x_{i+1}}{2}\right)+h_x^{i+1}a\left(\frac{x_{i-1}+x_i}{2}\right)\right)}{h_x^i h_x^{i+1}\left(h_x^i+h_x^{i+1}\right)}+c(x_i),\ -\frac{2a\left(\frac{x_i+x_{i+1}}{2}\right)}{h_x^{i+1}\left(h_x^i+h_x^{i+1}\right)}\right]$$

and, similarly, L_y is discretized by A_{FD}^y. The discrete analog of L can be expressed as

$$A_{FD} = A_{FD}^x \otimes I_{n_y} + I_{n_x} \otimes A_{FD}^y, \tag{2.2}$$

where $\otimes$ denotes the Kronecker product and I_k is the identity matrix of order k. The matrix A_{FD} is nonsymmetric when the grid is nonuniform. To preserve the symmetric positive definite (SPD) property for obvious reasons, A_{FD} is usually scaled to become a SPD matrix A by

$$\begin{aligned} A &= (\Theta_x \otimes \Theta_y)A_{FD} \\ &= (\Theta_x \otimes \Theta_y)(A_{FD}^x \otimes I_{n_y}) + (\Theta_x \otimes \Theta_y)(I_{n_x} \otimes A_{FD}^y) \\ &= (\Theta_x A_{FD}^x) \otimes \Theta_y + \Theta_x \otimes (\Theta_y A_{FD}^y) \\ &= A_x \otimes \Theta_y + \Theta_x \otimes A_y, \end{aligned} \tag{2.3}$$

where

$$\Theta_x = \operatorname{diag}\left(\frac{h_x^i + h_x^{i+1}}{2}\right),$$

$$\Theta_y = \operatorname{diag}\left(\frac{h_y^i + h_y^{i+1}}{2}\right),$$

and $A_x \equiv \Theta_x A_{FD}^x$, $A_y \equiv \Theta_y A_{FD}^y$ are tridiagonal SPD matrices. When $c(x) \equiv 0$ and $d(y) \equiv 0$, the five-point-star finite difference stencil (2.3) is identical to the linear finite element stiffness matrix.

Suppose Ω is decomposed into two subdomains Ω_1 and Ω_2 by an interface Γ which is a horizontal grid line, and there are m_1 and m_2 interior horizontal grid lines in Ω_1 and Ω_2, respectively. It is easy to see that $m_1 + m_2 + 1 = n_y$. Assume that the horizontal grid lines are ordered from the boundary towards Γ for each subdomain; then we can correspondingly write A_y and Θ_y as

$$A_y = \begin{bmatrix} A_y^1 & 0 & \beta_y^{10}\mathbf{e}_{m_1} \\ 0 & A_y^2 & \beta_y^{20}\mathbf{e}_{m_2} \\ \beta_y^{10}\mathbf{e}_{m_1}^T & \beta_y^{20}\mathbf{e}_{m_2}^T & \alpha_y^0 \end{bmatrix} \tag{2.4}$$

and

$$\Theta_y = \begin{pmatrix} \Theta_y^1 & 0 & 0 \\ 0 & \Theta_y^2 & 0 \\ 0 & 0 & \theta_y^0 \end{pmatrix}, \tag{2.5}$$

where A_y^i and Θ_y^i are the corresponding tridiagonal and diagonal matrices for Ω_i with the proper ordering, and $\mathbf{e}_k$ is the unit vector of order k. The matrix A also has the block form

$$A = \begin{bmatrix} A_1 & 0 & B_1 \\ 0 & A_2 & B_2 \\ B_1^T & B_2^T & D \end{bmatrix}, \tag{2.6}$$

where

$$\begin{aligned} A_i &= A_x \otimes \Theta_y^i + \Theta_x \otimes A_y^i, \qquad i = 1, 2, \\ D &= \theta_y^0 A_x + \alpha_y^0 \Theta_x, \\ B_i &= \beta_y^{i0} \Theta_x \otimes \mathbf{e}_{m_i}, \qquad i = 1, 2. \end{aligned} \tag{2.7}$$

The interface matrix

$$S \equiv D - \sum_{i=1}^{2} B_i^T A_i^{-1} B_i, \tag{2.8}$$

which is called the *Schur complement of* $\operatorname{diag}(A_i)$ *in* A and denoted by $(A/\operatorname{diag}(A_i))$ [7], plays a key role in domain decomposition-based methods. Theorem 2.1 states that S can be expressed as a matrix function.

THEOREM 2.1. *Let*

$$\begin{aligned} T_x &\equiv \Theta_x^{-1/2} A_x \Theta_x^{-1/2}, \\ T_y^i(t) &\equiv t\Theta_y^i + A_y^i, \qquad i = 1, 2; \end{aligned} \tag{2.9}$$

then the Schur complement can be expressed as

$$S = \Theta_x^{1/2} f(T_x) \Theta_x^{1/2}, \tag{2.10}$$

where

$$f(t) = (\theta_y^0 t + \alpha_y^0) - \sum_{i=1}^{2} \left(\frac{\beta_y^{i0}}{l^i_{m_i m_i}} \right)^2, \tag{2.11}$$

and $l^i_{m_i m_i}$ is the last diagonal element of the Cholesky factor of $T_y^i(t)$.

Proof. From (2.7) and (2.8) we have

$$S = (\theta_y^0 A_x + \alpha_y^0 \Theta_x) - \sum_{i=1}^{2} (\beta_y^{i0} \Theta_x \otimes \mathbf{e}_{m_i}^T) A_i^{-1} (\beta_y^{i0} \Theta_x \otimes \mathbf{e}_{m_i}). \tag{2.12}$$

To express S as a matrix function we want to change Θ_x to I_{n_x} in the right-hand side of (2.12) in order to use the Kronecker product properties. Multiplying (2.12) by $\Theta_x^{-1/2}$ symmetrically, we have

$$\Theta_x^{-1/2} S \Theta_x^{-1/2} = (\theta_y^0 T_x + \alpha_y^0 I_{n_x}) - \sum_{i=1}^{2} (\beta_y^{i0})^2 (I_{n_x} \otimes \mathbf{e}_{m_i}^T) \tilde{A}_i^{-1} (I_{n_x} \otimes \mathbf{e}_{m_i}), \tag{2.13}$$

where

$$\begin{aligned} \tilde{A}_i &= (\Theta_x^{-1/2} \otimes I_{m_i}) A_i (\Theta_x^{-1/2} \otimes I_{m_i}) \\ &= (\Theta_x^{-1/2} \otimes I_{m_i})(A_x \otimes \Theta_y^i + \Theta_x \otimes A_y^i)(\Theta_x^{-1/2} \otimes I_{m_i}) \\ &= T_x \otimes \Theta_y^i + I_{n_x} \otimes A_y^i. \end{aligned} \tag{2.14}$$

Using the properties of the Kronecker product, we can express the right-hand side of (2.13) as a function of T_x by formally substituting T_x by t, I_{n_x} by 1, and $\otimes$ by a normal product, which leads to

$$\Theta_x^{-1/2} S \Theta_x^{-1/2} = f(T_x) \tag{2.15}$$

with $f(t)$ defined by

$$f(t) = (\theta_y^0 t + \alpha_y^0) - \sum_{i=1}^{2} (\beta_y^{i0})^2 \mathbf{e}_{m_i}^T \left[T_y^i(t) \right]^{-1} \mathbf{e}_{m_i}. \tag{2.16}$$

It is easy to verify by forward and backward substitutions that

$$\mathbf{e}_{m_i}^T \left[T_y^i(t) \right]^{-1} \mathbf{e}_{m_i} = (l^i_{m_i m_i})^{-2}, \quad i = 1, 2. \tag{2.17}$$

Combining (2.15) through (2.17) we thus complete the proof. □

We make several remarks about Theorem 2.1.

Remark 2.1. The function $f(t)$ only depends on the operators A_y, Θ_y, and the location of Γ, namely m_1 and m_2. It is independent of any information in the x-direction.

Remark 2.2. To evaluate $f(t)$ at a given point using (2.11) one only needs to factor two tridiagonal matrices. From the proof of Theorem 2.1, it is seen that actual forward and backward substitutions in (2.17) can also be avoided by the special ordering of horizontal grid lines for each subdomain. In addition, no storage is required in the Cholesky factorization since only the last diagonal element $l^i_{m_i m_i}$ is used in (2.11) for each i.

Remark 2.3. Instead of using the Cholesky factorization, one can use the eigendecomposition for $[\Theta_y^i]^{-1/2} A_y^i [\Theta_y^i]^{-1/2}$ to get

$$T_y^i(t) = [\Theta_y^i]^{1/2} W_i (t I_{m_i} + \Lambda_i) W_i^T [\Theta_y^i]^{1/2}, \tag{2.18}$$

where $\Lambda_i = \operatorname{diag}(\lambda_i(j))$ and W_i are eigenvalues and eigenvectors for $[\Theta_y^i]^{-1/2} A_y^i [\Theta_y^i]^{-1/2}$. Let $\mathbf{z}_i = W_i^T [\Theta_y^i]^{-1/2} \mathbf{e}_{m_i}$; then we have, from (2.16),

$$f(t) = (\theta_y^0 t + \alpha_y^0) - \sum_{i=1}^{2} (\beta_y^{i0})^2 \sum_{j=1}^{m_i} \frac{(z_i(j))^2}{t + \lambda_i(j)}, \tag{2.19}$$

where $z_i(j)$ is the jth element of $\mathbf{z}_i$.

We can obtain another expression of $f(t)$ by introducing two sets of orthogonal polynomials $\{P_j^i(t)\}$ and $\{Q_j^i(t)\}$ defined in terms of the three-term recurrence relation, for $i = 1, 2$,

$$\begin{aligned}
P_{-1}^i(t) &= 0;\\
P_0^i(t) &= 1;\\
\beta_j^i P_j^i(t) &= (\theta_y^i(j)t + \alpha_j^i) P_{j-1}^i(t) - \beta_{j-1}^i P_{j-2}^i(t), \qquad j = 1, 2, \ldots, m_i;\\
Q_{-1}^i(t) &= 0;\\
Q_0^i(t) &= 1;\\
\beta_{m_i-j}^i Q_j^i(t) &= (\theta_y^i(m_i - j + 1)t + \alpha_{m_i-j+1}^i) Q_{j-1}^i(t) - \beta_{m_i-j+1}^i Q_{j-2}^i(t),\\
&\qquad j = 1, \ldots, m_i;
\end{aligned} \tag{2.20}$$

where $A_y^i \equiv [\beta_{j-1}^i,\ \alpha_j^i,\ \beta_j^i]$ and $\theta_y^i(j)$ is the jth diagonal element of Θ_y^i. Then, as in (2.12) in [1] (note Q_j^i here is R_j^i in [1]), we have

$$\{[T_y^i(t)]^{-1}\}_{qs} = \begin{cases} \dfrac{P_{s-1}^i(t) Q_{m_i-q}^i(t)}{\beta_{m_i}^i P_{m_i}^i(t)}, & q \geq s, \\[2ex] \dfrac{P_{q-1}^i(t) Q_{m_i-s}^i(t)}{\beta_{m_i}^i P_{m_i}^i(t)}, & s \geq q. \end{cases} \tag{2.21}$$

Therefore, (2.16) can be written in terms of these polynomials as

$$f(t) = (\theta_y^0 t + \alpha_y^0) - \sum_{i=1}^{2} (\beta_y^{i0})^2 \frac{P_{m_i-1}^i(t)}{\beta_{m_i}^i P_{m_i}^i(t)}. \tag{2.22}$$

By setting $\beta_{m_i}^i \equiv \beta_y^{i0}$ in (2.20), (2.22) becomes

$$f(t) = \frac{(\theta_y^0 t + \alpha_y^0) P_{m_1}^1(t) P_{m_2}^1(t) - \beta_y^{10} P_{m_1-1}^1(t) - \beta_y^{20} P_{m_2-1}^1(t)}{P_{m_1}^1(t) P_{m_2}^2(t)}. \tag{2.23}$$

As in (2.20), if we define another set of orthogonal polynomials $\{P_j(t)\}$ according to the global tridiagonal matrix for the operator $t\Theta_y + A_y$ with the natural bottom-to-top ordering for all

interior horizontal grid lines in Ω, then by denoting $t\Theta_y + A_y \equiv [\beta_{j-1},\ \theta_j t + \alpha_j,\ \beta_j]$ with $\beta_{n_y} = 1$, it can be verified that (2.23) can be written as

$$f(t) = \frac{P_{n_y}(t)}{P^1_{m_1}(t)P^2_{m_2}(t)}. \tag{2.24}$$

Therefore, $\{P_j(t)\}$, $\{P^1_j(t)\}$, and $\{P^2_j(t)\}$ play a role analogous to the Chebyshev polynomials as in the constant coefficient and uniform grid case. The expressions (2.19) and (2.24) for $f(t)$, or similarly for $1/f(t)$, can be viewed as an extension of the work in [1] and [14] to the nonuniform grid case. They are all equivalently related to each other through the theory of orthogonal polynomials.

Remark 2.4. There are two reasons to favor the use of (2.11) to evaluate $f(t)$ in our approach. First, as shown in [11] and [12], $f(t)$ needs to be evaluated at only a few points for the interpolation. However, those approaches [14] using (2.19), where $f(t)$ or its reciprocal is expanded into a sum of partial fractions, require the computation of all the eigenvalues and some of the eigenvectors of matrices that are related to A_y. Using (2.22) with the three-term recurrence requires about the same work as using (2.11). Expression (2.24) has a better mathematical form than (2.22), but it requires about twice as much work as that of (2.22) because of computing $P_{n_y}(t)$. The second reason is the numerical stability problem, which is even more important. Numerical instability occurs especially when grids are very nonuniform. Numerical experiments show that the eigenvalue problem is often very ill conditioned and that the three-term recurrence computation is also very unstable. This can be easily seen from (2.20) because usually $\theta^i_y(j)t + \alpha^i_j >> \beta^i_j$, namely, a wrong pivot is used in the computation. Fortunately, the matrices $T^i_y(t)$, $i = 1, 2$, are SPD, and therefore the Cholesky factorization is numerically stable.

Remark 2.5. Finally, it is easy to see that all the theory developed in [1] can be similarly extended to the nonuniform grid case along the line of the argument in Remark 2.3. Therefore, the marching algorithms in [1] and the parallel direct solvers in [9] can be correspondingly extended in a trivial way.

3. Function approximation and preconditioners. This section discusses finding a simple function $r(t)$ that approximates $f(t)$ in (2.11) such that conditions (1.2) are satisfied. Therefore, the matrix

$$M \equiv \Theta_x^{1/2} r(T_x) \Theta_x^{1/2} \tag{3.1}$$

is a good preconditioner for S in (2.10) when the PCG method is applied because

$$\begin{aligned} \kappa(M^{-1}S) &= \kappa(\Theta_x^{-1/2} q(T_x) \Theta_x^{1/2}) \\ &= \frac{\max\limits_{t_i \in \sigma(T_x)} |q(t_i)|}{\min\limits_{t_i \in \sigma(T_x)} |q(t_i)|}. \end{aligned} \tag{3.2}$$

As discussed in [11], a natural candidate for $r(t)$ is a rational function of low degree. If $r(t) = \Pi_k (t - a_k)/(t - b_k)$, then $M^{-1}S$ can be computed by a sequence of solves and multiplies with tridiagonal matrices since T_x is tridiagonal. We first describe a general approach to construct such a rational approximation by the *weighted rational Chebyshev approximation*

$$\min_{r(t) \in R^l_m} \max_{t \in [a,b]} \left| \frac{g(t) - r(t)}{w(t)} \right|, \tag{3.3}$$

where $g(t)$ is a target function to be approximated, $w(t)$ is a weight function, and R_m^l is the approximation function space

$$R_m^l = \left\{ r(t) \;\middle|\; r(t) = \frac{p_l(t)}{q_m(t)}, \right.$$
$$\left. p_l(t) \text{ and } q_m(t) \text{are polynomials of degree } l \text{ and } m, \text{ respectively} \right\}.$$

THEOREM 3.1. *Assume that $r(t)$ is the optimal solution of* (3.3) *with $g(t) = f(t)$, $w(t) = f(t)$, $a = \lambda_{\min}(T_x)$, $b = \lambda_{\max}(T_x)$. Let*

$$\varepsilon = \max_{t\in[a,b]} \left| \frac{f(t) - r(t)}{f(t)} \right|, \tag{3.4}$$

and assume $\varepsilon < 1$; then we have

$$\kappa(M^{-1}S) \le \frac{1+\varepsilon}{1-\varepsilon}. \tag{3.5}$$

Proof. From (3.4), we have

$$\left| 1 - \frac{r(t)}{f(t)} \right| \le \varepsilon \quad \forall t \in [a, b]. \tag{3.6}$$

This can be written as

$$1 - \varepsilon \le \frac{r(t)}{f(t)} \le 1 + \varepsilon \quad \forall t \in [a, b]. \tag{3.7}$$

It follows that

$$\begin{aligned} \max_{t_i\in\sigma(T_x)} |q(t_i)| &\le \frac{1}{1-\varepsilon}, \\ \min_{t_i\in\sigma(T_x)} |q(t_i)| &\ge \frac{1}{1+\varepsilon}. \end{aligned} \tag{3.8}$$

Notice that

$$\begin{aligned} \kappa(M^{-1}S) &\equiv \frac{\lambda_{\max}(M^{-1}S)}{\lambda_{\min}(M^{-1}S)} \\ &= \frac{\lambda_{\max}(\Theta_x^{-1/2} q(T_x) \Theta_x^{1/2})}{\lambda_{\min}(\Theta_x^{-1/2} q(T_x) \Theta_x^{1/2})} \\ &= \frac{\max_{t_i\in\sigma(T_x)} |q(t_i)|}{\min_{t_i\in\sigma(T_x)} |q(t_i)|}. \end{aligned} \tag{3.9}$$

Then (3.5) is obtained from (3.8) and (3.9). This completes the proof. □

There are many efficient algorithms devised for the weighted rational Chebyshev approximation; see, for example, [13]. The error ε in (3.4) depends on R_m^l, i.e., the degrees l and m, and for fast convergence of the PCG method one wishes ε to be as small as possible, which requires increasing l and m. On the other hand, using large l and m implies high expense in solving the preconditioning system and also, of less importance, in solving the

weighted rational Chebyshev approximation problem. So there is a trade-off in choosing l and m properly.

Another approach, as proposed in [11], is the more intuitive strategy of observing that $f(t)$ has a *two-part* property. That is, $f(t)$ looks mostly like a linear function; this part is called the *easy part*. At the left-end region of $\sigma(T_x)$ the few smallest eigenvalues make $f(t)$ behave like $t^{1/2}$; this part is called the *hard part*. Furthermore, for a uniform y-direction grid, [11] shows that $f(t)$ does not depend very much on the location of Γ, i.e., on m_1 and m_2; the difference can only be seen in the hard part. If we use a simple rational function

$$z(t) \equiv \frac{at + b}{ct + d} \tag{3.10}$$

to construct $r(t)$ in two phases, then $z(t)$ can be easily determined by three interpolating points using divided differences. The corresponding preconditioning problem, after a scaling, reads

$$(Tx + e_1 I_{n_x})\mathbf{u} = (T_x + e_2 I_{n_x})\mathbf{v}, \tag{3.11}$$

where $e_1 = b/a$ and $e_2 = d/c$. The linear system (3.11) can be solved by

$$\mathbf{u} = \mathbf{v} + (e_2 - e_1)(T_x + e_1 I_{n_x})^{-1}\mathbf{v} \tag{3.12}$$

using a SPD tridiagonal solver. We construct $r(t)$ as follows: First use $r_1(t)$ of the form (3.10) to remove the hard part from $f(t)$ by

$$f_1(t) = f(t) \,/\, r_1(t), \tag{3.13}$$

where $r_1(t)$ approximates well the hard part of $f(t)$. It is natural to compute $r_1(t)$ by interpolating the first three smallest eigenvalues of T_x. Observe that for small eigenvalues we can use T_x^* instead of T_x to get fairly good estimates, where T_x^* is the analog of T_x when the x-direction grid is made uniform. For a constant coefficient operator L_x, there exists an analytic expression for the eigenvalues of T_x^* so that one can avoid completely computing eigenvalues for T_x. Now $f_1(t)$ is almost a linear function, so we can find a good approximation $r_2(t)$ to it of the form (3.10). We choose the first interpolating point the same as for $r_1(t)$, the other two are chosen as the two largest eigenvalues of T_x; only rough estimates of them are required. Thus, we define $r(t)$ as

$$r(t) \equiv r_1(t) r_2(t). \tag{3.14}$$

Examples in [11] show that this *two-phase* strategy is very effective for the case of constant coefficient operators and uniform y-direction grids. In general, the behavior of $f(t)$ depends on the y-direction information, including the grid nonuniformity, the location of Γ, and the operator coefficients. A detailed experimental study about this dependence is found in [12]. In any case, a particular $f(t)$ has the so-called *two-part* property; therefore, the two-phase approximation strategy can also be applied. We give one example to illustrate the effectiveness of this approach and refer to [12] for more extensive experimental results. In this experiment, we solve a model problem of Poisson equation with Dirichlet condition on a unit square domain using a nonuniform grid as shown in Fig. 3.1. The effects of variable coefficients of the operator are similar to those of the grid nonuniformity. The grid size is 61×33. The spacings in each direction are of an exponential distribution to account for an exponential type of singularity in the solution of (2.1). More specifically, for the x direction the distribution used is

$$h_x^i = \min\{x_i^\alpha, 0.1\},$$

where $\alpha = 1.5$ and we start from $x = 1$ towards $x = 0$ until the specified number of grid points is reached, which is denoted as an x^{α}-*distribution*. Similarly for the y-direction, we take $\alpha = 1.2$ and $\max_i\{h_y^i\} = 0.05$. The interface Γ is chosen such that $m1 = m2 = 15$, namely, the work is equipartitioned for two subdomains.

