SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 1, March 1982

CONTROL OF ERRORS IN THE COMPUTATION OF MOMENTS OF RANKED DISCRETE VARIABLES*

EDWARD L. MELNICK†

Abstract. Most discrete probability functions do not admit closed forms for the moments of their ranked variates. A computational method is proposed that allows error bounds to be set that are a function of both truncation error and roundoff error.

Key words. moments of ranked discrete variates, controlling computational errors, roundoff errors

1. Introduction. Nonparametric statistics deals almost exclusively with order statistics. Most testing procedures are based upon the moments of these statistics which, in general, cannot be expressed in closed form. They require numerical approximations. Such solutions generally contain errors in two areas: (1) errors resulting from the mathematical formulation of the problem and (2) errors incurred in finding the numerical solution (Ralston and Rabinowitz [5]). The first error type occurs when the mathematical statement is an approximation to the actual problem, for example, the replacement of an infinite process by a finite approximation. Roundoff error represents the second type of error.

David [3, Chapt. 4] provides an excellent discussion of a number of computational methods for computing moments when the underlying distribution is continuous. One method assumes a strictly increasing cumulative distribution function and uses Schwarz's inequality to bound the moments of order statistics. Although this method can also be applied to underlying discrete parents by approximating it with arbitrary precision using strictly increasing continuous functions, roundoff error is introduced with the calculation of the continuous functions. Related to this approach is a method proposed by Sugiura [6] that provides bounds and approximations for means, variances and covariances of order statistics based on orthogonal inverse expansions. This method is not recommended for computing higher moments because of the resulting formidable formulation of the truncation error bound and the extremely difficult problem of constructing reasonable roundoff error bounds of the sequence of calculated polynomials comprising the orthonormal system of equations. Another method, first proposed by Pearson [4] and David and Johnson [2], is based on the Taylor expansion of the inverse cumulative distribution function. This approach does not appear to carry over to the area of discrete distributions since the differential calculus has many conveniences that the calculus of differences lacks; for example, the inverse Taylor series approach has no simple counterpart in the calculus of differences.

This paper presents a computational procedure for computing moments of ordered variates from an underlying discrete distribution function by setting the truncation error and a uniform roundoff error bound. Let $X_{1:n} \leqq \cdots \leqq X_{r:n} \leqq \cdots \leqq X_{n:n}$ be an ordered sample of observations from a discrete probability mass function $f(x)$ and corresponding cumulative distribution function

$$F(k) = \sum_{x=0}^{k} f(x). \tag{1.1}$$

* Received by the editors April 18, 1980, and in revised form June 17, 1981.

† New York University, Graduate School of Business Administration, 100 Trinity Place, New York, New York 10006.

The probability that the rth smallest of n ranked variates has the value k is

$$P(X_{r:n}=k)=\int_{F(k-1)}^{F(k)}\frac{(1-v)^{n-r}v^{r-1}}{\beta(r,n-r+1)}\,dv, \tag{1.2}$$

where $\beta(p,q)$ is the beta function with parameters p and q. By definition,

$$E(X^t_{r:n})=\sum_{k=0}^{\infty}k^t\int_{F(k-1)}^{F(k)}\frac{v^{r-1}(1-v)^{n-r}}{\beta(r,n-r+1)}\,dv. \tag{1.3}$$

Since k takes integer values, it is easily shown that ([3, eq. (3.3.5)])

$$E(X_{r:n})=\sum_{k=0}^{\infty}\int_{F(k)}^{1}\frac{v^{r-1}(1-v)^{n-r}}{\beta(r,n-r+1)}\,dv, \tag{1.4}$$

so that the first moment is a sum of incomplete beta-integrals. If the range of k is finite, (1.4) is probably the best form for the computation of $E(X_{r:n})$. For an infinite range of variation, the series must be truncated at some point.