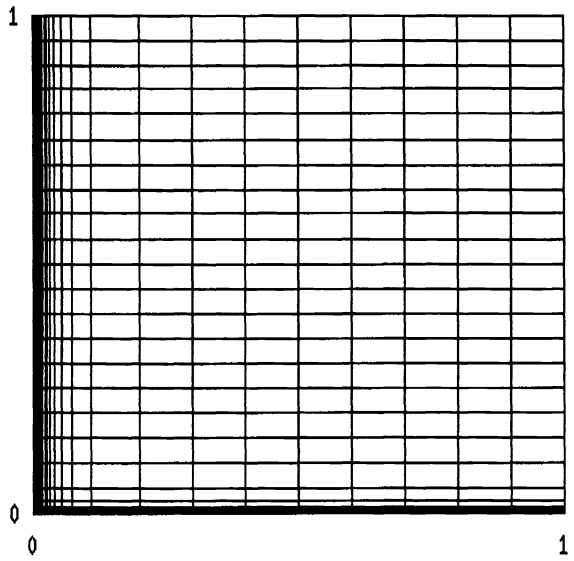


FIG. 3.1. *The nonuniform grid used in the experiment is refined along the axes using the distributions of* $y^{1.2}$ *(vertical) and* $x^{1.5}$ *(horizontal).*

The two-phase approximation strategy is used to construct the rational approximation for the preconditioner in this example. The corresponding function curves involved in the rational approximation are shown in Figs. 3.2–3.4. They illustrate the two-part behavior of $f(t)$, its domain and range used in the approximation, and the behavior of $q(t)$ that determines the convergence behavior of the PCG method. The condition number of the preconditioned system is 1.106.

With this preconditioner, the PCG method converges in only four steps. The least squares error for the last two iterates is 2.9×10^{-6} (single precision 32 bits is used in the computation). In contrast, the ordinary CG method does not converge after 100 steps and its error is 4.2×10^{-3} at this point. It is also important to notice that the matrix T_x, or similarly T_y, is too ill-conditioned for the eigendecomposition approach, so that no useful information is generated by standard IMSL eigenvalue subroutines. Therefore, any approach that requires eigendecomposition cannot be applied for this case.

4. Preconditioning for nonrectangular domains. To precondition the PCG method for a nonrectangular domain, it is natural to use an embedded rectangular domain to reduce the nonrectangular problem to a rectangular one because the remote parts of the domain have a less significant effect on the interface Schur complement. More specifically, let S be the Schur complement for the nonrectangular domain Ω, Ω_0 be the embedded rectangle by shifting Γ up to $\partial\Omega$ in both directions, and S_0 be the corresponding Schur complement for Ω_0. Further, let M be a preconditioner for Ω_0 (one of those discussed earlier). The combined effect of these two preconditioners is given by Theorem 4.1.

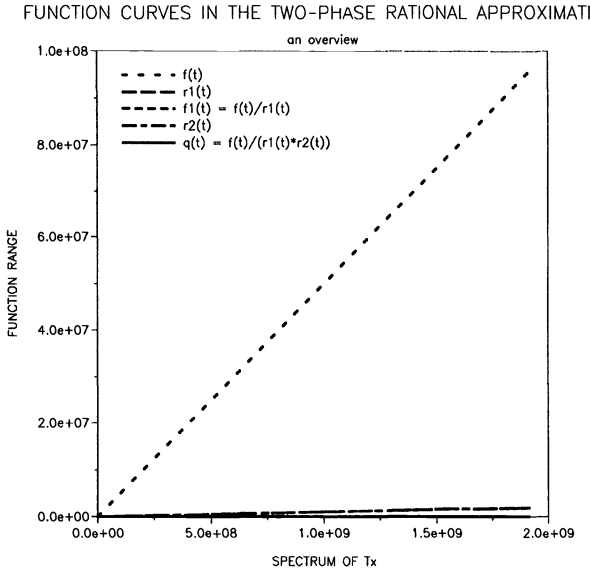


FIG. 3.2. *The function $f(t)$, its approximations, and resulting $q(t)$. On this scale one only sees the linear part of $f(t)$; the curve $q(t)$ is superimposed on the x-axis.*

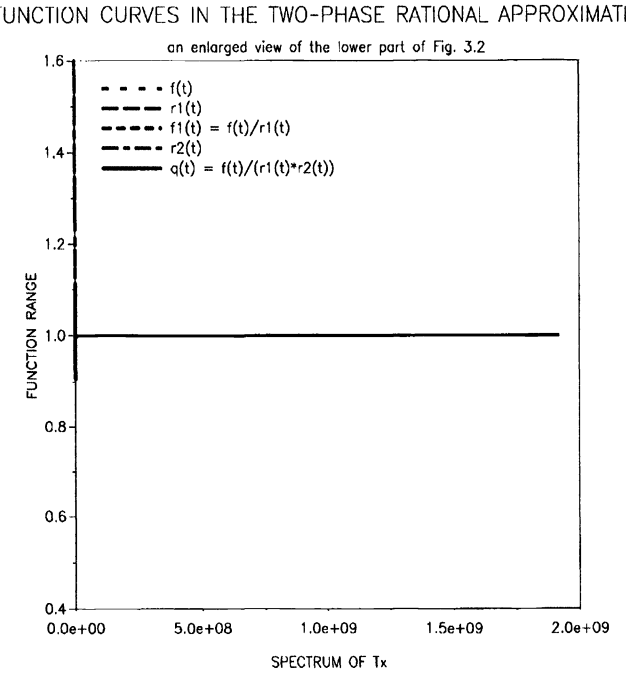


FIG. 3.3. *An enlarged view of the lower part of Fig.* 3.2 *shows that $q(t)$ is nearly* 1.0 *everywhere and the other functions rise along the y-axis and immediately go off the plot.*

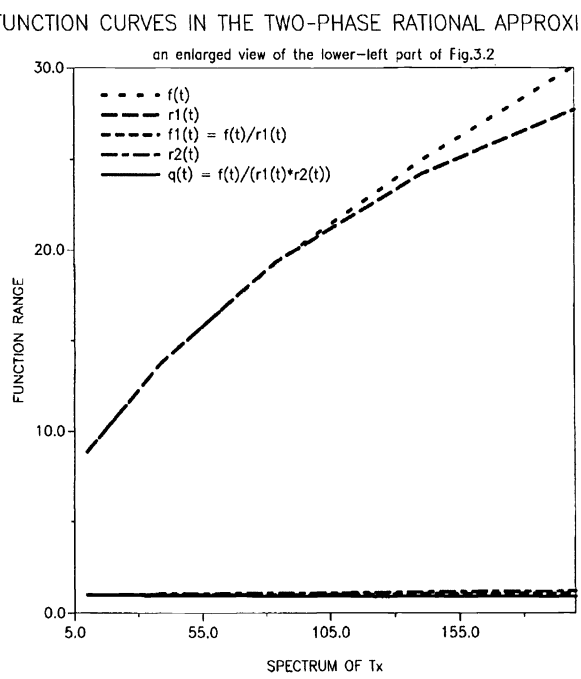


FIG. 3.4. *An enlarged view of the lower left part of Fig.* 3.2 *showing the hard part of* $f(t)$. *The approximation* $r_1(t)$ *fits* $f(t)$ *well for the small eigenvalues;* $f_1(t)$ *is nearly linear and rises much more slowly than* $f(t)$. *The approximation* $r_2(t)$ *fits* $f_1(t)$ *well and the resulting* $q(t)$ *is very close to* 1.0.

THEOREM 4.1.

$$\kappa(M^{-1}S) \le \kappa(S_0^{-1}S)\kappa(M^{-1}S_0). \tag{4.1}$$

Proof. From the definition, we have

$$\kappa(M^{-1}S) = \frac{\lambda_{\max}(M^{-1}S)}{\lambda_{\min}(M^{-1}S)}. \tag{4.2}$$

Notice that M, S_0, and S are all SPD. First, we have the estimate

$$\begin{aligned}
\lambda_{\max}(M^{-1}S) &= \lambda_{\max}(S_0^{1/2}M^{-1}SS_0^{-1/2}) \\
&\le \|S_0^{1/2}M^{-1}SS_0^{-1/2}\|_2 \\
&= \|(S_0^{1/2}M^{-1}S_0^{1/2})(S_0^{-1/2}SS_0^{-1/2})\|_2 \\
&\le \|S_0^{1/2}M^{-1}S_0^{1/2}\|_2\|S_0^{-1/2}SS_0^{-1/2}\|_2 \\
&= \lambda_{\max}(S_0^{1/2}M^{-1}S_0^{1/2})\lambda_{\max}(S_0^{-1/2}SS_0^{-1/2}) \\
&= \lambda_{\max}(M^{-1}S_0)\lambda_{\max}(S_0^{-1}S).
\end{aligned} \tag{4.3}$$

Similarly, we have

$$
\begin{aligned}
1/\lambda_{\min}(M^{-1}S) &= \lambda_{\max}(S^{-1}M) \\
&\le \lambda_{\max}(S^{-1}S_0)\lambda_{\max}(S_0^{-1}M) \\
&= \frac{1}{\lambda_{\min}(S_0^{-1}S)\lambda_{\min}(M^{-1}S_0)}.
\end{aligned} \tag{4.4}
$$

Therefore, we obtain

$$
\begin{aligned}
\kappa(M^{-1}S) &\le \frac{\lambda_{\max}(S_0^{-1}S)\lambda_{\max}(M^{-1}S_0)}{\lambda_{\min}(S_0^{-1}S)\lambda_{\min}(M^{-1}S_0)} \\
&= \kappa(S_0^{-1}S)\kappa(M^{-1}S_0).
\end{aligned} \tag{4.5}
$$

The proof is complete. □

The bounds (1.6) imply that there is a generic constant upper bound for the factor $\kappa(S_0^{-1}S)$ in (4.1). More accurate bounds are derived in [6] for some particular domains. From those results, we have the following corollary.

COROLLARY 4.2. *Assume that the operator is a Laplacian. For all L-shaped and T-shaped domains, we have*

$$\kappa(M^{-1}S) \le 2\kappa(M^{-1}S_0), \tag{4.6}$$

and for all C-shaped domains, we have

$$\kappa(M^{-1}S) \le (2+\sqrt{2})\kappa(M^{-1}S_0). \tag{4.7}$$

For other operators, similar techniques in [6] can be applied to derive corresponding upper bounds.

5. Conclusions. This paper extends the approach of constructing preconditioners through a function approximation, presented in [11], to more general cases where grids can be nonuniform, operators can have variable coefficients but are separable and self-adjoint, and domains can be nonrectangular. Theoretical and experimental results show that this new approach is very simple, effective, and efficient. The extended theory of expressing the Schur complement, or the original matrix, as a function of a simple matrix can be applied for other purposes, such as fast direct solvers and in parallel computations.

Acknowledgment. We would like to thank Professor E. Gallopoulos for pointing us to the recent work [14] and his own work [9], which are related to our approach to some extent.

REFERENCES

[1] R. E. BANK (1977), *Marching algorithms for elliptic boundary value problems.* II: *The variable coefficient case*, SIAM J. Numer. Anal., 14, pp. 950–970.

[2] R. E. BANK AND D. J. ROSE (1977), *Marching algorithms for elliptic boundary value problems.* I: *The constant coefficient case*, SIAM J. Numer. Anal., 14, pp. 792–829.

[3] P. E. BJORSTAD AND O. B. WIDLUND (1984), *Solving elliptic problems on regions partitioned into substructures*, in Elliptic Problem Solvers II, G. Birkhoff and A. Schoenstadt, eds., Academic Press, New York, pp. 245–256.

[4] J. H. BRAMBLE, J. E. PASCIAK, AND A. H. SCHATZ (1986), *The construction of preconditioners for elliptic problems by substructuring,* I, Math. Comp., 46, pp. 361–369.

[5] T. F. CHAN (1987), *Analysis of preconditioners for domain decomposition*, SIAM J. Numer. Anal., 24, pp. 382–390.

[6] T. F. CHAN, T. Y. HOU, AND P. L. LIONS (1991), *Geometry related convergence results for domain decomposition algorithms*, SIAM J. Numer. Anal., 28, pp. 378–391.

[7] R. W. COTTLE (1974), *Manifestations of the Schur complement*, Linear Algebra Appl., 8, pp. 189–211.

[8] M. DRYJA (1982), *A capacitance matrix method for Dirichlet problem on polygonal region*, Numer. Math., 39, pp. 51–64.

[9] E. GALLOPOULOS AND Y. SAAD (1989), *Some fast elliptic solvers on parallel architectures and their complexities*, Internat. J. High Speed Comput., 1, pp. 113–141.

[10] G. H. GOLUB AND D. MAYERS (1983), *The use of pre-conditioning over irregular regions*, Lecture at Sixth Int. Conf. on Computing Methods in Applied Sciences and Engineering, Versailles, France.

[11] M. MU (1995), *A new family of preconditioners for domain decomposition*, SIAM J. Sci. Stat. Comput., 16, to appear; Tech. Reports CSD-TR-92-064 and CER-92-27, Department of Computer Sciences, Purdue University, West Lafayette, IN, September, 1992.

[12] M. MU AND J. R. RICE (1993), *Constructing preconditioners with rational approximation*, in Proceedings of the Sixth SIAM Conference on Parallel Processing and Scientific Computing, R. F. Sincovec, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA.

[13] J. R. RICE (1992), *Numerical Methods, Software, and Analysis*, Chapter 11, Second Edition, Academic Press, Cambridge, MA.

[14] S. SANDER (1992), *Domain decomposition and rational approximation problems*, in Proceedings of the IMACS Symposium on Iterative Methods in Linear Algebra, April, 1991, Brussels, Belgium, P. De Groen and R. Beauwens, eds., Elsevier, Amsterdam, 1992.

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1467–1488, November 1994

USING KRYLOV METHODS IN THE SOLUTION OF LARGE-SCALE DIFFERENTIAL-ALGEBRAIC SYSTEMS*

PETER N. BROWN†, ALAN C. HINDMARSH†, AND LINDA R. PETZOLD‡

Abstract. In this paper, a new algorithm for the solution of large-scale systems of differential-algebraic equations is described. It is based on the integration methods in the solver DASSL, but instead of a direct method for the associated linear systems which arise at each time step, we apply the preconditioned GMRES iteration in combination with an Inexact Newton Method. The algorithm, along with those in DASSL, is implemented in a new solver called DASPK. We outline the algorithms and strategies used, and discuss the use of the solver. We develop and analyze some preconditioners for a certain class of DAE stems, and finally demonstrate the application of DASPK on two example problems.

Key words. differential-algebraic systems, Krylov methods

AMS subject classification. 65L05

1. Introduction.

This paper is concerned with the solution of large systems of differential-algebraic equations (DAEs). We write the system in the general form

$$F(t, y, y') = 0, \tag{1.1}$$

where F, y, and y' are N-dimensional vectors, and a consistent set of initial conditions $y(t_0) = y_0$, $y'(t_0) = y'_0$ is given.[1] The starting point for this work is the solver DASSL [20], [3]. In that code, the linear systems which arise at each time step are solved with dense or banded direct linear system solvers. For large problems, this is highly restrictive. Instead we consider the preconditioned GMRES (Generalized Minimal Residual) iterative method [21]. For large-scale systems, including method of lines solution of partial differential equations in two and three dimensions, this method can be quite effective, combined with a suitable preconditioner.

A number of previous papers [17], [11], [18], [7] have dealt with the solution of large-scale systems of ordinary differential equations (ODEs), $y' = f(t, y)$, via backward differentiation formulas for time-stepping in combination with preconditioned Krylov methods for solving the linear systems at each time step. A solver called LSODPK that is based on these methods was developed by Brown and Hindmarsh, and is described in [7]. A similar solver called VODPK, developed by Brown, Byrne, and Hindmarsh, is described in [9]. Chronopoulos and Pedro [12] have also developed a version of DASSL that includes iterative methods. However, their code uses completely different strategies for tests such as Newton convergence and linear iteration convergence, and their work does not address the issues that are specific to differential algebraic equations (DAEs) (as opposed to ODEs) that we focus on here.

*Received by the editors April 14, 1993; accepted for publication (in revised form) October 29, 1993. This research was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, and by Lawrence Livermore National Laboratory contract W-7405-ENG-48.

†Computing and Mathematics Research Division, L-316, Lawrence Livermore National Laboratory, Livermore, California 94550 (brown@icarus.llnl.gov, alanh@daphne.llnl.gov).

‡Department of Computer Science, University of Minnesota, Minneapolis, Minnesota 55455 (petzold@cs.umn.edu). The work of this author was partially supported by Army Research Office contract DAAL03-89-C-0038 with the University of Minnesota Army High Performance Computing Research Center, Army Research Office contract DAAL03-92-G-0247, Department of Energy contract DE-FG02-92ER25130, and the Minnesota Supercomputer Institute.

[1] By a consistent set of initial conditions, for the systems under consideration we mean that (1.1) should be satisfied at the initial time. For a more complete description of what it means for an initial condition to be consistent, see [3].

In this paper we extend the work on preconditioned Krylov methods for ODEs to DAEs. While many of the considerations remain the same, the solution of DAEs by this approach introduces some additional questions and difficulties. In particular, a preconditioner is *always* needed for DAEs (i.e., DAEs that are not ODEs). Preconditioners can be developed for classes of DAE systems arising with a particular structure. Here we develop and analyze preconditioners for a certain class of stiff DAE systems.

A new code, DASPK, has been developed based on this approach. We begin in §2 by describing the code in some detail, outlining especially those strategies and considerations that differ from the ODE case. In §3 we describe the use of this code. In §4 we develop and analyze a class of preconditioners for DAEs arising from reaction-diffusion systems. Finally, in §5 we present some numerical experiments applying the new code and preconditioners to the solution of several large-scale problems.

2. Overview of the DASPK algorithm. In DASPK, we have combined the time-stepping methods of DASSL with the preconditioned iterative method GMRES, for solving large-scale systems of DAEs of the form (1.1). Here we describe the algorithm.

2.1. Time-stepping. The underlying idea for solving DAE systems is due to Gear [16] and consists of replacing the solution and derivative in (1.1) by difference approximation, and solving the resulting equation for the solution at the current time t_n using Newton's method. For example, replacing the derivative by the backward difference in (1.1), we obtain the first-order formula

$$F\left(t_n, y_n, \frac{y_n - y_{n-1}}{h_n}\right) = 0, \tag{2.1}$$

where $h_n = t_n - t_{n-1}$. This equation is then solved at each time step using a modified Newton method,

$$y_n^{m+1} = y_n^m - \left(\frac{1}{h_n}\frac{\partial F}{\partial y'} + \frac{\partial F}{\partial y}\right)^{-1} F\left(t_n, y_n^m, \frac{y_n^m - y_{n-1}}{h_n}\right), \tag{2.2}$$

where m is the iteration index. As in DASSL, DASPK uses the backward differentiation formulas (BDFs) of orders 1 through 5 to approximate the derivative in (1.1). On every step, it chooses the order and step size based on the behavior of the solution. The integration methods and strategies for time-stepping are virtually identical to those in DASSL, and are described in detail in [3]. The equation to be solved on each time step is

$$F\left(t_n, y_n, \frac{\rho y_n}{h_n}\right) = 0, \tag{2.3}$$

where $\rho y_n = \sum_{i=0}^{k} \alpha_i y_{n-i}$ and α_i, $i = 0, 1, \ldots, k$ are the coefficients of the BDF method.

2.2. Nonlinear system solution. It is important to solve the nonlinear equation (2.3) efficiently. To simplify notation, we can rewrite this equation as

$$F(t, y, \alpha y + \beta) = 0, \tag{2.4}$$

where $\alpha = \alpha_0 / h_n$ is a constant which changes whenever the step size or order changes, β is a vector which depends on the solution at past times, and t, y, α, and β are evaluated at t_n. To simplify the discussion, we will sometimes refer to the above function simply as $F(y)$. Both DASPK and DASSL solve this equation by a modified version of Newton's method,

$$y^{m+1} = y^m - c\left(\alpha\frac{\partial F}{\partial y'} + \frac{\partial F}{\partial y}\right)^{-1} F(t, y^m, \alpha y^m + \beta). \tag{2.5}$$

Both solvers must therefore deal in some way with the iteration matrix

$$A = \alpha \frac{\partial F}{\partial y'} + \frac{\partial F}{\partial y}. \tag{2.6}$$

The direct methods within DASPK are virtually identical to those in DASSL for the treatment of (2.4). In that case, the iteration matrix (2.6) is computed and factored, and is then used for as many time steps as possible. By contrast, in the iterative methods option of DASPK, a preconditioner matrix P, which is an approximation to A that leads to a cheap linear system solution, is computed and preprocessed and used for as many time steps as possible. As we will see in the examples below, it is often possible to use a preconditioner over more steps than it would be possible to keep an iteration matrix in the direct option, because the iterative methods do the rest of the work in solving the system. One of the powerful features of the iterative approach is that it does not need to compute and store the iteration matrix A explicitly.[2] This is because the GMRES method, as we will see below, never actually needs this matrix explicitly. Instead, it requires only the action of A times a vector v. In DASPK, this matrix-vector product is approximated via a difference quotient on the function F in (2.4),

$$Av = F'(y)v \approx \frac{F(t, y + \sigma v, \alpha(y + \sigma v) + \beta) - F(t, y, \alpha y + \beta)}{\sigma}. \tag{2.7}$$

The GMRES algorithm requires products Av in which v is a vector of unit length (the norm is a weighted norm based on the user-defined error tolerances as described in §2.3.3) and y is the current iterate. In DASPK, σ is taken to be 1, as explained in §2.3.3. We note that, because y is current in (2.7), this amounts to taking a full Newton iteration in the iterative option of DASPK (rather than modified Newton, as in DASSL and the direct option of DASPK). In fact, for some highly nonlinear problems we have seen the iterative option in DASPK outperform the direct option in terms of time steps, corrector failures, etc., apparently for this reason.

In general, the value of α when A (or P) was last computed is different from the current value of α. We will denote the "old" value of α by $\hat{\alpha}$. If α is too different from $\hat{\alpha}$ in the direct method, then (2.5) may not converge. The constant c in (2.5) is chosen to speed up the convergence in the direct method when $\alpha \neq \hat{\alpha}$, and is given by

$$c = \frac{2}{1 + \alpha/\hat{\alpha}}. \tag{2.8}$$

See [3] for a derivation. For the iterative method, while the preconditioner may be based on an old $\hat{\alpha}$, the linear system to be solved is based on the current α, so in this case $c \equiv 1$.

The rate of convergence ρ of (2.5) is estimated, whenever two or more iterations have been taken, by

$$\rho = \left(\frac{||y^{m+1} - y^m||}{||y^1 - y^0||} \right)^{1/m}. \tag{2.9}$$

It is important to note at this point that all norms here are weighted norms, in which the weights depend on the error tolerances specified by the user, so as to account for the scaling of the problem; the details are given in the next subsection. The iteration is taken to have converged when

$$\frac{\rho}{1 - \rho} ||y^{m+1} - y^m|| < 0.33. \tag{2.10}$$

[2]Depending on the preconditioner, it may need to compute and store a preconditioner matrix explicitly. However, this matrix is hopefully much cheaper to generate and to store than the actual iteration matrix.

The basis for this test is that if the iteration is converging linearly at a rate ρ to y^*, then

$$\frac{\rho}{1-\rho}||y^{m+1} - y^m|| \approx ||y^{m+1} - y^*||. \tag{2.11}$$

If $\rho > 0.9$ or $m > 4$, and the iteration has not yet converged, then the stepsize is reduced, and/or an iteration matrix based on current approximations to y, y', and α is formed, and the step is attempted again. If the difference between the predictor and the first correction is very small (for the direct solver, this is relative to roundoff error in y; for the iterative solver, it is relative to the accuracy requested in solving the linear system), the iteration is taken to have converged (because the initial correction is so close that it is impossible to get a good rate estimate). The heuristic constants .33, .9, and 4 here have been taken from the DASSL algorithm without change and are discussed in [3].

For the iterative methods, convergence tests such as (2.10) need to be justified, because the Newton iterates are not computed exactly but instead with a relatively large error which is due to solving the linear system inexactly. The test (2.10) can be justified, at least to some extent, by considering the Newton/GMRES method in the framework of the theory of Inexact Newton Methods [14]. In this framework, the Newton iteration for $F(y) = 0$, including errors r^m due to solving the linear system inexactly, is written as

$$F'(y^m)\delta y^m = -F(y^m) + r^m, \tag{2.12a}$$

$$y^{m+1} = y^m + \delta y^m. \tag{2.12b}$$

Theorem 2.2 in Brown and Hindmarsh [6] justifies a convergence test based on rate of convergence for the inexact Newton iteration, provided the residuals r^m satisfy $||r^m|| \le \eta||F(y^m)||$, for $\eta < 1$. However, as in the case of ODEs, we prefer to terminate the linear iteration based on a condition like $||r^m|| \le \delta$. In this case, it can be argued heuristically, as in [6], that the test is still justified, provided that $\delta << \epsilon/\lambda$, where ϵ is the tolerance for the final computed Newton iterate, $||y^{m+1} - y^*|| \le \epsilon$, and $\lambda = ||(F')^{-1}(y^*)||$. Now, this is almost what we need, except that for most DAEs (and stiff ODEs), λ is likely to be quite large, which would seem to mandate a quite conservative test for the linear iteration. To see how the theory can be used to justify a less conservative test, multiply $F(y) = 0$, and hence (2.12a), by P^{-1}, where P is the preconditioner matrix described below. This changes nothing in terms of the Newton iterates or the final solution for y^*. Now, *assuming that the preconditioner is a good approximation to* F', in the sense that $||P^{-1}F'|| \approx O(1)$, the theory and heuristic arguments in [6], applied to $P^{-1}F$ instead of F, justify a termination criterion for the Newton iteration based on the rate of convergence, provided the *preconditioned* residuals for the linear iteration satisfy $||P^{-1}r^m|| \le \delta$, where $\delta << \epsilon$. In DASPK, we take $\epsilon = .33$, and take $\delta = .05\epsilon$ as the default tolerance for solving the linear iteration (the constant δ/ϵ can be adjusted optionally by the user, as described in §3). The norms used are weighted norms based on the user-defined error tolerances and are described in §2.3.2. We note that another desirable property of these tests is that they are invariant under scalings of the DAE (i.e., multiplying F on the left by some arbitrary matrix), provided the preconditioner has also been scaled accordingly.

In contrast to the ODE solvers LSODPK [7] and VODPK [9], which use preconditioned Krylov methods with left and/or right preconditioning, the DASPK solver allows only left preconditioning. The reason has to do with a basic difference between ODEs and DAEs. For a DAE system defined by $F(t, y, y') = 0$, the components of the vector F need not have any relation to those of y. For example, the two vectors need not have the same physical units in corresponding components. If a left preconditioner P_1 and a right preconditioner P_2 are allowed in the solution of the linear system $Ax = b$, where $A = F'$ and b is a value of $-F$, then the Krylov method in effect deals with the matrix $P_1^{-1}AP_2^{-1}$ and with residual vectors

$P_1^{-1}r$ $(r = b - Ax)$, and performs a convergence test on weighted norms of those vectors. But consistent choices of P_1 and P_2 are possible, with $P_1 P_2 \approx A$, for which $P_1^{-1}r$ does not have the same units as y. Then norms of the preconditioned residuals $P_1^{-1}r$ are meaningless as a measure of the quality of the current approximate solution vector x. In contrast, if $P_2 = I$ and P_1 is a consistent approximation to A, then $P_1^{-1}r$ has the same units as y in each component, and the convergence test in the Krylov algorithm, with the same weighted norm as used in the local error test, is completely consistent. Moreover, that convergence test is invariant under a change of scale in either the function F or the vector y (provided the absolute tolerances are rescaled consistently if y is). This consistency and scale invariance are not possible with preconditioning either on the right only or on both sides.

In DASPK, the iterative option *requires* the user to provide a preconditioner P. This is in part because the Newton iteration test, and hence ultimately the code reliability, is not justified without a halfway-reasonable preconditioner. It is also because *any nontrivial* DAE *needs a preconditioner*. Even a "nonstiff" DAE needs a preconditioner, to approximate the Jacobian of the constraint matrix. Also, it is known that the iteration matrix for any nontrivial DAE becomes more and more ill-conditioned as the step size is reduced [3]. Therefore, the preconditioner may need to rescale; scalings for some common classes of DAEs are discussed in [3]. When the DAE is really an ODE, with $F(t, y, y') = y' - f(t, y)$, the preconditioner could be taken as $P \approx \alpha * I$, although even for ODEs it is often better to provide a nontrivial preconditioner [7].

2.3. Linear system solution. Solving (2.5) requires the solution of a linear system

$$Ax = b \tag{2.13}$$

at each Newton iteration, where A is the $N \times N$ iteration matrix in (2.6), $x = y^{m+1} - y^m$ is an N-vector, and $b = -cF(t, y^m, \alpha y^m + \beta)$ is an N-vector.

In the direct option, this linear system is solved by either dense direct or banded direct Gaussian elimination with partial pivoting via LINPACK [15]. The iteration matrix is either provided by the user, or computed via finite difference quotients, as described in [3].

2.3.1. Description of the GMRES algorithm. In the case of iterative methods, the linear system (2.13) is solved by the preconditioned GMRES iterative method [21]. Depending on the options chosen, the method may be either the complete or the incomplete GMRES method, and it may or may not include restarting.

GMRES is one of a class of *Krylov subspace projection methods* [22]. The basic idea of these methods is as follows. If x_0 is an initial guess for the solution, then letting $x = x_0 + z$, we get the equivalent system $Az = r_0$, where $r_0 = b - Ax_0$ is the initial residual. We choose $z = z_l$ in the *Krylov subspace* $K_l = \text{span}\{r_0, Ar_0, \ldots, A^{l-1}r_0\}$. For the GMRES algorithm, z_l, hence $x_l = x_0 + z_l$, is specified uniquely by the condition

$$\|b - Ax_l\|_2 = \min_{x \in x_0 + K_l} \|b - Ax\|_2 \quad (= \min_{z \in K_l} \|r_0 - Az\|_2).$$

Here, $\|\cdot\|_2$ denotes the Euclidean norm.

GMRES uses the *Arnoldi process* [1] to construct an orthonormal basis of the Krylov subspace K_l. This results in an $N \times l$ matrix $V_l = [v_1, \ldots, v_l]$ and an $l \times l$ upper Hessenberg matrix H_l such that

$$H_l = V_l^T A V_l \quad \text{and} \quad V_l^T V_l = I_l \quad (= l \times l \text{ identity matrix}).$$

If the vectors $r_0, Ar_0, \ldots, A^l r_0$ are linearly independent, so that the dimension of K_{l+1}

is $l+1$, then the matrices $V_{l+1} = [v_1, \ldots, v_{l+1}]$ and $\bar{H}_l \in \mathbf{R}^{(l+1)\times l}$ defined by

$$\bar{H}_l = \begin{bmatrix} H_l \\ r^T \end{bmatrix}, \quad \text{where } r = (0, \ldots, 0, h_{l+1,l})^T \in \mathbf{R}^l$$

satisfy

$$AV_l = V_{l+1}\bar{H}_l.$$

Furthermore, letting $z = V_l y$, we find that $\|r_0 - Az\|_2 = \|\beta e_1 - \bar{H}_l y\|_2$, where $\beta = \|r_0\|_2$ and e_1 is the first standard unit vector in $\mathbf{R}^{l+1}$. The vector $y = y_l$ minimizing this residual is computed by performing a QR factorization of $\bar{H}_l$ using Givens rotations. Then the GMRES solution is $x_l = x_0 + V_l y_l$. As noted by Saad and Schultz [21], this QR factorization can be done progressively as each column appears, and one can compute the residual norm $\|b - Ax_l\|_2$ without computing x_l at each step. If the $\sin\theta$ elements of the Givens rotations are denoted by s_j $(j = 1, \ldots, l)$, then one obtains

$$\|b - Ax_l\|_2 = \beta|s_1 \cdots s_l|. \tag{2.14}$$

Combining these various parts gives the following algorithm for the basic (complete) GMRES method. Here $l_{\max}$ and δ are given parameters.

ALGORITHM 2.1 (GMRES).

1. Compute $r_0 = b - Ax_0$ and set $v_1 = r_0/\|r_0\|_2$.
2. For $l = 1, \ldots, l_{\max}$ do:
   (a) Form Av_l and orthogonalize it against $v_1, \ldots, v_l$ via

$$w_{l+1} = Av_l - \sum_{i=1}^{l} h_{il}v_i, \qquad h_{il} = (Av_l, v_i),$$
$$h_{l+1,l} = \|w_{l+1}\|_2,$$
$$v_{l+1} = w_{l+1}/h_{l+1,l}.$$

   (b) Update the QR factorization of $\bar{H}_l$.
   (c) Use (2.14) to compute $\rho_l = \|r_0\|_2 \cdot |s_1 \cdots s_l| = \|b - Ax_l\|_2$.
   (d) If $\rho_l \le \delta$, go to Step 3. Otherwise, go to (a).
3. Compute $x_l = x_0 + V_l y_l$, and stop.

In the above algorithm, if the test on ρ_l fails, and if $l = l_{\max}$ iterations have been performed, then one has the option of either accepting the final approximation x_l or setting $x_0 = x_l$ and then going back to step 1 of the algorithm. This last procedure has the effect of "restarting" the algorithm. When used with the default input options, DASPK does such restarts when necessary to achieve convergence.

As l gets large, much work is required to make v_{l+1} orthogonal to all the previous vectors $v_1, \ldots, v_l$. This has motivated the development of an incomplete version of GMRES (denoted by IGMRES), which differs from Algorithm 2.1 only in that the sum in step 2(a) begins at $i = i_0 = \max(1, l - p + 1)$ instead of at $i = 1$. An inexpensive evaluation of the residual norm is still possible, with the insertion of another factor in (2.14). Details are given in [7]. In many cases, the incomplete algorithm can give significant savings over the complete one. For example, when A is symmetric (or nearly so), $\bar{H}_l$ is tridiagonal (or nearly so), and one can take $p = 2$. In DASPK, the default iterative method is (complete) GMRES, but IGMRES is available with the use of optional input parameters.

When $l_{\max} = N$, Saad and Schultz [21] have given a convergence analysis of Algorithm 2.1 that shows the GMRES iterates converge to the true solution of (2.10) in at most N iterations. While GMRES is guaranteed to converge, it can exhibit poor performance, especially when used without preconditioning. This is commonly referred to as *stagnation*. In such cases, there is little or no reduction in the residual norm ρ_l for several iterates. In [7], a dramatic example of stagnation is given wherein $\rho_l = \rho_0$ for all $l < N$, and $\rho_N = 0$. When $l_{\max} < N$ and A is positive definite, the restarted version of GMRES is always guaranteed to converge, although it also can stagnate. For indefinite matrices, the restarted GMRES method can sometimes fail to converge. For more details, see [5], [7], and [21].

2.3.2. Scaling and preconditioning. Realistic DAE problems require the inclusion of scale factors, so that all vector norms become weighted norms in the problem variables. However, even the scaled iterative methods seem to be competitive only for a fairly narrow class of problems, namely ODEs characterized mainly by tight clustering in the spectrum of the system Jacobian. Thus, for robustness, it is essential to enhance the methods further. As in other contexts involving linear systems, preconditioning of the linear iteration is a natural choice. In what follows, the use of scaling and preconditioning is reviewed.

Scaling. The user of DASPK must provide parameters that define error tolerances to be imposed on the computed solution. These are relative and absolute tolerances RTOL and ATOL such that the combination

$$w_i = \text{RTOL}_i \left| y_{n-1}^i \right| + \text{ATOL}_i$$

is applied as a scale factor for component y^i during the time step from t_{n-1} to t_n. Specifically, a weighted root-mean-square norm

$$\|x\|_{\text{WRMS}} = \left[N^{-1} \sum_1^N (x^i/w_i)^2 \right]^{1/2}$$

is used on all error-like vectors. Thus if we define a diagonal matrix

$$D = \sqrt{N}\,\text{diag}(w_1, \ldots, w_N),$$

we can relate this to an ℓ_2 norm:

$$\|x\|_{\text{WRMS}} = \|D^{-1}x\|_2.$$

Because D contains the tolerances, the local error test on a vector e of estimated local errors is simply $\|e\|_{\text{WRMS}} \leq 1$.

The linear system in (2.13) can be restated in scaled form in terms of $D^{-1}x = \tilde{x}$ and $D^{-1}b = \tilde{b}$. Likewise, the nonlinear system $F(y) = 0$ can be restated in a scaled form $\tilde{F}(\tilde{y}) = 0$.

We note that, while DASSL allows the user to replace the norm subroutine, DASPK does not allow this. This is because a scaled L_2 norm is needed in the implementation of the GMRES algorithm. However, DASPK does allow a user-replaceable subroutine to define the weights in the norm. (The default is to set the weights according to the tolerances RTOL and ATOL via EWT(I) = RTOL(I)*ABS(Y(I)) + ATOL(I).) We recommend that this be attempted only after careful thought and consideration.

Preconditioning. When a basic iteration fails to show acceptable convergence on a given problem, preconditioning is often beneficial, especially when the cause of the slow convergence can be identified with one or more parts of the problem which are (individually) easier to deal

with than the whole problem. Generally, preconditioning in an iterative method for solving $Ax = b$ means applying the method instead to the equivalent system

$$(P^{-1}A)x = P^{-1}b \quad \text{or} \quad \bar{A}x = \bar{b}, \tag{2.15}$$

where P is chosen in advance. The preconditioned problem is easier to solve than the original problem provided that (1) linear systems $Px = c$ can be solved economically, and (2) P is in some way close to A. Condition (1) is essential because carrying out the method on $\bar{A}x = \bar{b}$ clearly requires evaluating vectors of the form $P^{-1}c$, at the beginning of the iteration, during each iteration, and at the end. Condition (2) is less well defined, but means simply that the convergence of the method for $\bar{A}x = \bar{b}$ should be much better than for $Ax = b$, because $\bar{A}$ is somehow close to the identity matrix (for which convergence is immediate).

It is essential that the scaling of the linear system (discussed above) be retained in the preconditioned methods. Since the scaling matrix D is based on the tolerance inputs to the ODE solver, D^{-1} can be thought of as removing the physical units from the components of x so that the components of $D^{-1}x$ can be considered dimensionless and mutually comparable. On the other hand, the matrix $A = \alpha \partial F/\partial y' + \partial F/\partial y$ is *not* similarly scaled, and so, because P is based on approximating A, the matrix

$$\bar{A} = P^{-1}A$$

is also not similarly (dimensionally) scaled. More precisely, it is easy to show that if the (i, j) elements of P each have the same physical dimension as that of A, i.e. the dimension of F_i/y_j, then the (i, j) element of $\bar{A}$ has the dimension of y_i/y_j. Similarly, for the vectors x and $\bar{b}$, the ith component of each has the same physical dimension as that of y_i. It follows that the diagonal scaling D^{-1} should be applied to x and $\bar{b}$ in the same way that it was applied to x and b without preconditioning. Thus we change the system (2.15) again to the equivalent scaled preconditioned system

$$(D^{-1}\bar{A}D)(D^{-1}x) = (D^{-1}\bar{b}) \quad \text{or} \quad \tilde{A}\tilde{x} = \tilde{b}. \tag{2.16}$$

Combining the two transformations, we have

$$\tilde{A} = D^{-1}P^{-1}AD, \quad \tilde{x} = D^{-1}x, \quad \tilde{b} = D^{-1}P^{-1}b. \tag{2.17}$$

2.3.3. Implementation details. In implementing the GMRES method in DASPK, many of the algorithmic issues that arise are the same as for the ODE case. We have carried over the treatment of these matters from LSODPK [7]. Below is a summary of those details.