As in the continuous case, we have a simple relationship of the form

$$(n-r)E(X^t_{r:N})=nE(X^t_{r:n-1})-rE(X^t_{r+1:n}),\qquad 0\leqq r<n \tag{1.5}$$

(David [3]). Since all moments may be expressed as a linear combination of $E(X^t_{1:m})$ and $E(X^t_{m:m})$, where $m\leqq n$, we concentrate our attention upon the moments of the extreme order statistics $X_{1:n}$ and $X_{n:n}$, where

$$E(X^t_{1:n})=\sum_{k=0}^{\infty}k^t[(1-F(k-1))^n-(1-F(k))^n] \tag{1.6}$$

and

$$E(X^t_{n:n})=\sum_{k=0}^{\infty}k^t[F(k)^n-(F(k-1))^n]. \tag{1.7}$$

Despite their apparent simplicity, (1.6) and (1.7) are very difficult to use for general n and t.

2. Control of errors in the computation of $E(X^t_{1:n})$ and $E(X^t_{n:n})$.

The first theorem presents a bound on the truncation error incurred in the representation of the tth moment of the largest order statistics by a finite approximation.

THEOREM 1.

$$\left|E(X^t_{n:n})-\sum_{k=1}^{m}k^t\Pr(X_{n:n}=k)\right|<n\sum_{n=m+1}^{\infty}k^tf(k). \tag{2.1}$$

Proof. Write the tth moment of the largest order statistic by truncating the series after the mth term as

$$E(X^t_{n:n})=\sum_{k=1}^{m}k^t\Pr(X_{n:n}=k)+R_m(t), \tag{2.2}$$

where $R_m(t)$ (the remainder after the mth term) is given by

$$R_m(t)=\sum_{k=m+1}^{\infty}k^t\Pr(X_{n:n}=k). \tag{2.3}$$

Using the formula for the difference between the nth power of two quantities gives

$$R_m(t)=\sum_{k=m+1}^{\infty}k^tf(k)\left[\sum_{i=0}^{n-1}F^i(k)F^{n-1-i}(k-1)\right]. \tag{2.4}$$

Since $F(k) \leqq 1$, it follows that

$$R_m(t) < n \sum_{k=m+1}^{\infty} k^t f(k) = n\left(\sum_1^{\infty} k^t f(k) - \sum_1^{m} k^t f(k)\right). \tag{2.5}$$

The theorem is proved.

We now look at the control of roundoff error. An error ε in $f(k)$ will produce an error of modulus not greater than $(k+1)|\varepsilon|$ in $F(k)$. Let

$$E_k = (k+1)|\varepsilon|. \tag{2.6}$$

If $\tilde{F}(k)$ is the computed value of $F(k)$, then at worst

$$\tilde{F}(k) = F(k) \pm E_k, \tag{2.7}$$

so that from (1.7) the computed value

$$E_c(X^t_{n:n}) = \sum_{k=1}^{m} k^t[\tilde{F}(k)^n - (\tilde{F}(k-1))^n] \tag{2.8}$$

is subjected to roundoff error. Theorem 2 provides a bound for this error.

THEOREM 2.[1]

$$\left|\sum_{k=1}^{m} k^t[\tilde{F}^n(k) - \tilde{F}^n(k-1)] - \sum_{k=1}^{m} k^t[F^n(k) - F^n(k-1)]\right| \leqq 2m^t(m+1)n|\varepsilon|. \tag{2.9}$$

Proof. Use the formula for the difference between the nth power of two quantities to show

$$|\tilde{F}^n(k) - F^n(k)| = |\tilde{F}(k) - F(k)| \cdot \left|\sum_{j=0}^{n-1} \tilde{F}^j(k) F^{n-1-j}(k)\right|. \tag{2.10}$$

Since $F(k) \leqq 1$, it follows that

$$|\tilde{F}^n(k) - F^n(k)| \leqq n|\tilde{F}(k) - F(k)| \leqq nE_k. \tag{2.11}$$

Let

$$D_n(k) = \tilde{F}^n(k) - F^n(k),$$

and rewrite the roundoff error

$$\begin{aligned}
&\left|\sum_{k=1}^{m} k^t[\tilde{F}^n(k) - \tilde{F}^n(k-1)] - \sum_{k=1}^{m} k^t[F^n(k) - F^n(k-1)]\right| \\
&\qquad = \left|\sum_{k=1}^{m} k^t[D_n(k) - D_n(k-1)]\right| \\
&\qquad = \left|m^t D_n(m) + \sum_{k=0}^{m-1} [k^t - (k+1)^t] D_n(k)\right| \qquad (2.12)\\
&\qquad \leqq m^t|D_n(m)| + \sum_{k=0}^{m-1} |D_n(k)|[(k+1)^t - k^t] \\
&\qquad \leqq 2m^t(m+1)n|\varepsilon|.
\end{aligned}$$

The last line follows from (2.11). The theorem is proved.

[1] The author is indebted to an anonymous referee for tightening the bound in this theorem.

A similar result holds for the tth moment of the smallest order statistic. If $E_c(X^t_{p:n})$ is the computed value of $E(X^t_{p:n})$ $p=1$ or n, then from Theorems 1 and 2

$$|E(X^t_{p:n})-E_c(X^t_{p:n})|\leqq R_m(t)+2m^t(m+1)n|\varepsilon|. \tag{2.13}$$

Under the assumption that $\sum_{k=1}^{\infty} k^t f(k)$ can be calculated without roundoff, a bound of the roundoff error in the calculation of $R_m(t)$ can be obtained by noting that the error in $n\sum_{k=1}^{m} k^t f(k)$ is bounded by $n|\varepsilon|\sum_{k=1}^{m} k^t$. The resulting roundoff error of $R_m(t)$ is of the same order of magnitude as found in Theorem 2.

By repeatedly applying (1.5) to the right-hand side of itself, recursive relations [3, § 3.4] are derived as

$$E(X^t_{r:n})=(-1)^r\sum_{i=r}^{n}\binom{i-1}{r-1}\binom{n}{i}(-1)^i E(X^t_{i:i}) \tag{2.14}$$

and

$$E(X^t_{r:n})=(-1)^{n-r+1}\sum_{i=n-r+1}^{n}(-1)^i\binom{n}{i}\binom{i-1}{n-r}E(X^t_{1:i}). \tag{2.15}$$

From these relationships and the moments of the extreme order statistics, moments of the rth smallest of n ranked variates can be obtained from (2.14), if $r>n-r+1$ (or from (2.15), if $r\leqq n-r+1$). Thus, if the largest error in $E(X^t_{i:i})$, where $r\leqq i\leqq n$ is $|\delta|$, the error in $E(X^t_{r:n})$ could be as large as $2^{n-r}\binom{n}{r}|\delta|$.

3. Example. Let $X_{1:n}\leqq X_{2:n}\leqq\cdots\leqq X_{n:n}$ be a ranked sample of n independent observations from the Poisson probability mass function

$$f(k)=\begin{cases}\dfrac{e^{-\lambda}\lambda^k}{k!}, & k=0,1,\cdots,\\ 0 & \text{elsewhere.}\end{cases} \tag{3.1}$$

In the range $n=1(1)10$; $\lambda=1(1)10$ and $t=1,2$, we have, for $p=1$ and n,

$$|E(X^t_{p:n})-E_c(X^t_{p:n})|\leqq 110n\sum_{m-1}^{\infty} e^{-10}\frac{10^k}{k!}+2m^t(m+1)n|\varepsilon| \tag{3.2}$$

(the worst cases occurring when $\lambda=10$, $n=20$, and $t=2$). The remainder, Rm (2), for the Poisson distribution is

$$\begin{aligned}\mathrm{Rm}(2)\leqq n\sum_{m+1}^{\infty}k^2f(k)=n\sum_{m+1}^{\infty}k^2e^{-\lambda}\frac{\lambda^k}{k!}&=n\sum_{m+1}^{\infty}\left(e^{-\lambda}\frac{\lambda^k}{(k-2)!}+e^{-\lambda}\frac{\lambda^k}{(k-1)!}\right)\\ &\leqq n(\lambda^2+\lambda)\,\mathrm{Pr}\,(k\geqq m-1).\end{aligned} \tag{3.3}$$

Thus, by setting $|\varepsilon|=10^{-13}$ and $m=35$, the resulting error is everywhere less than 2×10^{-7}.