- DASPK takes $x_0 = 0$, having no choice readily available that is clearly better.
- The scaling is incorporated in an explicit sense, storing vectors $\tilde{v}_i$ that arise in the method as it stands, rather than unscaled vectors $D\tilde{v}_i = v_i$.
- DASPK uses a difference quotient representation

$$Jv \approx [F(t, y + \sigma v, \alpha(y + \sigma v) + \beta) - F(t, y, \alpha y + \beta)]/\sigma.$$

- DASPK takes $\sigma = 1$ because $\|v\|_{\text{WRMS}} = 1$. Thus the perturbation vector v can be regarded as a small correction to y, since its WRMS norm (= 1) is a value that is accepted for local errors in y in the local error test. It is possible that other choices for σ might improve the truncation error in this difference quotient, but we have not studied this issue.
- The modified Gram–Schmidt procedure is used for orthogonalizing basis vectors.

- Failures to pass the convergence test in GMRES are handled in the context of the integration algorithm. If GMRES failed to converge but did reduce the residual norm, then it is restarted. On a convergence failure that did not reduce the residual norm, the time step is retried with the Jacobian routine (JAC) called to update matrix data used (if any) if that data is not current, or else the step size is first reduced (by a factor of .25) before the step retried. The user's PSOL routine can also return a failure flag that signals either an immediate halt or a retry of the step.
- The convergence test constant δ used as a bound on the residuals $\|b - Ax_l\|_{\mathrm{WRMS}}$ is taken to be $\delta = .05\epsilon$, where $\epsilon = .33$ is the tolerance on the nonlinear iteration in (2.10).

We can now state our algorithm for scaled preconditioned versions of the GMRES method. This is given for arbitrary x_0, for the sake of generality, and is denoted SPIGMR.

SCALED PRECONDITIONED INCOMPLETE GMRES (SPIGMR).

1. (a) $r_0 = b - Ax_0$; stop if $\|r_0\|_{\mathrm{WRMS}} < \delta$.
   (b) $\tilde{r}_0 = D^{-1}P^{-1}r_0$, compute $\|\tilde{r}_0\|_2 = \|P^{-1}r_0\|_{\mathrm{WRMS}}$, $\tilde{v}_1 = \tilde{r}_0/\|\tilde{r}_0\|_2$.
2. For $l = 1, 2, \ldots, l_{\max}$, do:
   (a) Compute $\tilde{A}\tilde{v}_l = D^{-1}P^{-1}AD\tilde{v}_l$.
   (b) $\tilde{h}_{il} = (\tilde{A}\tilde{v}_l, \tilde{v}_i)$, $\tilde{w}_{l+1} = \tilde{A}\tilde{v}_l - \sum_{i=i_0}^{l} \tilde{h}_{il}\tilde{v}_i$, where $i_0 = \max(1, l - p + 1)$.
   (c) $\tilde{h}_{l+1,l} = \|\tilde{w}_{l+1}\|_2$, $\tilde{v}_{l+1} = \tilde{w}_{l+1}/\tilde{h}_{l+1,l}$.
   (d) Update the QR factorization of $\tilde{H}_l = (\tilde{h}_{ij}) = Q_l R_l$ (an $(l+1) \times l$ matrix).
   (e) Compute the residual norm ρ_l indirectly.
   (f) If $\rho_l < \delta$, go to step 3; otherwise go to (a).
3. Compute $\|\tilde{r}_0\|_2 Q_l^T e_1 = (\bar{g}_l, g)^T$, $\tilde{z} = \tilde{V}_l \bar{R}_l^{-1} \bar{g}_l$, $x_l = x_0 + D\tilde{z}$.

3. Using DASPK. We have attempted to make DASPK as easy to use as possible, and also upward compatible with DASSL. However, the use of iterative rather than direct methods requires more information from the user and a deeper understanding of both the application and the solution process, particularly in the choice and implementation of an effective preconditioner. The direct methods of DASSL are available as an option of DASPK, and the user may find them useful for the purposes of getting started or debugging by way of smaller test problems.

The actual names of the single- and double-precision Fortran versions of DASPK are SDASPK and DDASPK, respectively. The call sequence is described in detail in the initial source file prologue, but we summarize the main points below.

3.1. Getting started. To get started, DASPK needs a consistent set of initial values T, Y, and YPRIME. This means that we must have F(T,Y,YPRIME) = 0 at the initial time.[3] Finding a consistent set of initial conditions for a given problem may not be trivial. There is an option in DASPK to compute the initial value of YPRIME, if the initial values of Y are known. For some problems, this method may require a good initial guess for YPRIME. In cases where not all the components of the initial vector Y are known, a nonlinear equation solver for large-scale systems such as NKSOL [8] can be very effective. It is important that the error tolerances for such a solver be set to be quite stringent, especially in comparison to the tolerances specified in DASPK. Otherwise, DASPK may not be able to get past the initial step because of a difficulty in satisfying its error tolerances. We note that there has been some recent work addressing the specification of consistent initial conditions for general DAE problems [19]; however, at the time of this writing we are aware of no corresponding general-purpose software.

[3] We note here that F is referred to as G in the actual code documentation.

3.2. Specifying the DAE. As in DASSL, the information about the function F in (1.1) is provided via a subroutine RES, which takes as input the time T and the vectors Y and YPRIME, and produces as output the vector DELTA, where DELTA = F(T,Y,YPRIME) is the amount by which the function F fails to be zero for the input values of T, Y, and YPRIME. The call sequence of RES in the use of DASPK differs from that in using DASSL, in that it includes CJ, which is the scalar α of (2.4), for possible use in scaling.

3.3. Solution by direct methods. The solution of the linear system at each time step can be done with either the direct methods of DASSL or the preconditioned GMRES method. For direct methods, specify INFO(12) = 0; the possiblities are the same as in DASSL. The user can provide a subroutine JAC to evaluate the iteration matrix (or Jacobian matrix), or else select an option for DASPK to approximate the matrix via finite differences. The matrix needed is $(\mathrm{CJ})\partial F/\partial y' + \partial F/\partial y$, where CJ is a scalar ($= \alpha$) which is proportional to $1/h$ and is provided as input to JAC. In the direct case, the JAC routine must have the following form.

SUBROUTINE JAC (T, Y, YPRIME, PD, CJ, RPAR, IPAR)

The linear system is solved by either dense or banded Gaussian elimination by routines from LINPACK [15].

3.4. Solution by iterative methods. For the preconditioned GMRES method, specify INFO(12) = 1. The user can specify some of the details in the linear system solution, namely MAXL (the number of iterations allowed before restarting), KMP (the number of vectors on which orthogonalization is done), NRMAX (the maximum number of restarts), and EPLI (the convergence tolerance for the linear iteration). Defaults for these constants are: MAXL = MIN(5,NEQ), KMP = MAXL (this corresponds to complete GMRES iteration, rather than the incomplete form), NRMAX = 2, and EPLI = 0.05. These defaults can be overridden by setting INFO(13) = 1 and following the instructions in the code documentation. Changing MAXL or KMP affects the amount of work storage required, as described in the code documentation.

With the GMRES method, the user must supply a subroutine PSOL which solves linear systems of the form $Px = b$, where P is the left preconditioner matrix. The subroutine PSOL has the following form.

SUBROUTINE PSOL (NEQ, T, Y, YPRIME, SAVR, WK, CJ,
WGHT, WP, IWP, B, EPLIN, IER, RPAR, IPAR)

The right-hand side vector b is in the B array on input, and PSOL must return the solution vector x in B. The Y, YPRIME, and SAVR arrays contain the current values of Y, YPRIME, and the residual F, respectively. The preconditioner matrix P is an approximation to the iteration matrix $(\mathrm{CJ})\partial F/\partial y' + \partial F/\partial y$, where CJ is a scalar ($= \alpha$) which is proportional to $1/h$ and is provided as input to PSOL and to JAC.

For the purposes of DASPK there are two types of preconditioners, depending on whether information about the iteration matrix is saved from one iteration or time step to the next. For example, in implementing an ILU preconditioner one would want to save factorization information from one iteration/time step to the next. To specify this type of preconditioner, set INFO(15) = 1; the user will then need to supply a subroutine JAC. There, an approximate iteration matrix would be formed, and then the ILU decomposition would be performed and stored. This work would all be done in the subroutine JAC. For the iterative methods option, the subroutine JAC has the following form.

SUBROUTINE JAC (RES, IRES, NEQ, T, Y, YPRIME, REWT,
SAVR, WK, H, CJ, WP, IWP, IER, RPAR, IPAR)

The arrays WP and IWP are real and integer work arrays which can be used for communication between the JAC routine and the PSOL routine. For example, one might store the ILU factorization in WP. For this preconditioner, the subroutine PSOL solves the linear system by back-substitution using the saved matrix data. For the ILU example, since this type of preconditioner is relatively expensive, one would like to save it and use it over many iterations and even many time steps. The strategy that DASPK uses to decide when to re-evaluate the preconditioner matrix is the same as the strategy DASSL uses [3] to decide when to re-evaluate its Jacobian, except that in addition a new preconditioner matrix is generated whenever there is a failure of the linear iteration.

The simpler type of preconditioner is one which does not make use of saved information. In this case, set INFO(15) = 0; then there is no need to supply a JAC routine. Examples of these types of preconditioners are diagonal scaling or matrix-free symmetric successive over-relaxation (SSOR), as described in [10].

One can get some idea about how well a given preconditioner is working by monitoring the following information: IWORK(16) contains the number of convergence test failures for the linear iteration, IWORK(20) contains the total number of linear iterations, IWORK(19) contains the total number of nonlinear iterations. One can compute the average dimension of the Krylov subspace by taking the ratio IWORK(20)/IWORK(19) of linear to nonlinear iterations. This gives an indication of how hard the iterative method is having to work to solve the part of the problem which is not being approximated well by the preconditioner. (Actually, this ratio can exceed the maximum dimension MAXL because of restarts; in fact, with up to 2 restarts, up to 3*MAXL linear iterations are allowed per nonlinear iteration.)

3.5. Higher-index DAEs. As in DASSL, there is always the possibility in DASPK that the problem specified does not have a well-defined solution, or has index higher than 1.

The index is a measure of the degree of singularity of the system; see [3]. Standard-form ODEs,

$$y' = f(t, y),$$

are index-0; ODEs with constraints

$$y' = f(x, y),$$
$$0 = g(x, y),$$

are index-1 if $\partial g/\partial x$ is nonsingular. Problems of index higher than 1 can cause difficulties for numerical methods, and in particular for the step size and order selection mechanisms in DASSL. A higher-index problem in DASSL will usually cause failures of the error tests and Newton convergence test (this information is available in IWORK(14) and IWORK(15)), and we recommend printing it out routinely on any successful or unsuccessful termination from DASSL for diagnostic purposes). A surprising number of problems are higher-index; for example, incompressible Navier–Stokes equations are index-2. It is possible to modify DASSL/DASPK to deal with higher-index systems, especially in the case of index-2. It is also possible to rewrite higher-index systems in lower-index forms which have the same analytical solution. For more information on this, see [3]. Either of these alternatives must be undertaken very carefully, because such modifications can sometimes affect the stability properties of the system or of the numerical method [2].

It is also possible to write problems in the form (1.1) for which there is no solution or no unique solution. If this is the case, what will probably happen is a termination with IDID = −8 (the matrix of partial derivatives is singular). In our experience, for large problems arising from the method of lines discretization of partial differential equations, this is usually due to an error in formulating the boundary conditions. In any case, a user who gets this message should either rethink the equations or else look for a bug.

4. Preconditioners for DAE systems. The choice of preconditioner can be critical in the performance of DASPK, as in other settings. We discuss here an approach to forming preconditioners for problems that arise from the semi-discretization of certain partial differential equations, following work that was done in [7] for the ODE context. Specifically, consider a mixed system of reaction-diffusion equations in some spatial region, in which some of the species (some PDE variables) obey time-dependent (evolution) equations, while the rest obey time-independent (quasi-steady) equations. We assume that some finite difference spatial discretization has been performed on all the equations, and that the resulting DAE system has index 1.

Let M denote the number of spatial mesh points, p the number of evolutionary species, and q the number of quasi-steady species. For practical purposes, it is usually best to order the variables by mesh point and then by species index, i.e., $p+q$ variables at mesh point 1, followed by $p+q$ variables at mesh point 2, etc. However, for ease of presentation, consider the "transpose" ordering—by species and then by mesh point. Thus if $u_i = (u_{i,1}, \ldots, u_{i,M})^T$ is the vector of species i variables at all mesh points, we take $y = (u_1, \ldots, u_p, u_{p+1}, \ldots, u_{p+q})^T$ as the DASPK vector. Each of the vectors $u_1, \ldots, u_p$ satisfies an ODE system in time, $u_i' = f_i(t, y)$, while each of $u_{p+1}, \ldots, u_{p+q}$ satisfies an algebraic system $0 = f_i(t, y)$. We assume that each f_i consists of a reaction term R_i that contains no spatial derivatives, and a diffusion (or more general transport) term S_i that involves no interaction among the species. Thus the system has the form

$$\begin{aligned} u_i' &= f_i = R_i(t, y) + S_i(t, u_i) \qquad (i \le p), \\ 0 &= f_i = R_i(t, y) + S_i(t, u_i) \qquad (i > p), \end{aligned}$$

in which $R_i = (R_{i,m})$ with $R_{i,m}$ depending only on the variables $u_{j,m}$ at mesh point m, and S_i depends on u_i but no other u_j. We can also write the DAE system as

$$0 = F(t, y, y') = (u_1' - f_1, \ldots, u_p' - f_p, -f_{p+1}, \ldots, -f_{p+q})^T. \tag{4.1}$$

If we let I_1 denote the identity matrix of order pM, and denote $f = (f_1, \ldots, f_{p+q})^T = R + S$, then the system Jacobian is

$$J = \alpha F_{y'} + F_y = \alpha \begin{pmatrix} I_1 & 0 \\ 0 & 0 \end{pmatrix} - \partial f/\partial y = \alpha \bar{I}_1 - \partial R/\partial y - \partial S/\partial y, \tag{4.2}$$

where $\bar{I}_1 \equiv \begin{pmatrix} I_1 & 0 \\ 0 & 0 \end{pmatrix}$.

For a preconditioner approximating J, there are two choices immediately available from this representation. First, if the reaction terms dominate the problem, then

$$P_R \equiv \alpha \bar{I}_1 - \partial R/\partial y \tag{4.3}$$

should be an effective choice. Here when the variables are ordered by mesh point and then species, as suggested initially, P_R is a block-diagonal matrix with M blocks each of size $p+q$, and so is very economical to deal with. But if the transport terms dominate the problem, consider instead

$$P_S \equiv \alpha \bar{I}_1 - \partial S/\partial y. \tag{4.4}$$

In this case, with the present ordering, P_S is block-diagonal with $p+q$ blocks of size M. Depending on the particular geometry, discretization, and mesh point ordering, each

$M \times M$ block of P_S should be amenable to standard solution methods for discrete elliptic PDE problems.

Following [7], preconditioners based on the idea of operator splitting can be formed that combine both of the above simpler choices. They are

$$P_{SR} \equiv (I - \alpha^{-1}\partial S/\partial y)(\alpha \bar{I}_1 - \partial R/\partial y), \quad \text{and} \tag{4.5}$$

$$P_{RS} \equiv (I - \alpha^{-1}\partial R/\partial y)(\alpha \bar{I}_1 - \partial S/\partial y). \tag{4.6}$$

The preconditioner P_{SR} represents the application of the reaction operator alone, followed by a correction involving the transport operator alone, and vice versa for P_{RS}. Each factor is much more efficiently treated than J itself, and so the same is true for P_{SR} and P_{RS}.

It is possible to do some analysis of the quality of these preconditioners, at least in the limit of small step size h, which corresponds to large α. Denoting

$$A = \partial S/\partial y = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix},$$

$$B = \partial R/\partial y = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix},$$

and $\epsilon = \alpha^{-1}$, we have

$$\begin{aligned} J &= \epsilon^{-1}\bar{I}_1 - A - B, \\ P_R &= \epsilon^{-1}(\bar{I}_1 - \epsilon B), \\ P_S &= \epsilon^{-1}(\bar{I}_1 - \epsilon A), \\ P_{SR} &= \epsilon^{-1}(I - \epsilon A)(\bar{I}_1 - \epsilon B), \quad \text{and} \\ P_{RS} &= \epsilon^{-1}(I - \epsilon B)(\bar{I}_1 - \epsilon A). \end{aligned}$$

We begin with a lemma.

LEMMA 4.1. *Assume B_{22} is nonsingular. Define the matrix C by*

$$C \equiv \begin{pmatrix} B_{11} & B_{12} \\ -B_{22}^{-1}B_{21}B_{11} & -B_{22}^{-1}B_{21}B_{12} \end{pmatrix}. \tag{4.7}$$

Then for $\epsilon\|C\| < 1$, we have

$$(\bar{I}_1 - \epsilon B)^{-1} = \epsilon^{-1}X_{-1} + X_0 + \epsilon X_1 + \cdots, \tag{4.8}$$

where

$$X_{-1} = \begin{pmatrix} 0 & 0 \\ 0 & -B_{22}^{-1} \end{pmatrix},$$

$$X_0 = \begin{pmatrix} I_1 & -B_{12}B_{22}^{-1} \\ -B_{22}^{-1}B_{21} & B_{22}^{-1}B_{21}B_{12}B_{22}^{-1} \end{pmatrix},$$

and $X_k = C^k X_0$ for $k \geq 1$.

Proof. We first note that

$$\begin{aligned} \bar{I}_1 - \epsilon B &= \begin{pmatrix} I_1 & 0 \\ 0 & \epsilon I_2 \end{pmatrix}\left[\begin{pmatrix} I_1 & 0 \\ -B_{21} & -B_{22} \end{pmatrix} - \epsilon \begin{pmatrix} B_{11} & B_{12} \\ 0 & 0 \end{pmatrix}\right] \\ &= \begin{pmatrix} I_1 & 0 \\ -\epsilon B_{21} & -\epsilon B_{22} \end{pmatrix}(I - \epsilon C), \end{aligned}$$

with C as given in (4.7). Thus, for $\epsilon\|C\| < 1$, we have

$$\begin{aligned}\left(\bar{I}_1 - \epsilon B\right)^{-1} &= (I - \epsilon C)^{-1} \begin{pmatrix} I_1 & 0 \\ -\epsilon B_{21} & -\epsilon B_{22} \end{pmatrix}^{-1} \\ &= (I + \epsilon C + \epsilon^2 C^2 + \cdots) \begin{pmatrix} I_1 & 0 \\ -B_{22}^{-1} B_{21} & -\epsilon^{-1} B_{22}^{-1} \end{pmatrix} \\ &\equiv (I + \epsilon C + \epsilon^2 C^2 + \cdots)(\epsilon^{-1} D + E).\end{aligned}$$

Hence,

$$\left(\bar{I}_1 - \epsilon B\right)^{-1} = \epsilon^{-1} D + (CD + E) + \sum_{k=1}^{\infty} \epsilon^k C^k (CD + E).$$

A quick calculation gives $X_{-1} = D$ and $X_0 = CD + E$, and this proves the lemma. □

We first consider the asymptotic behavior of the preconditioners P_R and P_{SR}. As will be seen below, the leading order term in the error for both preconditioners is the same. Hence, the order-ϵ terms for both preconditioners are needed to compare their overall effectiveness.