4. Concluding remarks. Many computational algorithms produce numerical solutions by placing bounds on errors resulting from truncating series expansions. Rarely is the accumulation of roundoff errors considered in the evaluation of the output. One noticeable exception is the interesting paper by Aggarwal and Burgmeier [1] in which an error model is developed that produces error bounds for evaluating computational roundoff errors. In our paper, a method for computing moments of ranked discrete variates is presented. The method is based on a recursive relationship between the moments of the rth order statistic and the moments of the extreme order statistics. Bounds on the estimated moments are constructed as a function of both truncation error and computational roundoff errors. As m increases, the truncation error bound decreases while the roundoff error bound increases. When a larger value of m is needed to obtain a negligible truncation error, high precision in the computations (i.e., setting of $|\varepsilon|$) is required since the roundoff error is a function of m^{t+1}. Although it may be possible to construct a set of tighter error bounds for different values of m, such a procedure has not yet been developed. The proposed algorithm is a straightforward method that generates moments with any desired level of accuracy.

REFERENCES

[1] V. B. AGGARWAL AND J. W. BURGMEIER, *A roundoff error model with applications to arithmetic expressions*, this Journal, 8 (1979), pp. 60–71.

[2] F. N. DAVID AND N. L. JOHNSON, *Statistical treatment of censored data*, Biometrika, 41 (1954), pp. 228–240.

[3] H. A. DAVID, *Order Statistics*, John Wiley, New York, 1970.

[4] K. PEARSON, *On the mean character and variance of a ranked individual and on the mean and variance of the intervals between ranked individuals*, Biometrika, 23 (1931), pp. 364–376.

[5] A. RALSTON AND P. RABINOWITZ, *A First Course in Numerical Analysis*, McGraw-Hill, New York, 1978.

[6] N. SUGIURA, *On the orthogonal inverse expansion with an application to the moments of order statistics*, Osaka Math. J., 14 (1962), pp. 253–263.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 1, March 1982

AN ADAPTIVE FINITE ELEMENT METHOD FOR INITIAL-BOUNDARY VALUE PROBLEMS FOR PARTIAL DIFFERENTIAL EQUATIONS*

STEPHEN F. DAVIS† AND JOSEPH E. FLAHERTY‡

Abstract. A finite element method is developed to solve initial-boundary value problems for vector systems of partial differential equations in one space dimension and time. The method automatically adapts the computational mesh as the solution progresses in time and is, thus, able to follow and resolve relatively sharp transitions such as mild boundary layers, shock layers or wave fronts. This permits an accurate solution to be calculated with fewer mesh points than would be necessary with a uniform mesh.

The overall method contains two parts, a solution algorithm and a mesh selection algorithm. The solution algorithm is a finite element-Galerkin method on trapezoidal space-time elements, using either piecewise linear or cubic polynomial approximations, and the mesh selection algorithm builds upon similar work for variable knot spline interpolation.

A computer code implementing these algorithms has been written and applied to a number of problems. These computations confirm that the theoretical error estimates are attained and demonstrate the utility of variable mesh methods for partial differential equations.

Key words. finite element methods, initial-boundary value problems, adaptive grids, moving elements, partial differential equations

1. Introduction.

In this paper we construct an adaptive grid finite element procedure to find numerical solutions of M-dimensional vector systems of partial differential equations having the form

$$\mathbf{L}\mathbf{u} := \mathbf{u}_t + \mathbf{f}(x, t, \mathbf{u}, \mathbf{u}_x) - [\mathbf{D}(x, t, \mathbf{u})\mathbf{u}_x]_x = 0, \qquad 0 < x < a, \quad t > 0, \tag{1.1}$$

subject to the initial and linear separated boundary conditions

$$\mathbf{u}(x, 0) = \mathbf{u}^0(x), \qquad 0 \leqq x \leqq a, \tag{1.2}$$

$$\begin{aligned} \mathbf{B}_1\mathbf{u}(0, t) &:= \mathbf{A}_{11}(t)\mathbf{u}(0, t) + \mathbf{A}_{12}(t)\mathbf{u}_x(0, t) = \mathbf{b}_1(t), \\ \mathbf{B}_2\mathbf{u}(a, t) &:= \mathbf{A}_{21}(t)\mathbf{u}(a, t) + \mathbf{A}_{22}(t)\mathbf{u}_x(a, t) = \mathbf{b}_2(t), \qquad t > 0. \end{aligned} \tag{1.3}$$

There are k_1 initial boundary conditions at $x = 0$ and k_2 terminal boundary conditions at $x = a$. We are primarily concerned with solving diffusion problems where $\mathbf{D}$ is positive definite and $k_1 + k_2 = 2M$; however, we will not restrict ourselves to this case, but instead, we assume that conditions are specified so that (1.1)–(1.3) have an isolated solution.

Problems of the above form arise in many applications which model problems as diverse as heat conduction (cf. Friedman [18]), determining bacterial motion (cf. Keller and Odell [28], [33]), combustion (cf. Kapila [27]), chemical reactions (cf. Fife [16]), population dynamics (cf. Hoppensteadt [23]) and convecting flows (cf. Batchelor

* Received by the editors March 26, 1981, and in final form July 27, 1981. This research was sponsored by the U.S. Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant AFOSR 80-0192. Partial support for work performed under NASA contract NAS1-15810 was also provided while the second author was in residence at ICASE, NASA Langley Research Center.

† Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York 12181. Present address: Ballistic Research Laboratory, U.S. Army Armament Research and Development Command, Aberdeen Proving Ground, Maryland 21005. This work was submitted in partial fulfillment of his Ph.D. requirements at the Rensselaer Polytechnic Institute.

‡ Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York 12181, and Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665.

[4]). Therefore, a general purpose code to solve (1.1)–(1.3) numerically would be extremely useful.

Many of the problems mentioned above have solutions which contain sharp transitions such as boundary layers, shock layers or wave fronts. In order to resolve such nonuniformities using a minimum number of mesh points, it is desirable to concentrate the mesh within the transition layers. Since these transition layers can move, it is all the more desirable for the mesh to adapt itself with the evolving solution. To do this we develop methods that (i) discretize (1.1)–(1.3) on a nonuniform mesh and (ii) determine the proper mesh point locations.

We discretize (1.1)–(1.3) using a finite element Galerkin method on trapezoidal space-time elements. This approach is similar to that of Jamet and Bonnerot [6], [25], and it was chosen because it is generally easier to generate high order approximations to partial differential equations on a nonuniform mesh with finite element methods than with finite difference methods. The accuracy and order of convergence of our methods are analyzed in Davis [13] and are demonstrated experimentally in § 4 of this paper.

Adaptive mesh selection strategies typically involve some recomputation of the solution. That is, an initial solution is computed on a coarse mesh, and this is used to determine whether to add mesh points to some portion of the domain and redo the calculation, redo the calculation using a more accurate method, redo the computation using some combination of these methods or accept the present computation. Algorithms of this general type have been developed and successfully applied to adaptive quadrature (cf., e.g., Rice [36] and Lyness and Kaganove [31]), two-point boundary value problems (cf., e.g., Childs et al. [9]), elliptic boundary value problems (cf., e.g., Carey [8] and Brandt [7]) and parabolic and hyperbolic problems (cf., e.g., Gropp [20], Berger et al. [5] and White [40]).

Primarily because of the expense involved in recomputing the solution of the partial differential equations at possibly every time step, we have developed an algorithm which initially places a fixed number of mesh points in optimal locations and then attempts to move them so that their locations remain optimal. Algorithms of this type have been used by Lawson [29], de Boor [14], [15] and Jupp [26] for variable knot spline interpolation, and it is their work that motivated our mesh selection algorithms.

A different approach to this problem was proposed by Miller and Miller [32] and later extended by Galinas et al. [19]. They approximated the solution of parabolic partial differential equations by piecewise linear polynomials where both the polynomial coefficients and the mesh on which they were defined were unknown functions of time. These functions were determined by minimizing the least squares residuals. They found that the mesh points could coalesce in certain situations, and they avoided this by adding a number of spring and damping terms as constraints to the equations.

One advantage of the above approach is that it readily extends to higher dimensional problems. However, we are not convinced that it is necessary to couple the solution and mesh selection methods. This can dramatically increase the size of the discrete system without offering any corresponding increase in order of accuracy. Furthermore, the entire solution procedure must halt if an acceptable mesh cannot be calculated. Under the same circumstances, our methods can continue to compute a solution on a suboptimal mesh. Since both methods are under development we have not attempted any detailed comparisons.

In § 2 of this paper we develop a finite element Galerkin approximation to (1.1)–(1.3) using trapezoidal space-time elements. In § 3 we describe a practical and efficient mesh selection procedure that approximately minimizes the L_2-error of the computed solution. In § 4 we apply the method to a number of problems and discuss the computed results. Finally, in § 5 we present an overall discussion of this effort and some suggestions for future work.