THEOREM 4.2. *Assume that B_{22} is nonsingular. Then for $\epsilon\|C\| < 1$, we have*

$$P_R^{-1} J - I = \begin{pmatrix} 0 & 0 \\ 0 & B_{22}^{-1} A_{22} \end{pmatrix} + \epsilon \begin{pmatrix} -A_{11} & B_{12} B_{22}^{-1} A_{22} \\ B_{22}^{-1} B_{21} A_{11} & -B_{22}^{-1} B_{21} B_{12} B_{22}^{-1} A_{22} \end{pmatrix} + O(\epsilon^2). \tag{4.9}$$

Proof. First note that $J - P_R = -A$. Hence,

$$\begin{aligned}P_R^{-1} J - I &= -P_R^{-1} A \\ &= -\left(X_{-1} + \epsilon X_0 + O(\epsilon^2)\right) A \\ &= -X_{-1} A - \epsilon X_0 A + O(\epsilon^2).\end{aligned}$$

A simple calculation for $X_{-1}A$ and $X_0 A$ now gives the result. □

This result means that, for small h at least, to the extent that $B_{22}^{-1} A_{22}$ is small in norm, then so is the error matrix $P_R^{-1} J - I$, and P_R is a good approximation to J for the purposes of preconditioning. Here B_{22} represents the Jacobian of the reaction of the quasi-steady species relative to themselves, while A_{22} represents the transport operator for those species.

THEOREM 4.3. *Assume that B_{22} is nonsingular. Then for $\epsilon \cdot \min\{\|C\|, \|A\|\} < 1$, we have*

$$\begin{aligned}P_{SR}^{-1} J - I \;&=\; \begin{pmatrix} 0 & 0 \\ 0 & B_{22}^{-1} A_{22} \end{pmatrix} \\ &+\epsilon \begin{pmatrix} 0 & B_{12} B_{22}^{-1} A_{22} \\ B_{22}^{-1} A_{22} B_{21} & B_{22}^{-1} A_{22} (A_{22} + B_{22}) - B_{22}^{-1} B_{21} B_{12} B_{22}^{-1} A_{22} \end{pmatrix} + O(\epsilon^2).\end{aligned} \tag{4.10}$$

Proof. From the definition of P_{SR}, a simple calculation gives

$$P_{SR} = J + \begin{pmatrix} 0 & 0 \\ 0 & A_{22} \end{pmatrix} + \epsilon AB.$$

Hence,

$$P_{SR}^{-1} J - I = -P_{SR}^{-1} \left(\bar{A}_{22} + \epsilon AB\right),$$

letting $\bar{A}_{22} \equiv \operatorname{diag}(0, A_{22})$. From the definition of P_{SR}, it is apparent that $P_{SR} = (I - \epsilon A) P_R$. Hence, for $\epsilon \cdot \min\{\|C\|, \|A\|\} < 1$ and B_{22} nonsingular, we have

$$P_{SR}^{-1} = \left(X_{-1} + \epsilon X_0 + O(\epsilon^2)\right)\left(I + \epsilon A + O(\epsilon^2)\right).$$

Therefore,

$$\begin{aligned} P_{SR}^{-1} J - I &= -\left(X_{-1} + \epsilon X_0 + O(\epsilon^2)\right)\left(I + \epsilon A + O(\epsilon^2)\right)\left(\bar{A}_{22} + \epsilon AB\right), \\ &= -X_{-1}\bar{A}_{22} - \epsilon\left[X_0 \bar{A}_{22} + X_{-1}\left(AB + \bar{A}_{22}^2\right)\right] + O(\epsilon^2). \end{aligned}$$

By an easy calculation, we find that $-X_{-1}\bar{A}_{22} = \operatorname{diag}(0, B_{22}^{-1}A_{22})$ and

$$-X_0\bar{A}_{22} - X_{-1}\left(AB + \bar{A}_{22}^2\right) = \begin{pmatrix} 0 & B_{12}B_{22}^{-1}A_{22} \\ B_{22}^{-1}A_{22}B_{21} & B_{22}^{-1}A_{22}(A_{22} + B_{22}) - B_{22}^{-1}B_{21}B_{12}B_{22}^{-1}A_{22} \end{pmatrix},$$

and this completes the proof. □

Comparing the upper left blocks in these two results, note that A_{11} *is* present in the order-ϵ term in (4.9), and *is not* in (4.10). Another difference is that all the blocks in the order-ϵ term of (4.10) have a factor of $B_{22}^{-1}A_{22}$, while the lower left block in (4.9) does not. Thus, while the leading expression in the error for both preconditioners is the same, we can expect P_{SR} to be a better preconditioner than P_R when either A_{11} is large or $B_{22}^{-1}A_{22}$ is small, in some relative sense. The apparent effects of these differences will be seen in the next section.

A result similar to (4.10) can be obtained for the preconditioner in which the two factors of P_{RS} are multiplied in the opposite order. The leading term in the error equation is the same, but the $O(\epsilon)$ term is not.

If instead we expect A to dominate B, we can consider the remaining two preconditioners P_S and P_{RS} in (4.4) and (4.6), respectively. The following theorem follows trivially along the same lines as above.

THEOREM 4.4. *Assume that A_{22} is nonsingular. Then for $\epsilon\|A_{11}\| < 1$, we have*

$$P_S^{-1} J - I = \begin{pmatrix} 0 & 0 \\ A_{22}^{-1}B_{21} & A_{22}^{-1}B_{22} \end{pmatrix} + O(\epsilon). \tag{4.11}$$

In addition, if $\epsilon \cdot \min\{\|A_{11}\|, \|B\|\} < 1$, then

$$P_{RS}^{-1} J - I = \begin{pmatrix} 0 & 0 \\ 0 & A_{22}^{-1}B_{22} \end{pmatrix} + O(\epsilon). \tag{4.12}$$

This result indicates that both P_S and P_{RS} are likely to be effective preconditioners when A_{22} dominates B_{21} and B_{22}. However, the fewer potential nonzeros in the leading order error term for P_{RS} suggests using it over P_S, all other things being equal. Moreover, if the two factors in P_{RS} are taken in the opposite order, one gets the same (larger) leading error term as for P_S, suggesting that the order of the factors in P_{RS} is to be preferred.

By departing from the operator splitting approach, it is possible to devise preconditioners P such that $P^{-1}J - I = O(\epsilon)$. However, the cost of applying P^{-1} is likely to be higher, and may be prohibitive on some problems. Along these lines, one can show the following result.

THEOREM 4.5. *Define $P_1 = (\epsilon^{-1}\tilde{I}_1 - \tilde{B})Q$, where*

$$\tilde{B} \equiv \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & I_2 \end{pmatrix} \quad \text{and} \quad Q \equiv \begin{pmatrix} I_1 & 0 \\ 0 & A_{22} + B_{22} \end{pmatrix}. \tag{4.13}$$

Then

$$P_1^{-1} J - I = O(\epsilon). \tag{4.14}$$

Next, define $P_2 = TP_1$, *where* $T = \mathrm{diag}(I_1 - \epsilon A_{11}, I_2)$. *Then*

$$P_2^{-1}J - I = O(\epsilon). \tag{4.15}$$

In the first case, the error in the preconditioner itself is

$$P_1 - J = \begin{pmatrix} A_{11} & B_{12} - B_{12}(A_{22} + B_{22}) \\ 0 & 0 \end{pmatrix}.$$

In the second case, the factor T is designed to remove a block in the error matrix, so that

$$P_2 - J = \begin{pmatrix} 0 & B_{12} - B_{12}(A_{22} + B_{22}) \\ 0 & 0 \end{pmatrix} + O(\epsilon).$$

Thus one would expect P_2 to perform better than P_1 when A_{11} is large. The dominant cost of inverting P_1 and P_2 above is most likely to be in solving linear systems involving $A_{22} + B_{22}$. Note that this matrix will be nonsingular for index-1 systems.

Often the numerical behavior of DASSL and its variants on constrained systems of this type is improved if the constraint equations in the DAE system are scaled. For index-m DAEs, the condition number of the iteration matrix is $O(\alpha^m)$ [3, p. 144]. Scaling is a way to reduce the dependence of the conditioning of a DAE system on the step size. In the case of the reaction-diffusion system, this would mean applying a scale factor σ to the lower qM equations in the system (4.1). For semi-explicit index-1 problems, the scale factor $\sigma = \alpha$ is suggested [3, p. 145]. If we define a scaling matrix

$$S = \begin{pmatrix} I_1 & 0 \\ 0 & \sigma I_2 \end{pmatrix}, \tag{4.16}$$

then the scaled problem is $\bar{F} \equiv SF = 0$. The choice of preconditioner for this problem corresponding to P for the unscaled problem is $\bar{P} = SP$. It is easy to see that the Jacobian of $\bar{F}$ is $\bar{J} = SJ$, and so $\bar{P}^{-1}\bar{J} = P^{-1}J$. Thus scaling does not affect the analytical properties of the method, but it does have the potential to improve its numerical properties by way of improved conditioning (roundoff error growth) of $\bar{J}$ and $\bar{P}$.

If scaling is applied to the preconditioner P_{SR} of (4.5), we can rewrite $\bar{P}_{SR}$ as

$$\begin{aligned} \bar{P}_{SR} &= S(I - \alpha^{-1}A)(\alpha\bar{I}_1 - B) \\ &= S(I - \alpha^{-1}A)S^{-1}S(\alpha\bar{I}_1 - B) \\ &= (I - \alpha^{-1}SAS^{-1})(\alpha S\bar{I}_1 - SB). \end{aligned}$$

But $SAS^{-1} = A$ because A is block-diagonal, and $S\bar{I}_1 = \bar{I}_1$. Thus we have

$$\bar{P}_{SR} = (I - \alpha^{-1}A)(\alpha\bar{I}_1 - SB). \tag{4.17}$$

This means that the addition of scaling to this preconditioner can be implemented simply by scaling the lower blocks of the B terms in the second factor of P_{SR}.

The result is not as simple if scaling is applied to the preconditioner P_{RS} of (4.6). Since $SBS^{-1} \neq B$, we obtain only

$$\bar{P}_{RS} = (I - \alpha^{-1}SBS^{-1})(\alpha\bar{I}_1 - SA). \tag{4.18}$$

Thus, in addition to scaling the lower blocks of the A in the second factor, we must scale the off-diagonal blocks of B to σB_{21} and $\sigma^{-1}B_{12}$ in the first factor.

5. Examples. In this section we will present several examples which illustrate how the method is applied and how well it works on various types of problems.

5.1. Two-dimensional heat equation. The first test problem is a two-dimensional heat equation given by

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \tag{5.1}$$

on a square $0 \le x, y \le 1$, with homogeneous boundary conditions ($u = 0$ on the boundary) and initial condition profile $u(x, y, 0) = 16x(1 - x)y(1 - y)$. We used a uniform Cartesian product mesh with L internal points in each direction, with spacings $\Delta x = \Delta y = 1/(L + 1)$. For

$$y_{jk} \approx u(j\Delta x, k\Delta y) \qquad (0 \le j, k \le L + 1),$$

the discrete problem is

$$\begin{cases} y'_{jk} = (\Delta x)^{-2}(y_{j+1,k} + y_{j-1,k} + y_{j,k+1} + y_{j,k-1} - 4y_{jk}), & 1 \le j, k \le L, \\ 0 = y_{jk} & \text{otherwise.} \end{cases} \tag{5.2}$$

Thus the total size of the system is NEQ = $(L + 2)^2$.

The DASPK vector y consists of the y_{jk} ordered first by j, then by k. The DAE system has the form

$$0 = F(t, y, y') = Ey' - By,$$

where E is the identity matrix with 0 in place of 1 for boundary indices, and B has the usual five-stripe structure.

The inclusion of the boundary conditions as additional algebraic equations makes this a DAE system and not an ODE system. For this problem, that is certainly not necessary, and perhaps less natural than forming ODEs at the L^2 internal mesh points. However, we choose this formulation here for the sake of illustration, and note that for some more compliciated boundary value problems there is a definite advantage to that choice.

The iteration matrix required in DASPK is

$$A = (\mathrm{CJ})F_{y'} + F_y = (\mathrm{CJ})E - B.$$

The preconditioner P is a tridiagonal difference-quotient approximation to A, obtained by the method of Curtis, Powell, and Reid [13], as used in DASSL. Columns whose indices differ by 3 are differenced simultaneously, producing a tridiagonal approximation at a cost of four F evaluations. However, it is not the tridiagonal part of the original matrix. Instead, the elements of the original matrix which do not lie in the tridiagonal band have been lumped, via the differencing, into the tridiagonal band. The preconditioner was computed and factorized in a subroutine JAC as described in §3.4, and the back-substitution was done in subroutine PSOL.

We solved the problem with the error tolerances RTOL = 0, ATOL = 10^{-3}, for three cases: $L = 5$, $L = 10$, $L = 20$, with both the direct and iterative options of DASPK. For the direct option, the iteration matrix was approximated by the tridiagonal matrix described above; hence it is a relatively poor but cheap approximation to the actual iteration matrix. Solution errors in all cases were comparable. The results are given in Table 5.1, where F denotes the number of function evaluations, PE is the number of preconditioner evaluations (in the direct method, this is the number of evaluations of the approximate iteration matrix), PS is the number of preconditioner solves (back-substitutions), NLI is the number of nonlinear (modified Newton) iterations, LI is the total number of linear iterations, AVL is the average number of Krylov

iterations per Newton iteration (NLI/LI), and NCF is the number of nonlinear convergence test failures. (There were no convergence test failures for the linear iteration.)

TABLE 5.1
Test results for the heat equation.

Method	L	Steps	F	PE	PS	NLI	LI	AVL	NCF
Iterative	5	45	220	17	169	87	82	0.94	0
Direct	5	98	513	102	0	207	0	0.00	28
Iterative	10	47	280	18	226	91	135	1.48	0
Direct	10	671	4443	985	0	1488	0	0.00	324
Iterative	20	51	449	17	398	100	298	2.98	0
Direct	20	1779	12111	2651	0	4158	0	0.00	878

The results indicate that the direct method (which gives identical results to DASSL), is having some trouble with convergence of the Newton iteration. This is not unexpected, considering that the approximation we are using to the iteration matrix is not very good. The iterative method, using (as a preconditioner) the same matrix approximation as the direct method, but in addition using GMRES to give a more accurate solution to the linear system, has no trouble achieving convergence in either the linear or nonlinear iterations, and gives a solution much more efficiently.

5.2. Multispecies food web problem. The next problem is a model of a multispecies food web [4], in which mutual competition and/or predator-prey relationships in a spatial domain are simulated. Here we consider a model with s species, where species $s/2+1, \ldots, s$ (the predators) have infinitely fast reaction rates:

$$\begin{cases} \dfrac{\partial c^i}{\partial t} = f_i(x, y, t, c) + d_i(c^i_{xx} + c^i_{yy}) & (i = 1, 2, \ldots, s/2), \\ 0 = f_i(x, y, t, c) + d_i(c^i_{xx} + c^i_{yy}) & (i = s/2+1, \ldots, s), \end{cases} \tag{5.3}$$

with

$$f_i(x, y, t, c) = c^i \left(b_i + \sum_{j=1}^{s} a_{ij} c^j \right). \tag{5.4}$$

The interaction and diffusion coefficients (a_{ij}, b_i, d_i) could be functions of (x, y, t) in general. The choices made for this test problem are for a simple model of p prey and p predator species $(s = 2p)$, arranged in that order in the vector c. We take the various coefficients to be as follows:

$$\begin{cases} a_{ii} = -1 & (\text{all } i), \\ a_{ij} = -0.5 \cdot 10^{-6} & (i \le p, j > p), \\ a_{ij} = 10^4 & (i > p, j \le p) \end{cases} \tag{5.5}$$

(all other $a_{ij} = 0$),

$$\begin{cases} b_i = (1 + \alpha x y + \beta \sin(4\pi x) \sin(4\pi y)) & (i \le p), \\ b_i = -(1 + \alpha x y + \beta \sin(4\pi x) \sin(4\pi y)) & (i > p) \end{cases} \tag{5.6}$$

and

$$\begin{cases} d_i = 1 \ (i \le p), \\ d_i = .05 \ (i > p). \end{cases} \tag{5.7}$$

The domain is the unit square $0 \le x, y \le 1$, and $0 \le t \le 10$. The boundary conditions are of Neumann type (zero normal derivatives) everywhere. The coefficients are such that

a unique stable equilibrium is guaranteed to exist when α and β are both zero, and time derivatives appear in the equations for species $s/2 + 1, \ldots, s$ [4]. Empirically, for (5.3) a stable equilibrium appears to exist when α and β are positive, although it may not be unique. In this problem we take $\alpha = 50$ and various values of β. The steady-state solution for the prey with $p = 1$ and $\beta = 300$ is represented in Fig. 5.1. The plot for the predator is identical except for a different scale on the vertical axis.

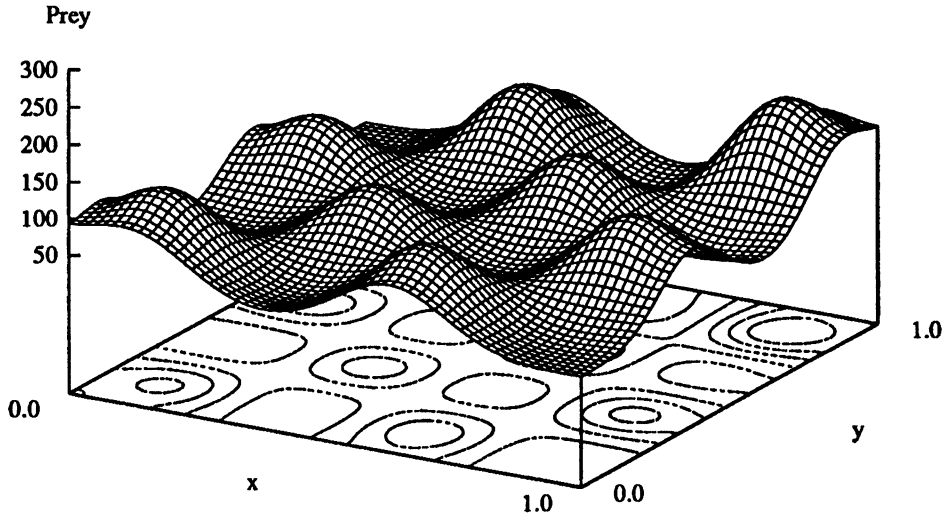


FIG. 5.1. *Food web problem. Plot of prey species at steady state* ($\beta = 300$).

The initial conditions used for this problem are taken to be simple peaked functions that satisfy the boundary conditions and very nearly satisfy the constraints, given by

$$c^i = 10 + i\,[16x(1-x)y(1-y)]^2 \qquad (i = 1, \ldots, s/2),$$

$$c^i = -\left(b_i + \sum_{j=1}^{s/2} a_{ij}c^j\right)/a_{ii} \qquad (i = s/2+1, \ldots, s).$$

The partial differential equation (PDE) system (5.3) (plus boundary conditions) was discretized with central differencing on an $L \times L$ mesh, much the same as for the first example. The resulting DAE system has size NEQ = sL^2.

This problem is of the form treated in §4, with $q = p$. Based on the ideas there, three different preconditioners were constructed. First, because the interaction terms f_i contain very large coefficients, we expect that P_R, given by (4.3), may be effective. Second, we consider the product P_{SR}, given by (4.5), and the same product taken in the opposite order, which we denote P'_{SR}. The inverse of the factor $(I - \alpha^{-1}\partial S/\partial y)$ representing the spatial part of the Jacobian is not computed exactly, but only approximated by a fixed number (namely five) of Gauss–Seidel iterations on the corresponding linear system.

To begin with, we compare the direct and Krylov solution modes for the case $\beta = 100$, $p = 1$, and $L = 20$. In the direct case, the Jacobian is treated as a banded matrix with half-bandwidths equal to $sL = 40$. In the Krylov case, we use the preconditioner P_{SR}. In order to measure global errors, a run in the direct mode was first made with tolerances RTOL = ATOL equal to 10^{-9}, and for each subsequent run with looser tolerances, the differences ΔY between the current and accurate solutions were taken at times $t = 10^{-7}, 10^{-4}, 10^{-1}, 3, 6, 9, 10$. A

weighted global error measure was computed as WGE = $\max[|\Delta Y_i|/(|Y_i|+1)]$, the maximum being over i and t. For the direct solution mode with RTOL = ATOL = 10^{-5}, the value of WGE is $2.5 \cdot 10^{-5}$. For the Krylov solution mode, values of RTOL = ATOL of 10^{-5}, 10^{-6}, and 10^{-7} gives WGE values of $1.4 \cdot 10^{-4}$, $4.3 \cdot 10^{-5}$, and $4.9 \cdot 10^{-6}$, respectively. Thus the Krylov run with 10^{-6} tolerances produces nearly the same accuracy as the direct run with 10^{-5} tolerances. Running on a Cray Y/MP computer, the cost of the two runs is nearly the same: 23.8 CPU sec in the direct case, and 23.3 CPU sec in the Krylov case. However, the storage requirements are far lower in the Krylov case, because of the use of a banded Jacobian in the direct case and a block-diagonal matrix in the Krylov case. For this pair of runs, the total number of real and integer words of work space is 104,910 in the direct case, and 16,931 in the Krylov case, a factor of 6.2 lower. For a finer mesh or larger number of species, the relative savings in storage would be even greater, and the cost advantage for the Krylov mode would also be greater.

Next, we show the results of DASPK runs in the Krylov mode for various values of β and L, still with $p = 1$, $s = 2$, and with tolerances of 10^{-5}. The runs were made on a Cray Y/MP. Table 5.2 gives, for each choice of β, L, and preconditioner, the total number of steps, the average number of Krylov iterations per Newton iteration AVL, and the CPU run time RT in sec. For $\beta = 100$ or 300, it is clear that P_R is inferior to the product preconditioners, whereas for $\beta = 1000$ it is superior. As expected, the less expensive preconditioner P_R always results in a higher average number of linear iterations per nonlinear iteration. The product order P_{SR} proved to perform better than P'_{SR} in all cases for which it was run.

TABLE 5.2
Test results for food web problem.

β	L	Prec.	Steps	AVL	RT
100	20	P_R	874	5.07	82.0
100	20	P_{SR}	198	1.32	14.9
100	40	P_{SR}	314	2.76	154
100	60	P_{SR}	320	2.69	350
300	20	P_R	989	4.88	90.2
300	20	P_{SR}	192	1.46	15.3
300	20	P'_{SR}	213	2.02	22.2
300	40	P_{SR}	200	1.15	54.6
300	40	P'_{SR}	226	1.56	75.2
300	60	P_{SR}	237	2.01	205
1000	20	P_R	188	1.94	8.9
1000	20	P_{SR}	219	1.49	17.7
1000	40	P_R	189	2.20	38.8
1000	40	P_{SR}	220	1.32	64.2
1000	60	P_R	205	2.86	113
1000	60	P_{SR}	198	1.61	147

One disturbing pattern in Table 5.2 is the unexpected growth of total cost as a function of L for fixed choice P_{SR} of preconditioner. We expect costs to grow as L^2, but for $\beta = 100$ the jump in run time as L is changed from 20 to 40 is anomalous, as is that for $\beta = 300$ and $L = 40$ and 60. While we have no complete explanation for this, we note that the higher costs are accompanied by a higher frequency of preconditioner evaluation, and suspect that they are related to errors in the value of $B = \partial R/\partial y$ incurred by evaluating this part of the Jacobian as infrequently as possible. Furthermore, by an analysis extending that in §4, it can be shown that relative errors in B are magnified in the error matrices $P_R^{-1}J - I$ and $P_{SR}^{-1}J - I$, by roughly 10^4, the coefficient in the lower left block of the array (a_{ij}).

The question of whether or not to apply scaling to the constraint equations arose in some of the tests on this problem, when the performance of the solver seemed to degrade at late times t. In some (but not all) such cases, scaling the constraints by α, as described in §4, improved the performance, by as much as 20% in run time. But for some cases the run time with scaling was larger than without it, with no apparent pattern for the cases in the two categories. For example, for the case $\beta = 100$, $L = 20$ with preconditioner P_{SR}, the run time is 14.9 sec without the rescaling and 12.2 sec with it. But the run with P_R on the same problem (the first case in Table 5.2) ran even slower with rescaling (92 sec vs 82 sec).

Finally, we present the results for one other case, the largest one in this test set: $\beta = 1000$, $L = 60$, $p = 7$, for which the problem size is NEQ = 50, 400. The run with preconditioner P_R and tolerances of 10^{-5} (as before) ran to completion in 471 sec on the Cray Y/MP, in 215 steps, with AVL = 2.75.

Acknowledgment. Most of the programming and debugging work in generating the DASPK solver from DASSL was done by Clement Ulrich. We are very grateful for his efforts.

REFERENCES

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

[2] U. ASCHER AND L. PETZOLD, *Stability of computational methods for constrained dynamics systems*, SIAM J. Sci. Comput., 14 (1993), pp. 95–120.

[3] K. BRENAN, S. CAMPBELL, AND L. PETZOLD, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier, New York, 1989.

[4] P. N. BROWN, *Decay to uniform states in food webs*, SIAM J. Appl. Math., 46 (1986), pp. 376–392.

[5] ———, *A theoretical comparison of the Arnoldi and GMRES algorithms*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 58–78.

[6] P. N. BROWN AND A. C. HINDMARSH, *Matrix-free methods for stiff systems of ODEs*, SIAM J. Numer. Anal., 23 (1986), pp. 610–638.

[7] ———, *Reduced storage matrix methods in stiff ODE systems*, J. Appl. Math. Comp., 31, (1989), pp. 40–91.

[8] P. N. BROWN AND Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 450–481.

[9] G. D. BYRNE, *Pragmatic experiments with Krylov methods in the stiff ODE setting*, in Computational Ordinary Differential Equations, J. R. Cash and I. Gladwell, eds., Oxford University Press, Oxford, 1992, pp. 323–356.

[10] T. F. CHAN AND K. R. JACKSON, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 533–542.

[11] ———, *The use of iterative linear equation solvers in codes for large systems of stiff IVPs for ODEs*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 378–417.

[12] 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.

[13] A. R. CURTIS, M. J. D. POWELL, AND J. K. REID, *On the estimation of sparse Jacobian matrices*, J. Inst. Math. Appl., 13 (1974), pp. 117–120.

[14] R. S. DEMBO, S. C. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[15] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER, AND G. W. STEWART, *LINPACK User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1979.

[16] C. W. GEAR, *Simultaneous numerical solution of differential/algebraic equations*, IEEE Trans. Circuit Theory, CT-18, 1 (1971), pp. 89–95.

[17] C. W. GEAR AND Y. SAAD, *Iterative solution of linear equations in ODE codes*, SIAM J. Sci. Stat. Comput., 4 (1983), pp. 583–601.

[18] K. R. JACKSON, *The use of Krylov subspace methods in codes for large stiff systems of ODEs*, in Advances in Computer Methods for Partial Differential Equations VI, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1987, pp. 452–459.

[19] B. LEIMKUHLER, L. R. PETZOLD, AND C. W. GEAR, *Approximation methods for the consistent initialization of differential-algebraic equations*, SIAM J. Numer. Anal., 28 (1991), pp. 205–226.

[20] L. PETZOLD, *A description of DASSL: A differential/algebraic system solver*, in Scientific Computing, IMACS Trans. Sci. Comput., R. S. Stepleman et al., eds., North-Holland, Amsterdam, 1983, pp. 65–68.

[21] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.

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

SIAM J. SCI. COMPUT.
Vol. 15, No. 6, pp. 1489–1505, November 1994

013

EFFICIENT SPECTRAL-GALERKIN METHOD I. DIRECT SOLVERS OF SECOND- AND FOURTH-ORDER EQUATIONS USING LEGENDRE POLYNOMIALS*

JIE SHEN†

Abstract. This paper presents some efficient algorithms based on the Legendre–Galerkin approximations for the direct solution of the second- and fourth-order elliptic equations. The key to the efficiency of these algorithms is to construct appropriate base functions, which lead to systems with sparse matrices for the discrete variational formulations. The complexities of the algorithms are a small multiple of N^{d+1} operations for a d-dimensional domain with $(N-1)^d$ unknowns, while the convergence rates of the algorithms are exponential for problems with smooth solutions. In addition, the algorithms can be effectively parallelized since the bottlenecks of the algorithms are matrix–matrix multiplications.

Key words. spectral-Galerkin method, Legendre polynomial, Helmholtz equation, biharmonic equation, direct solver

AMS subject classifications. 65N35, 65N22, 65F05, 35J05

1. Introduction.

This article is the first in a series for developing efficient spectral-Galerkin algorithms for elliptic problems. The spectral method employs global polynomials as the trial functions for the discretization of partial differential equations. It provides very accurate approximations with a relatively small number of unknowns. Consequently it has gained increasing popularity in the last two decades, especially in the field of computational fluid dynamics (see [11], [8], and the references therein).

The use of different test functions in a variational formulation leads to the three most commonly used spectral schemes, namely, the Galerkin, tau, and collocation versions. In the collocation method work is done in the physical space—a set of collocation points, while in the Galerkin and tau methods work is done in the spectral space—the coefficients of the polynomial series. The Galerkin and collocation methods usually lead to optimal error estimates, while the tau method, which is a modification of the Galerkin method, leads to nonsymmetric variational formulations and only suboptimal error estimates (see for instance [19] and [21]). In their pioneering book, Gottlieb and Orszag [11] presented an efficient Chebyshev-tau method. They also presented a basis for the Galerkin method that leads to full matrices; its application in practice is prohibited. It is surprising that virtually no effort has been made to construct appropriate bases (other than the Lagrangian interpolant basis) for the spectral-Galerkin method. The tau and collocation methods (the later being more natural for problems with variable coefficients) have been the focus of a great number of research papers (see [8] and the references therein), while the Galerkin method, which is more authentic and more accurate than the tau method, has drawn less attention. We should point out that the spectral element method, developed by Patera and his group, is in fact a spectral-Galerkin method (see the survey paper [13]). However, the spectral element method, in the case of a single domain, differs from the spectral-Galerkin method to be presented in this work in two aspects: (i) a Gaussian quadrature formula is used instead of exact integration, and (ii) a Lagrangian interpolant basis is used instead of the basis by simple combinations of Legendre polynomials.

This article presents appropriate bases for the Legendre–Galerkin method applied to second-order and fourth-order elliptic equations with various boundary conditions. The re-

*Received by the editors January 20, 1993; accepted for publication (in revised form) September 29, 1993. This work was supported in part by National Science Foundation grant DMS-9205300.

†Department of Mathematics, Penn State University, University Park, Pennsylvania 16802 (shen_j@math.psu.edu).

sulting discrete systems have sparse matrices similar to the ones obtained by finite difference discretizations. We shall develop efficient direct solution techniques for solving these discrete systems.

For second-order equations, the complexity of our algorithm is a small multiple of N^{d+1} operations for a d-dimensional domain with $(N-1)^d$ unknowns. Since its preprocessing time is negligible, it is definitely the method of choice among the existing spectral algorithms for solving one particular equation. For solving equations with multiple right-hand sides, it is as efficient as and more accurate than the Legendre-tau method; it is also very competitive with the Chebyshev-tau and Chebyshev-collocation methods. Furthermore, the roundoff errors in the Legendre–Galerkin approximation are considerably less appreciable than those in other spectral approximations.

Although the solution techniques for fourth-order equations obtained by finite difference methods are well developed (see [5] for a direct solver and [3] for a more efficient iterative solver), there are not many results available for the fourth-order equations obtained by spectral methods, which are more difficult to analyze and solve numerically. See the theoretical work in [10] and [2] for one-dimensional fourth-order equations and [1] for two-dimensional fourth-order equations. However, how to efficiently solve the resulting discrete systems is not clear, and only suboptimal error estimates are available for the collocation methods presented in [1].

We shall develop an efficient direct solver for the fourth-order equations based on the Legendre–Galerkin method. In the two-dimensional case, it has the complexity of a small multiple of $O(N^3)$ operations. In other words, we can solve fourth-order equations with an amount of work comparable to that involved when solving second-order equations. Hence, our algorithm should be clearly more efficient than the existing spectral methods for fourth-order equations (see for instance [24], [1]). Furthermore, it is clear that the Legendre–Galerkin approximation to the fourth-order equations leads to optimal error estimates. One of the applications of this algorithm is for solving the two-dimensional Stokes equations, since the stream function formulation of the two-dimensional Stokes equations is the two-dimensional biharmonic equation with Dirichlet boundary conditions. The bases developed here are also suitable for decomposing the solution into a high frequency part and a low frequency part, as is required in the implementation of the nonlinear Galerkin method [22].

A drawback of using Legendre polynomials is the lack of a fast transform between the physical space and the spectral space. Hence, in the second part of the series [20], we shall study the spectral-Galerkin method using the Chebyshev polynomials, for which the fast Fourier transform (FFT) is applicable. We hope that an efficient transform method for the Legendre polynomials can be eventually developed, at least for N sufficiently large, by using the ideas presented in [17] and [4].

The algorithms presented here are mostly appropriate for simple geometries and constant coefficient problems. We shall briefly address how to treat the variable coefficient case and leave the extension to more complex geometries for a later investigation. We should mention that the complexities of our algorithms are not optimal with respect to the number of unknowns. However, thanks to the spectral accuracy and the very small constants in front of the $O(N^{d+1})$ operation counts, our algorithms should be very competitive, at least for smooth functions or N not too large, with the high-precision algorithms with optimal or near-optimal complexity (cf. [18], [15], [16]).

The remainder of the article is organized as follows. In the next section we consider the second-order equations. In §3 we study the fourth-order equations. In §4 we point out some immediate extensions. Finally in §5 we present and compare some numerical results.

2. Second-order equations. In this section, we are interested in solving the Helmholtz equation

$$\alpha u - \Delta u = f \quad \text{in } \Omega = I^d, \qquad u|_{\partial\Omega} = 0, \tag{2.1}$$

where $I = (-1, 1)$ and $d = 1, 2$, or 3, by using the Legendre–Galerkin method. The extension to more general problems will be discussed in §4.

Let us first introduce some basic notation that will be used in the upcoming sections. We denote by $L_n(x)$ the nth degree Legendre polynomial, and we set

$$S_N = \text{span}\{L_0(x), L_1(x), \ldots, L_N(x)\}, \qquad V_N = \{v \in S_N : v(\pm 1) = 0\}.$$

Then the standard Legendre–Galerkin approximation to (2.1) is, find $u_N \in V_N^d$ such that

$$\alpha(u_N, v) + (\nabla u_N, \nabla v) = (f, v) \quad \forall v \in V_N^d, \tag{2.2}$$

where $(u, v) = \int_\Omega uv d\boldsymbol{x}$ is the scalar product in $L^2(\Omega)$; its norm will be denoted by $\|\cdot\|$. The approximation property of (2.2) is best described by using the Sobolev spaces. Let us denote $H^s(\Omega)$ to be the usual Sobolev spaces with the norm $\|v\|_s$. It is well known (cf. [8]) that for $\alpha \geq 0$, $s \geq 1$, and $u \in H^s(\Omega)$, the following optimal error estimate holds:

$$\|u - u_N\| + N\|u - u_N\|_1 \leq C(s)N^{-s}\|u\|_s. \tag{2.3}$$

Although approximation (2.2) is highly accurate, its practical value depends on the choice of a basis for V_N^d.

2.1. One-dimensional case. The crucial task is to choose an appropriate basis for V_N such that the linear system resulting from (2.2) is as simple as possible. The Lagrangian interpolant (or nodal) basis used in the collocation and spectral element formulations, and the basis

$$V_N = \text{span}\{\phi_2(x), \phi_3(x), \ldots, \phi_N(x)\}$$

with

$$\phi_k(x) = \begin{cases} L_k(x) - L_0(x), & k \text{ even}, \\ L_k(x) - L_1(x), & k \text{ odd}, \end{cases}$$

all lead to linear systems with full matrices even in the simplest case $\alpha = 0$. However, a clever choice of the basis would lead to a linear system with a sparse matrix.

We recall that the $\{L_n(x)\}$ satisfy the orthogonality relation

$$(L_k(x), L_j(x)) = \frac{2}{2k+1}\delta_{ij} \quad \forall k, j \geq 0, \tag{2.4}$$

and the recurrence relation

$$(2k+1)L_k(x) = L'_{k+1}(x) - L'_{k-1}(x). \tag{2.5}$$

We recall also that $L_n(x)$ is a polynomial of degree n and therefore $L''_n(x) \in S_{n-2}$, or more precisely,

$$L''_n(x) = \sum_{\substack{k=0 \\ k+n \text{ even}}}^{n-2} (k + \tfrac{1}{2})[n(n+1) - k(k+1)]L_k(x). \tag{2.6}$$

The following lemma is the key to the efficiency of our algorithms.

LEMMA 2.1. *Let us denote*

$$c_k = \frac{1}{\sqrt{4k+6}}, \qquad \phi_k(x) = c_k(L_k(x) - L_{k+2}(x)), \tag{2.7}$$
$$a_{jk} = (\phi_k'(x), \phi_j'(x)), \qquad b_{jk} = (\phi_k(x), \phi_j(x)),$$

then

$$a_{jk} = \begin{cases} 1, & k = j \\ 0, & k \neq j, \end{cases} \qquad b_{jk} = b_{kj} = \begin{cases} c_k c_j(\frac{2}{2j+1} + \frac{2}{2j+5}), & k = j, \\ -c_k c_j \frac{2}{2k+1}, & k = j+2, \\ 0 & \textit{otherwise}, \end{cases} \tag{2.8}$$

and

$$V_N = \text{span}\{\phi_0(\text{x}), \phi_1(\text{x}), \ldots, \phi_{\text{N}-2}(\text{x})\}.$$

Proof. Since $L_k(\pm 1) = (\pm 1)^k$, it follows that $\phi_k(x) \in V_N$ for $k = 0, 1, \ldots, N-2$. On the other hand, it is clear that $\{\phi_k(x)\}$ are linear independent and that $\dim(V_N) = N-1$. Hence,

$$V_N = \text{span}\{\phi_0(x), \phi_1(x), \ldots, \phi_{N-2}(x)\}.$$

Notice that

$$a_{jk} = (\phi_k'(x), \phi_j'(x)) = -(\phi_k''(x), \phi_j(x)) = -(\phi_k(x), \phi_j''(x)).$$

Using (2.4)–(2.5), one can easily derive (2.8). □

Remark 2.1. It is transparent that base functions similar to that in (2.7) can be constructed using Chebyshev or other Jacobi polynomials. However, whether they would also lead to efficient algorithms warrants further investigation because of the nonuniform weight in their orthogonal relations (see [20]).

In the following sections, we shall use capital letters to denote matrices or two-dimensional arrays, and boldface letters to denote column vectors.

It is now clear that (2.2) (with $d = 1$) is equivalent to

$$\alpha(u_N, \phi_k(x)) + (u_N', \phi_k'(x)) = (f, \phi_k(x)), \qquad k = 0, 1, \ldots, N-2. \tag{2.9}$$

Let us denote

$$f_k = (f, \phi_k(x)), \qquad \boldsymbol{f} = (f_0, f_1, \ldots, f_{N-2})^T;$$

$$u_N = \sum_{n=0}^{N-2} v_n \phi_n(x), \qquad \boldsymbol{v} = (v_0, v_1, \ldots, v_{N-2})^T, \quad \text{and } B = \big(b_{kj}\big)_{0 \le k, j \le N-2}. \tag{2.10}$$

Then (2.9) is equivalent to the following matrix equation:

$$(\alpha B + I)\mathbf{v} = \boldsymbol{f}. \tag{2.11}$$

Since $b_{kj} = 0$ for $k \neq j$ and $k \neq j \pm 2$, we observe that B (respectively, the system (2.11) with $\alpha \neq 0$) can be decoupled into two tridiagonal submatrices (respectively, two tridiagonal

subsystems for the odd and even components of $\mathbf{v}$). Notice also that the system (2.11) reduces to a diagonal system for $\alpha = 0$.

Remark 2.2. The equation

$$\alpha u + \beta u_x - u_{xx} = f, \qquad u(\pm 1) = 0 \tag{2.12}$$

can be easily handled as well. In fact let $c_{jk} = (D_x\phi_k(x), \phi_j(x))$ and $C = (c_{kj})$. A simple computation leads to

$$c_{jk} = -c_{kj} = \begin{cases} 2c_k c_j, & k = j+1, \\ 0 & \text{otherwise.} \end{cases} \tag{2.13}$$

Hence, the discrete system corresponding to (2.12) is

$$(\alpha B + \beta C + I)\boldsymbol{v} = \boldsymbol{f},$$

which is simply a pentadiagonal system.