2. Finite element formulation. We discretize (1.1)–(1.3) using a finite element-Galerkin procedure. To this end, let S_n be the strip

$$S_n := \{(x,t) | 0 \leqq x \leqq a,\ t_n \leqq t \leqq t_{n+1}\}, \tag{2.1}$$

choose "test" or "weight" functions $v(x,t) \in C^0(S_n)$, multiply (1.1) by v, integrate over S_n and integrate the time derivative and diffusive terms by parts to get

$$\begin{aligned} \mathbf{F}(\mathbf{u}, v) := \int_{t_n}^{t_{n+1}} \int_0^a \{-\mathbf{u}v_t + \mathbf{f}(x,t,\mathbf{u},\mathbf{u}_x)v + \mathbf{D}(x,t,\mathbf{u})\mathbf{u}_x v_x\}\,dx\,dt \\ + \int_0^a \mathbf{u}v\,dx \bigg|_{t=t_n}^{t_{n+1}} - \int_{t_n}^{t_{n+1}} \mathbf{D}\mathbf{u}_x v\,dt \bigg|_{x=0}^{a} = 0. \end{aligned} \tag{2.2}$$

Equation (2.2) is called the Galerkin form of the problem, and any function $\mathbf{u}$ that satisfies (2.1) and the initial and boundary conditions (1.2), (1.3) is called a "weak solution."

We introduce a mesh $\{0 = x_1^n < x_2^n < \cdots < x_N^n = a\}$ at $t = t_n$ and a different mesh $\{0 = x_1^{n+1} < x_2^{n+1} < \cdots < x_N^{n+1} = a\}$ at $t = t_{n+1}$. We connect the corresponding points x_i^n and x_i^{n+1} by straight lines, and consequently divide the strip S_n into a set of $N-1$ trapezoids. We let $x_i(t)$ denote the straight line connecting x_i^n and x_i^{n+1} and let T_i^n denote the trapezoid with vertices (x_i^n, t_n), (x_{i+1}^n, t_n), (x_{i+1}^{n+1}, t_{n+1}), (x_i^{n+1}, t_{n+1}) (cf. Fig. 1).

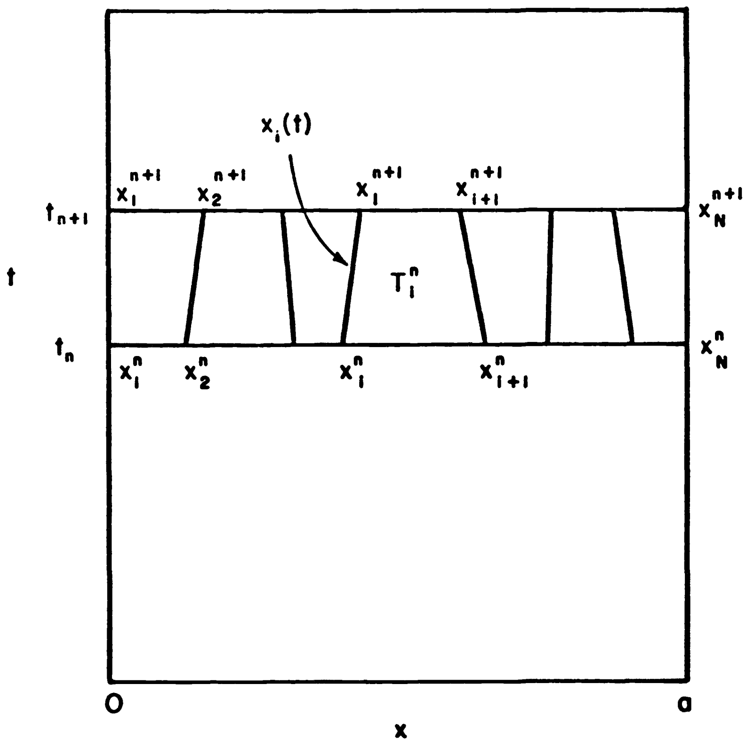


FIG. 1. *Space-time discretization for the time step $t_n \leqq t \leqq t_{n+1}$.*