2.2. Two-dimensional case. It is clear that

$$V_N^2 = \operatorname{span}\{\phi_i(x)\phi_j(y) : i, j = 0, 1, \ldots, N-2\}.$$

Let us denote

$$u_N = \sum_{k,j=0}^{N-2} u_{kj}\phi_k(x)\phi_j(y), \qquad f_{kj} = (f, \phi_k(x)\phi_j(y)),$$

and

$$U = (u_{kj})_{k,j=0,1,\ldots,N-2}, \qquad F = (f_{kj})_{k,j=0,1,\ldots,N-2}.$$

Taking $v = \phi_l(x)\phi_m(y)$ in (2.2) for $l, m = 0, 1, \ldots, N-2$, we find that (2.2) is equivalent to the following matrix equation:

$$\alpha BUB + UB + BU = F, \tag{2.14}$$

where B are the matrices defined in (2.10).

This equation can be solved in particular by the matrix decomposition method described in [7], which is better known in the field of spectral methods as the matrix diagonalization method [12]. To this end, let Λ be the diagonal matrix whose diagonal entries $\{\lambda_p\}$ are the eigenvalues of B, and let E be the orthonormal matrix formed by the eigenvectors of B (since B is symmetric), i.e., $E^T B E = \Lambda$. Now setting $U = EV$, equation (2.14) becomes

$$\alpha E\Lambda V B + EVB + E\Lambda V = F.$$

Multiplying E^T to the above equation gives

$$\alpha\Lambda V B + VB + \Lambda V = E^T F \equiv G.$$

The transpose of the above equation reads

$$\alpha BV^T\Lambda + BV^T + V^T\Lambda = G^T. \tag{2.15}$$

Let $\boldsymbol{v}_p = (v_{p0}, v_{p1}, \ldots, v_{pN-2})^T$ and $\boldsymbol{g}_p = (g_{p0}, g_{p1}, \ldots, g_{pN-2})^T$ for $p = 0, 1, \ldots, N-2$. Then the pth column of equation (2.15) can be written as

$$\left((\alpha\lambda_p + 1)B + \lambda_p I\right)\boldsymbol{v}_p = \boldsymbol{g}_p, \qquad p = 0, 1, \ldots, N-2, \tag{2.16}$$

which is equivalent to $N-1$ one-dimensional equations of the form (2.11).

In summary, the solution of (2.14) consists of the following steps:

(1) Preprocessing: compute the eigenvalues and eigenvectors of B;
(2) Compute $G = E^T F$;
(3) Obtain V by solving (2.16);
(4) Set $U = EV$.

Since B can be split into two symmetric tridiagonal submatrices, the eigenvalues and eigenvectors of B can be easily computed in $O(N^2)$ operations. Step 3 consists of solving $N-1$ tridiagonal systems of order $N-1$. From the structure of B, we realize that $e_{kj} = 0$ for $k+j$ odd. Hence the amount of work for the matrix multiplications can be reduced by a factor of 2. Consequently steps 2 and 4 take a total of $2(N-1)^3$ arithmetic operations. Therefore, for each right-hand side, the system (2.14) can be solved in about $2N^3$ operations.

We remark that for each particular solution of (2.1), the Legendre–Galerkin method described above takes at most the same amount of operations as the Legendre-tau method. However, thanks to the symmetry of the Galerkin procedure, the complexity of the preprocessing stage in the Galerkin case is an order of magnitude less (in terms of N) than in the tau (and the collocation) case. Furthermore, the error estimate of the Galerkin approximation is optimal (see (2.3)), while that of the tau approximation is not (see for instance [21] and [19]).

Remark 2.3. As in Remark 2.2, equation (2.1) with a first-order term βu_y can be efficiently treated. In this case, the discrete equation (2.14) should be replaced by

$$\alpha BUB + \beta BUC + UB + BU = F,$$

which can still be solved by the matrix diagonalization method. By the same token, we can treat equation (2.1) with a first-order term βu_x.

Remark 2.4. We note that the bottleneck of the algorithm is the two matrix–matrix multiplications in steps 2 and 4, which can be effectively parallelized and their complexity improved to $O(N^{d+1}/p)$ with p parallel processors. The same remark holds for other algorithms to be presented later in this paper.

2.3. Three-dimensional case. The three-dimensional Helmholtz equation can also be efficiently solved by the above matrix diagonalization method. However, the formulation of the algorithm requires a little extra care. Let us denote

$$u_N = \sum_{n,m,l=0}^{N-2} u_{nml}\phi_n(x)\phi_m(y)\phi_l(z), \qquad f_{ijk} = (f, \phi_i(x)\phi_j(y)\phi_k(z)).$$

Since

$$V_N^3 = \operatorname{span}\{\phi_i(x)\phi_j(y)\phi_k(z) : i, j, k = 0, 1, \ldots, N-2\},$$

taking $v = \phi_i(x)\phi_j(y)\phi_k(z)$ in (2.2) for $i, j, k = 0, 1, \ldots, N-2$, we find that (2.2) with $d = 3$ is equivalent to the following equation:

$$\alpha b_{in}u_{nml}b_{jm}b_{kl} + u_{iml}b_{jm}b_{kl} + b_{in}u_{njl}b_{kl} + b_{in}u_{nmk}b_{jm} = f_{ijk}, \quad i, j, k = 0, 1, \ldots, N-2, \tag{2.17}$$

where we have used the conventional notation such that a pair of repeated index implies a summation of the index from 0 to $N-2$. Hence, by the definition of E and Λ in §2.2, we have

$$b_{in}e_{nq} = \lambda_q e_{iq}, \qquad e_{iq}e_{ip} = \delta_{qp}.$$

Setting $u_{nml} = e_{nq} v_{qml}$ and using the above relation, we can rewrite equation (2.17) as

$$\alpha\lambda_q e_{iq} v_{qml} b_{jm} b_{kl} + e_{iq} v_{qml} b_{jm} b_{kl} + \lambda_q e_{iq} v_{qjl} b_{kl} + \lambda_q e_{iq} v_{nmk} b_{jm} = f_{ijk},$$
$$i, j, k = 0, 1, \ldots, N-2.$$

Multiplying e_{ip} to the above equation, we obtain

$$(\alpha\lambda_p + 1) v_{pml} b_{jm} b_{kl} + \lambda_p (v_{pjl} b_{kl} + v_{pmk} b_{jm}) = e_{ip} f_{ijk} \equiv g_{pjk}, \quad p, j, k = 0, 1, \ldots, N-2.$$

Now set $V^p = (v_{pml})_{0 \le m,l \le N-2}$ and $G^p = (g_{pml})_{0 \le m,l \le N-2}$. We can rewrite the above equation as

$$(\alpha\lambda_p + 1) B V^p B + \lambda_p (V^p B + B V^p) = G^p, \qquad p = 0, 1, \ldots, N-2. \tag{2.18}$$

For each p, the above equation corresponds to a two-dimensional equation of the form (2.14).

In summary, the solution of (2.17) consists of the following steps:

(1) Preprocessing: compute the eigenvalues and eigenvectors of B;
(2) Compute $g_{pjk} = e_{ip} f_{ijk}$ for $p, j, k = 0, 1, \ldots, N-2$;
(3) Obtain V^p by solving (2.18) for $p = 0, 1, \ldots, N-2$;
(4) Set $u_{nml} = e_{nq} v_{qml}$ for $n, m, l = 0, 1, \ldots, N-2$.

Step 3 consists of solving $N-1$ two-dimensional equations of the form (2.14). Hence, it takes about $2N^4$ operations. Steps 2 and 4 take $2N^4$ operations. Therefore, the system (2.17) can be solved in about $4N^4$ operations.

3. Fourth-order equations. In this section, we consider the fourth-order equation

$$\alpha u - \beta \Delta u + \Delta^2 u = f \quad \text{in } \Omega = I^d, \qquad u|_{\partial\Omega} = \frac{\partial u}{\partial \boldsymbol{n}}|_{\partial\Omega} = 0, \tag{3.1}$$

where $\boldsymbol{n}$ is the normal vector to $\partial\Omega$.

Let

$$W_N = \{v \in S_N : v(\pm 1) = v_x(\pm 1) = 0\}.$$

Then the Legendre–Galerkin approximation of (3.1) consists of finding $u_N \in W_N^d$ such that

$$\alpha(u_N, v) + \beta(\nabla u_N, \nabla v) + (\Delta u_N, \Delta v) = (f, v) \quad \forall v \in W_N^d. \tag{3.2}$$

It can be shown that for $\alpha, \beta > 0$ and $u \in H^s(\Omega) \cap H_0^2(\Omega)$ for $s \ge 2$, then the following optimal error estimate holds:

$$\|u - u_N\| + N\|u - u_N\|_1 + N^2\|u - u_N\|_2 \le C(s) N^{-s} \|u\|_s. \tag{3.3}$$

3.1. One-dimensional case. Equation (3.1) in the one-dimensional case can serve as a model for the *clamped rod* problem. It can also serve as a model for the two-dimensional Stokes equations in the primitive variable formulation or in the stream function formulation, subject to the periodic boundary condition in the y direction and Dirichlet boundary condition in the x direction. A semi-implicit time discretization of the important Kuramoto–Sivashinsky equation modeling a flame propagation [23] is also of the form (3.1) with $d = 1$.

LEMMA 3.1. *Let*

$$d_k = \frac{1}{\sqrt{2(2k+3)^2(2k+5)}}$$

and

$$\psi_k(x) = d_k(L_k(x) - \frac{2(2k+5)}{2k+7}L_{k+2}(x) + \frac{2k+3}{2k+7}L_{k+4}(x)), \quad k = 0, 1, \ldots, N-4. \tag{3.4}$$

Then

$$W_N = \text{span}\{\psi_0(x), \psi_1(x), \ldots, \psi_{N-4}(x)\}.$$

Furthermore, we have

$$a_{kj} = (\psi_j''(x), \psi_k''(x)) = \begin{cases} 1, & k = j, \\ 0, & k \neq j, \end{cases} \tag{3.5}$$

and the only nonzero elements of $b_{kj} = (\psi_j(x), \psi_k(x))$, $c_{kj} = (\psi_j'(x), \psi_k'(x))$ *are*

$$\begin{aligned} b_{kk} &= d_k^2(e_k + h_k^2 e_{k+2} + g_k^2 e_{k+4}), \\ b_{kk+2} &= b_{k+2k} = d_k d_{k+2}(h_k e_{k+2} + g_k h_{k+2} e_{k+4}), \\ b_{kk+4} &= b_{k+4k} = d_k d_{k+4} g_k e_{k+4}, \\ c_{kk} &= -2(2k+3)d_k^2 h_k, \\ c_{kk+2} &= c_{k+2k} = -2(2k+3)d_k d_{k+2}, \end{aligned} \tag{3.6}$$

where

$$e_k = \frac{2}{2k+1}, \quad g_k = \frac{2k+3}{2k+7}, \quad h_k = -(1+g_k).$$

Proof. Since $L_k(\pm 1) = (\pm 1)^k$ and $L_k'(\pm 1) = \frac{1}{2}(\pm 1)^{k-1}k(k+1)$, we can readily check that $\psi_k(x) \in W_N$. On the other hand, it is clear that $\{\psi_i(x)\}$ are linear independent and the dimension of W_N is $N-3$. Hence

$$W_N = \text{span}\{\psi_0(x), \psi_1(x), \ldots, \psi_{N-4}(x)\}.$$

(3.5) and (3.6) can be derived by direct computations using (2.4)–(2.6) and integration by parts. □

To simplify the notation, we denote hereafter $q = N-4$. Let us denote now

$$\begin{aligned} B &= (b_{kj})_{0\le k,j\le q}, & C &= (c_{kj})_{0\le k,j\le q}, \\ f_k &= (f, \psi_k(x)), & \boldsymbol{f} &= (f_0, f_1, \ldots, f_q)^T, \\ u_N &= \textstyle\sum_{n=0}^q v_n \psi_n(x), & \boldsymbol{v} &= (v_0, v_1, \ldots, v_q)^T. \end{aligned}$$

We find that (3.2) with $d = 1$ is equivalent to the following matrix equation:

$$(\alpha B + \beta C + I)\boldsymbol{v} = \boldsymbol{f}. \tag{3.7}$$

It is obvious that B and C are symmetric positive definite matrices. Furthermore, B can be split into two pentadiagonal submatrices and C can be split into two tridiagonal submatrices. Hence, the system can be efficiently solved.

3.2. Two-dimensional case. Equation (3.1) in the two-dimensional case with $\alpha = \beta = 0$ is the well-known biharmonic equation. It has many important applications. In particular, it is known as a model for the plate problem; the steam-function of a flow governed by two-dimensional Stokes equations is also a solution of equation (3.1). The case with $\alpha \neq 0$ is also important for time-dependent problems.

It is obvious that

$$W_N^2 = \operatorname{span}\{\psi_i(x)\psi_j(y) : i, j = 0, 1, \ldots, q\}.$$

Using the same notations as in §2.2 (with $\phi_k(x)$ replaced by $\psi_k(x)$, and $N-2$ replaced by q), taking $v = \psi_l(x)\psi_m(y)$ in (3.2) for $l, m = 0, 1, \ldots, q$, we find that (3.2) is equivalent to the following matrix equation:

$$\alpha BUB + \beta(CUB + BUC) + BU + 2CUC + UB = F. \tag{3.8}$$

We can also rewrite the above equation in the following form using the tensor product notation:

$$D\boldsymbol{u} \equiv (\alpha B \otimes B + \beta(C \otimes B + B \otimes C) + B \otimes I + 2C \otimes C + I \otimes B)\boldsymbol{u} = \boldsymbol{f}, \tag{3.9}$$

where $\boldsymbol{f}$ and $\boldsymbol{u}$ are F and U written in the form of a column vector, i.e.,

$$\boldsymbol{f} = (f_{00}, f_{10}, \ldots, f_{q0}; f_{01}, \ldots, f_{q1}; \ldots; f_{0q}, \ldots, f_{qq})^T, \tag{3.10}$$

and $\otimes$ denotes the tensor product of matrices, i.e., $A \otimes B = (Ab_{ij})_{i,j=0,1,\ldots,q}$.

Let us first remark that if $BC = CB$, then equation (3.8) can be efficiently solved using the matrix diagonalization method (see below). But unfortunately, due to the fact that equation (3.1) is not separable, we have $BC \neq CB$. However, $BC - CB$ only has eight nonzero entries at positions $(0, 2)$, $(1, 3)$, $(q-3, q-1)$, $(q-2, q)$, and their symmetric counterpart $(2, 0)$, $(3, 1)$, $(q-1, q-3)$, $(q, q-2)$. These nonzero entries can be eliminated by modifying the following four entries of the matrix B:

$$\begin{aligned}
b_{00} &\longrightarrow \tilde{b}_{00} = (c_{00}b_{02} + c_{02}b_{22} - b_{02}c_{22} - b_{04}c_{42})/c_{02},\\
b_{11} &\longrightarrow \tilde{b}_{11} = (c_{11}b_{13} + c_{13}b_{33} - b_{13}c_{33} - b_{15}c_{53})/c_{13},\\
b_{q-1q-1} &\longrightarrow \tilde{b}_{q-1q-1} = (c_{q-1q-1}b_{q-1q-3} + c_{q-1q-3}b_{q-3q-3}\\
&\qquad - b_{q-1q-3}c_{q-3q-3} - b_{q-1N-9}c_{N-9q-3})/c_{q-1q-3},\\
b_{qq} &\longrightarrow \tilde{b}_{qq} = (c_{qq}b_{qq-2} + c_{qq-2}b_{q-2q-2} - b_{qq-2}c_{q-2q-2} - b_{qq-4}c_{q-4q-2})/c_{qq-2}.
\end{aligned}$$

Setting $\tilde{B} = (\tilde{b}_{ij})$ with $\tilde{b}_{ij} = b_{ij}$ for $(i, j) \neq (0, 0)$, $(1, 1)$, $(q-1, q-1)$, (q, q), we have

$$\tilde{B}C = C\tilde{B}.$$

Now let us explain how to efficiently solve the equation

$$\alpha \tilde{B}UB + \beta(CUB + \tilde{B}UC) + \tilde{B}U + 2CUC + UB = F. \tag{3.11}$$

Since $\tilde{B}$ and C are symmetric and $\tilde{B}C = C\tilde{B}$, it is well known that there exists an orthonormal matrix E such that

$$\tilde{B}E = E\Lambda, \qquad CE = E\Sigma, \tag{3.12}$$

and Λ and Σ are real diagonal matrices. The matrix E is the set of eigenvectors of $\tilde{B}$ and C, the diagonal entries $\{\lambda_p\}$ and $\{\sigma_p\}$ of Λ and Σ are, respectively, eigenvalues of $\tilde{B}$ and C. Hence, setting $U = EV$ in (3.8) and using (3.12), we obtain

$$\alpha E\Lambda VB + \beta E(\Sigma VB + \Lambda VC) + E\Lambda V + 2E\Sigma VC + EVB = F.$$

Multiplying E^T to the above equation, we get

$$\alpha\Lambda VB + \beta(\Sigma VB + \Lambda VC) + \Lambda V + 2\Sigma VC + VB = E^T F \equiv G. \tag{3.13}$$

As in §2.2, let $\boldsymbol{v}_p = (v_{p0}, v_{p1}, \ldots, v_{pq})^T$ and $\boldsymbol{g}_p = (g_{p0}, g_{p1}, \ldots, g_{pq})^T$ for $p = 0, 1, \ldots, q$. Then the pth row of equation (3.13) is

$$\left\{(\alpha\lambda_p + \beta\sigma_p + 1)B + (\beta\lambda_p + 2\sigma_p)C + \lambda_p I\right\}\boldsymbol{v}_p = \boldsymbol{g}_p, \qquad p = 0, 1, \ldots, q. \tag{3.14}$$

For each p, the above equation is simply a one-dimensional equation of the form (3.7).

In summary, the solution of (3.11) consists of the following steps:

(1) Preprocessing: compute the eigenvalues and eigenvectors Λ, Σ, E of $\tilde{B}$ and C;
(2) Compute $G = E^T F$;
(3) Obtain $\boldsymbol{v}_p$ by solving (3.14);
(4) Set $U = EV$.

Using a classical result on tridiagonal matrices (see §3.7 of [14]), we derive that the eigenvalues of C are all distinct. Consequently an eigenvector of C is automatically an eigenvector of $\tilde{B}$. Therefore for the preprocessing stage, we only have to compute the eigenvalues and the eigenvectors Σ and E of C; Λ can then be determined by the relation $\tilde{B}E = E\Lambda$. Since C can be split into two tridiagonal submatrices, the preprocessing stage only takes $O(N^2)$ operations. As before, steps 2 through 4 take about $2N^3$ operations.

Now let us describe how to use the fast solver for (3.11) to solve the original system (3.8) or equivalently (3.9). To this end, we rewrite (3.11) by using the tensor product notation

$$\tilde{D}u \equiv \left(\alpha\tilde{B}\otimes B + \beta(C\otimes B + \tilde{B}\otimes C) + \tilde{B}\otimes I + 2C\otimes C + I\otimes B\right)\boldsymbol{u} = \boldsymbol{f}. \tag{3.15}$$

It is easy to see that D and $\tilde{D}$ differ at only $4(q+1)$ rows; more precisely, they differ at the rows $(i-1)*q+1$, $(i-1)*q+2$, $i*q-1$, $i*q$ for $i = 1, 2, \ldots, q$. Following the idea in [6], we can use the fast solver for (3.15) to solve equation (3.9) by the method of capacitance matrix. We briefly describe the method of capacitance matrix below.

Without loss of generality, we assume that the first p rows of D are changed to obtain $\tilde{D}$. Although this is not the case here, the same result can be achieved by using an implicit indexing scheme. Partition D, $\tilde{D}$, and $\boldsymbol{f}$ in the form

$$D = \begin{pmatrix} D_1 \\ D_2 \end{pmatrix}, \quad \tilde{D} = \begin{pmatrix} \tilde{D}_1 \\ D_2 \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \end{pmatrix},$$

where D_1 and $\tilde{D}_1$ are $p \times r$ matrices, with $r = (q+1)^2$ the order of the matrix D, and $\boldsymbol{f}_1$ a vector of length p. Then equation (3.9) can be solved as follows:

(1) Preprocessing: compute the $p \times p$ capacitance matrix

$$P = D_1\tilde{D}^{-1}\begin{pmatrix} I \\ 0 \end{pmatrix}, \tag{3.16}$$

where I is the $p \times p$ identity matrix; factor P into a LU form;

(2) Compute $\boldsymbol{v}_1 = D_1 \tilde{D}^{-1} \boldsymbol{f}$;
(3) Solve $P\mathbf{w}_1 = \boldsymbol{f}_1 - \boldsymbol{v}_1$;
(4) Solve $\tilde{D}\boldsymbol{u} = \boldsymbol{f} + \binom{\boldsymbol{w}_1}{0}$.

One can check that u is in fact the solution of the equation $D\boldsymbol{u} = \boldsymbol{f}$.

It is clear that steps 2 through 4 can be performed in $4N^3 + O(N^2)$ operations. The preprocessing stage, i.e., the computation of the capacitance matrix, is a little tricky, however. As explained in [5], one should take advantage of the sparseness of D_1 and $\binom{I}{0}$ in the construction of P so that the preprocessing stage can be done in $O(pr + p^3) = O(N^3)$ operations. Refer to [5] for more details of this aspect. However, unlike in the finite difference case [5], the capacitance matrix is no longer symmetric positive definite. We can write $D = \tilde{D} + \Delta$ with

$$\Delta = \begin{pmatrix} E, 0, \cdots, 0 \\ 0, E, \cdots, 0 \\ \vdots \; \vdots \\ 0, \cdots, 0, E \end{pmatrix},$$

where E is a diagonal matrix of order q with four nonzero entries at positions $(1, 1)$, $(2, 2)$, $(q-1, q-1)$, and (q, q). Hence, $\tilde{D}^{-1}D = I + \tilde{D}^{-1}\Delta$ and

$$P = I + \bar{\Delta}\bar{D}, \tag{3.17}$$

where $\bar{\Delta}$ is the diagonal matrix with the nonzero entries of Δ and $\bar{D}$ is the part of $\tilde{D}^{-1}$ corresponding to the entries in positions: $(i, j) : j = 0, 1, \ldots, q;\ i = 0, 1, q-1, q$. $\tilde{D}$ is symmetric positive definite and so is $\bar{D}$. In fact, all the nonzero entries in Δ are positive. Hence P is not symmetric but it is similar to the symmetric positive definite matrix $\bar{\Delta}^{-\frac{1}{2}} + \bar{\Delta}^{\frac{1}{2}}\bar{D}\bar{\Delta}^{\frac{1}{2}}$. Furthermore, we note that D and $\tilde{D}$ have alternating zero and nonzero elements, hence the system (3.9) (respectively, the capacitance matrix P) can be decoupled into four subsystems (respectively, four submatrices of order q or $q+1$). This would reduce the cost of preprocessing by a factor of 16.

Three-dimensional fourth-order equations can also be treated by the above procedure.

4. Miscellaneous extensions. The Legendre–Galerkin method described here can be applied to more general problems. In this section, we describe several immediate extensions.

4.1. Robin-type boundary conditions. When other boundary conditions are prescribed, it is necessary to construct a basis incorporating the boundary conditions. For the sake of simplicity, we only consider the one-dimensional equation

$$\alpha u - u_{xx} = f \quad \text{in } I$$

with the Robin-type boundary condition

$$a_{\pm}u(\pm 1) + b_{\pm}u_x(\pm 1) = 0.$$

Let $W_N = \{v \in S_N : a_{\pm}v(\pm 1) + b_{\pm}v_x(\pm 1) = 0\}$. Then the standard Legendre–Galerkin approximation to the above equation is, find $u_N \in W_N$ such that

$$\alpha(u_N, v) + (D_x u_N, D_x v) = (f, v) \quad \forall v \in W_N. \tag{4.1}$$

As before, we can find an appropriate basis for W_N by setting

$$\phi_k(x) = L_k(x) + a_k L_{k+1}(x) + b_k L_{k+2}(x),$$

where a_k and b_k are the unique constants such that $\phi_k(x) \in W_N$. Therefore

$$W_N = \text{span}\{\phi_0(x), \phi_1(x), \ldots, \phi_{N-2}(x)\}.$$

Taking $v = \phi_k(x)$ in (4.1) for $k = 0, 1, \ldots, N-2$, we can readily check that the matrix corresponding to the discrete system (4.1) is a symmetric pentadiagonal matrix.

In the special case where $a_\pm = 0$ and $b_\pm = 1$ (i.e., the homogeneous Neumann boundary condition), we have in particular $a_k = 0$ and $b_k = -(k(k+1)/(k+2)(k+3))$. In this particular case the matrix can be decoupled into two tridiagonal submatrices.

Let us mention that we can also construct a special basis for the fourth-order problems with the condition $u|_{\partial\Omega} = \Delta u|_{\partial\Omega} = 0$.

4.2. Nonhomogeneous boundary conditions. We can always modify the right-hand side to take care of the nonhomogeneous boundary conditions. Let us consider for instance the two-dimensional Helmholtz equation (2.1). Nonhomogeneous boundary conditions for fourth-order equations can be treated similarly.

If the solution u of (2.1) is subject to the nonhomogeneous Dirichlet boundary condition

$$u(\pm 1, y) = a_\pm(y), \qquad u(x, \pm 1) = b_\pm(x),$$

we proceed as follows.

Set

$$u^1(x, y) = \frac{b_+(x) - b_-(x)}{2} y + \frac{b_+(x) + b_-(x)}{2}, \qquad \tilde{a}_\pm(y) = a_\pm(y) - u^1(\pm 1, y).$$

By construction, we have $\tilde{a}_\pm(\pm 1) = 0$.

Then set

$$u^2(x, y) = \frac{\tilde{a}_+(y) - \tilde{a}_-(y)}{2} x + \frac{\tilde{a}_+(y) + \tilde{a}_-(y)}{2}, \qquad \tilde{u} = u - u^1 - u^2.$$

One can readily check that

$$\tilde{u}(\pm 1, y) = \tilde{u}(x, \pm 1) = 0.$$

Hence it suffices to solve the following modified Helmholtz equation:

$$\alpha\tilde{u} - \Delta\tilde{u} = f - \alpha(u^1 + u^2) + \Delta(u^1 + u^2) \quad \text{in } \Omega = I \times I, \qquad \tilde{u}|_{\partial\Omega} = 0.$$

We note that the extra term $\Delta(u^1 + u^2)$ on the right-hand side can be obtained in only $O(N)$ operations by using the recursive relation (2.5).

4.3. Nonseparable elliptic equations. For nonseparable equations of the form

$$\begin{aligned} Lu \equiv -\nabla \cdot [a(x, y)\nabla u] + b(x, y)u = f(x, y) \quad \text{in } \Omega = I \times I, \\ u|_{\partial\Omega} = g(x, y), \end{aligned} \tag{4.2}$$

or more generally for problems with variable coefficients, the Legendre–Galerkin methods lead to systems with dense matrices and cannot be efficiently solved by a direct method. However, the fast direct methods for constant coefficient problems developed in the previous sections can be used to efficiently solve equation (4.2) using an iterative procedure proposed in [9]. The heart of this iterative procedure is the utilization of the fast solver for the Helmholtz equation, regardless of the type of discretization employed. It is shown in [9] that the rate of convergence of the iterative procedure is independent of the number of modes used in each direction. Therefore an equation of the form (4.2) can be solved in $O(N^3)$ operations.

5. Numerical results. In this section we give several numerical examples obtained using the algorithms presented in the previous sections.

The pure spectral-Galerkin method is rarely used in practice, since for a general right-hand side function f we are unable to compute *exactly* its representation by Legendre polynomials. Instead, the so-called pseudo-spectral method is used to treat the right-hand side (or the product of two functions), i.e., we replace f by $I_N f$ in the computation, where $I_N f$ is the interpolation of f over the set of Gauss–Lobatto points. This would introduce an extra error term of order $C(\sigma)N^{-\sigma}\|f\|_\sigma$ for $f \in H^\sigma(\Omega)$ (see [8]) in the estimates (2.3) and (3.3). Hence, the spectral accuracy is still retained.

Unfortunately, in the Legendre case, we cannot use FFT to transform from the physical space to the spectral space and vice versa. Instead we shall use direct matrix multiplication, which takes $2dN^{d+1}$ operations for each transformation in a d-dimensional domain. Hence for each particular solution of the Helmholtz equation and the biharmonic equation, $2dN^{d+1}$ additional operations are needed to transform $I_N f$ to the spectral space and to transform the solution u_N to the physical space. However, these additional operations can be eventually reduced, at least for N sufficiently large, to $O(N^d(\log_2 N)^2)$ operations using the fast transform developed by Orszag [17], or to $O(N^d \log_2 N)$ operations using the fast multipole method presented in [4]. Note that this additional work is not needed if one is content with the solution in spectral space with the right-hand side given in spectral space.

We consider first the two examples given in [12].

Example 1. Consider the Poisson equation

$$-\Delta u = 32\sin(4\pi x)\sin(4\pi y) \quad \text{in } \Omega = I \times I, \qquad u|_{\partial\Omega} = 0,$$

with a smooth exact solution $u(x, y) = \sin(4\pi x)\sin(4\pi y)$.

Example 2. Consider the Poisson equation

$$-\Delta u = 1 \quad \text{in } \Omega = I \times I, \qquad u|_{\partial\Omega} = 0,$$

with an exact solution

$$u(x, y) = -\frac{64}{\pi^4} \sum_{\substack{n,m=1 \\ n,m \text{ odd}}}^{\infty} (-1)^{\frac{n+m}{2}} \frac{\cos(\frac{n\pi x}{2})\cos(\frac{m\pi y}{2})}{nm(n^2+m^2)},$$

which has singularities at the four corners.

We shall compare the Legendre–Galerkin method (LGM) with the Chebyshev-tau (CTM) and Chebyshev-collocation (CCM) methods.

Table 1 lists the maximum pointwise error of $u - u_N$ using the LGM, CTM, CCM, and the second- and fourth-order finite difference (FD2 and FD4) methods.

For the first example, all three spectral methods converge exponentially but the LGM is considerably more accurate than the CTM and CCM. For the second example, which has corner singularities, the LGM is as accurate as the CCM and is still clearly more accurate than the CTM. The main source of roundoff errors comes from the matrix decomposition. Since

TABLE 1

Maximum pointwise error of $u - u_N$ for Examples 1 and 2.

Example	N	LGM	CTM	CCM	FD4	FD2
1	16	2.93E-3	3.33E-2	5.25E-3	2.81E-2	2.34E-1
1	32	3.44E-13	4.77E-11	2.12E-12	1.62E-3	5.30E-2
1	64	5.55E-15	8.67E-13	1.55E-13	9.97E-5	1.30E-2
1	128	6.88E-15	2.00E-12	1.56E-13	6.21E-6	3.22E-3
2	16	1.42E-6	3.52E-5	7.47E-7	7.17E-6	9.02E-4
2	32	7.48E-8	2.23E-6	5.51E-8	1.79E-6	2.26E-4

decomposing a symmetric tridiagonal matrix in LGM is much less affected by roundoff errors than decomposing a full matrix in CTM and CCM, the roundoff errors in LGM are much less pronounced than those in CTM and CCM. This is confirmed by the numerical results in Table 1.

Table 2 lists the execution time in seconds on Sun-Sparc2 for the first example using the three spectral methods. The approximate preprocessing times are given in parentheses. The programs were written in Fortran and compiled with the option $-O$. LAPACK routines were used for matrix–matrix multiplications and for solving the eigenvalue problems.

TABLE 2

Execution time and preprocessing time for Example 1.

Example	N	LGM	CTM	CCM
1	32	0.10 (0.03)	0.09 (0.13)	0.05 (0.27)
1	64	0.64 (0.12)	0.44 (0.54)	0.43 (2.44)
1	128	6.96 (1.01)	3.36 (5.20)	4.66 (19.58)

For a particular solution of the Helmholtz equation, LGM is clearly the method of choice since its preprocessing time is negligible. For multiple solutions, the LGM is still competitive with CTM and CCM for N up to the range of 48 to 64.[1] It is not very appropriate to compare the execution time of the collocation method, which provides only the physical representation, with that of the Galerkin and tau methods, which provide both the physical and spectral representations, since derivatives of functions, which are often needed for nonlinear problems, can be efficiently evaluated in spectral representations. Most of the computing time in LGM is used for transforming from physical representation to spectral representation and vice versa. Hence the CPU in LGM can be greatly reduced if one only needs the spectral representation.

Example 3. Consider the two-dimensional biharmonic equation in $\Omega = I \times I$

$$\Delta^2 u = 128\pi^4(\cos(4\pi x)\cos(4\pi y) - \cos(4\pi x)\sin(2\pi y)^2 - \cos(4\pi y)\sin(2\pi x)^2),$$
$$u|_{\partial\Omega} = \frac{\partial u}{\partial \boldsymbol{n}}|_{\partial\Omega} = 0,$$

with a smooth exact solution $u(x, y) = (\sin(2\pi x)\sin(2\pi y))^2$.

The second row of Table 3 lists the maximum pointwise error of $u - u_N$; the third row lists the execution time for Example 3 with approximate preprocessing time in parentheses. The last row lists the condition number of the capacitance matrix P.

[1] This range may be increased for highly parallel computers or when a fast transform method is employed.

TABLE 3
Results for Example 3.

N	16	32	64	128
L^∞ error	1.48E-2	7.45E-12	2.04E-14	2.81E-14
CPU (Pre-P)	0.02 (0.08)	0.12 (0.60)	0.82 (5.15)	8.28 (45.03)
Cond. no. of P	115.524	1608.96	24772.8	393214

It is obvious that the approximate solutions converge exponentially to the exact solution. We note that the condition numbers of P grow like $O(N^4)$. Hence the inversion of P is probably the main source of roundoff errors observed for N large. It is worth noting that the execution time for solving a two-dimensional biharmonic equation is only about 20% more than that for solving a two-dimensional Poisson equation.

The last example we consider is the nonseparable equation (4.2) with

$$a(x, y) = [1 + \tfrac{1}{4}((x+1)^4 + (y+1)^4)]^2, \qquad u(x, y) = \sin(\pi x)\sin(\pi y).$$

For the details of the iterative scheme, we refer to [9]. Note however that we used the shifting factor

$$K \approx \left\| \frac{\Delta\sqrt{a(x, y)}}{\sqrt{a(x, y)}} \right\|_{L^2(\Omega)},$$

which seemed to give a slightly better convergence rate, instead of

$$K \approx \frac{1}{2}\left(\min \frac{\Delta\sqrt{a(x, y)}}{\sqrt{a(x, y)}} + \max \frac{\Delta\sqrt{a(x, y)}}{\sqrt{a(x, y)}} \right),$$

which was suggested in [9]. Note also that the preconditioned conjugate gradient method, as opposed to the simple Richardson iteration proposed in [9], can be used to accelerate the convergence rate. Various iterative methods along with treatments for more general variable coefficient problems will be investigated in a future work.

TABLE 4
Results for the nonseparable equation.

N	16	32	64	128
$\max(u_N^7 - u_N^6)$	2.93E-7	4.87E-7	6.15E-7	6.83E-7
$\max(u_N^7 - u)$	2.47E-8	4.65E-8	6.16E-8	6.98E-8
CPU	0.09	0.62	4.51	45.06

Table 4 summarizes the results after seven Richardson iterations with the initial guess $u_N^0 \equiv 0$. The results clearly demonstrate that the convergence rate of the scheme is independent of N. Note, however, that due to the expensive transformations required by the Galerkin method, a collocation method is probably preferable for this problem, especially when N is large.

6. Concluding remarks. We have presented a systematic way to construct appropriate bases for the Legendre–Galerkin method applied to constant coefficient elliptic problems. We have also developed efficient direct solvers whose complexities are a small multiple of N^{d+1} operations in a d-dimensional domain with $(N-1)^d$ unknowns. Furthermore, the algorithms can be effectively parallelized. Taking into account the fact that the convergence rate of the

Legendre–Galerkin method is exponential, we conclude that our algorithms are very valuable for the specific problems considered in this article. To the best of our knowledge, this algorithm for the fourth-order equations is the first fast direct solver with spectral discretization. Our direct solver for one particular second-order equation is probably the most efficient among the existing spectral methods. Our direct solver for the second-order equations with multiple right-hand sides is also very competitive with the direct solvers by the Chebyshev-tau, Chebyshev-collocation, and spectral element methods (see [12], [8], [18]).

We note that similar techniques can be applied to the Galerkin method using Chebyshev polynomials (cf. [20]) or other Jacobi polynomials.

REFERENCES

[1] C. BERNARDI, G. COPPOLETTA, AND Y. MADAY, *Some spectral approximations of two-dimensional fourth-order problems*, Math. Comp., 59 (1992), pp. 63–76.

[2] C. BERNARDI AND Y. MADAY, *Some spectral approximations of one dimensional fourth-order problems*, in Progress in Approximation Theory, P. Nevai and A. Pinkus, eds., Academic Press, New York, 1991, pp. 43–116.

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

[4] J. P. BOYD, *Multipole expansions and pseudospectral cardinal functions: A new generation of the fast Fourier transform*, J. Comput. Phys., 103 (1992), pp. 184–186.

[5] B. L. BUZBEE AND F. W. DORR, *The direct solution of the biharmonic equation on rectangular regions and the Poisson equation on irregular regions*, SIAM J. Numer. Anal., 11 (1974), pp. 753–763.

[6] B. L. BUZBEE, F. W. DORR, J. A. GEORGE, AND G. H. GOLUB, *The direct solution of the discrete Poisson's equation on irregular domains*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.

[7] B. L. BUZBEE, G. H. GOLUB, AND C. W. NIELSON, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[8] C. CANUTO, M. Y. HUSSAINI, A. QUARTERONI, AND T. A. ZANG, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York–Heidelberg–Berlin, 1987.

[9] P. CONCUS AND G. H. GOLUB, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, SIAM J. Numer. Anal., 10 (1973), pp. 1103–1120.

[10] D. FUNARO AND W. HEINRICHS, *Some results about the pseudospectral approximation of one dimensional fourth-order problems*, Numer. Math., 58 (1990), pp. 399–418.

[11] D. GOTTLIEB AND S. A. ORSZAG, *Numerical Analysis of Spectral Methods: Theory and Applications*, Society for Industrial and Applied Mathematics, Philadelphia, 1977.

[12] D. B. HAIDVOGEL AND T. A. ZANG, *The accurate solution of Poisson equation in Chebyshev polynomials*, J. Comput. Phys., 30 (1979), pp. 167–180.

[13] Y. MADAY AND A. T. PATERA, *Spectral element methods for the incompressible Navier–Stokes equations*, in State-of-the-Art Surveys on Computational Mechanics, A. K. N. and J. T. Oden, eds., 1989, pp. 71–143.

[14] M. MARCUS AND H. MINC, *A Survey of Matrix Theory and Matrix Inequalities*, Prindle, Weber & Schmidt, Boston, 1964.

[15] A. MAYO, *The fast solution of Poisson's and the biharmonic equations on general regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.

[16] A. MAYO AND A. GREENBAUM, *Fast iterative solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 101–118.

[17] S. A. ORSZAG, *Fast eigenfunction transforms*, Science and Computers: Advances in Mathematics Supplementary Studies, 10 (1986), pp. 23–30.

[18] A. T. PATERA, *Fast direct Poisson solvers for high-order finite element discretizations in rectangularly decomposable domains*, J. Comput. Phys., 65 (1986), pp. 474–480.

[19] G. SACCHI-LANDRIANI, *Spectral tau approximation of the two-dimensional Stokes problem*, Numer. Math., 23 (1988), pp. 683–699.

[20] J. SHEN, *Efficient spectral-Galerkin method* II: *Direct solvers for second- and fourth-order equations using Chebyshev polynomials*, SIAM J. Sci. Comput., 16 (1995), to appear.

[21] ———, *A spectral-tau approximation for the Stokes and Navier–Stokes equations*, Math. Model. Numer. Anal., 22 (1988), pp. 677–693.

[22] J. SHEN AND R. TEMAM, *Nonlinear Galerkin methods using Chebyshev or Legendre polynomials* I: *One dimensional case*, SIAM J. Numer. Anal., 32 (1995), to appear.

[23] G. SIVASHINSKY, *Nonlinear analysis of hydrodynamic instability in laminar flames, Part* I: *Derivation of basic equations*, Acta Astronaut., 4 (1977), pp. 1177–1206.

[24] R. W. de VRIES AND P. J. ZANDBERGEN, *The numerical solution of the biharmonic equation using a spectral multigrid method*, in Advances in Fluid Dynamics, W. F. Ballhaus and M. Y. Hussaini, eds., 1989, pp. 25–35.