We approximate $\mathbf{u}(x, t)$ on S_n by $\mathbf{U}(x, t) \in \mathcal{U}_K(S_n)$, which has the form

$$\mathbf{U}(x, t) = \sum_{i=1}^{K} \mathbf{c}_i(t)\phi_i(x, t). \tag{2.3}$$

The "trial" functions $\phi_i(x, t)$, $i = 1, 2, \cdots, K$, can be used to construct a basis for $\mathcal{U}_K(S_n)$. They are selected to be of class $C^0(S_n)$ and, in finite element methods, to have small support. Particular choices of ϕ_i are given in § 2.1; herein, it suffices to note that ϕ_i is nonzero only on $T_{i-1}^n \cup T_i^n$ and that K must be at least N.

We determine $\mathbf{U}$ on S_n by solving a discrete problem of the form

$$\mathbf{PU}(x, 0) = \mathbf{Pu}^0(x), \qquad n = 0, \tag{2.4}$$

$$\hat{\mathbf{F}}(\mathbf{U}, v) = 0 \quad \forall v \in \mathcal{V}_K, \tag{2.5}$$

$$\hat{\mathbf{B}}_1\mathbf{U}(0, t_{n+1}) = \hat{\mathbf{b}}_1(t_{n+1}), \qquad \hat{\mathbf{B}}_2\mathbf{U}(a, t_{n+1}) = \hat{\mathbf{b}}_2(t_{n+1}). \tag{2.6}$$

Here $\mathcal{V}_K$ is a finite dimensional space of $C^0(S_n)$ functions that depends on the boundary conditions (cf. § 2.3); $\mathbf{P}$ is an interpolation operator (cf. § 2.3); $\hat{\mathbf{B}}_1$, $\hat{\mathbf{b}}_1$, $\hat{\mathbf{B}}_2$, $\hat{\mathbf{b}}_2$ are approximations of $\mathbf{B}_1$, $\mathbf{b}_1$, $\mathbf{B}_2$, $\mathbf{b}_2$ obtained by numerical integration (cf. § 2.3); and $\hat{\mathbf{F}}(\mathbf{U}, v)$ is an approximation of $\mathbf{F}(\mathbf{U}, v)$ obtained by evaluating the integrals in (2.2) numerically (cf. § 2.2). Equations (2.5), (2.6) result in an MK-dimensional nonlinear algebraic system for determining the Galerkin coordinates $\mathbf{c}_i(t_{n+1})$, $i = 1, 2, \cdots K$, in terms of $\mathbf{c}_i(t_n)$, $i = 1, 2, \cdots, K$. Since $\mathbf{c}_i(0)$, $i = 1, 2, \cdots, K$, are determined from the initial conditions (2.4), (2.4–6) define a marching algorithm for determining $\mathbf{U}(x, t)$ in successive strips S_n, $n = 0, 1, \cdots$.

If there were no boundary conditions, we would select $\phi_i(x, t)$, $i = 1, 2, \cdots, K$, as a basis for $\mathcal{V}_K$. This prescription has to be modified slightly for $i = 1$ and/or $i = K$ (cf. § 2.3) since boundary conditions are generally imposed; however, it is still appropriate to write $\mathbf{F}(\mathbf{U}, v)$ as a sum of contributions from each trapezoid. Thus,

$$\begin{aligned}\mathbf{F}(\mathbf{U}, v) = &\sum_{i=1}^{N-1} \iint_{T_i^n} \{-\mathbf{U}v_t + \mathbf{f}(x, t, \mathbf{U}, \mathbf{U}_x)v + \mathbf{D}(x, t, \mathbf{U})\mathbf{U}_x v_x\}\, dx\, dt \\ &+ \sum_{i=1}^{N-1} \left[\left[\int_{x_i(t)}^{x_{i+1}(t)} \mathbf{U}v\, dt\right]_{t=t_n}^{t_{n+1}} - \left[\int_{t_n}^{t_{n+1}} \mathbf{D}\mathbf{U}_x v\, dt\right]_{x=0}^{a}\right] = 0 \quad \forall v \in \mathcal{V}_K.\end{aligned} \tag{2.7}$$

Since the bases for both the trial and test spaces have small support, most of the integrals in (2.7) will be zero. The algebraic system (2.5), (2.6) will be sparse, and hence, it may be solved efficiently.

2.1. Selection of a basis. A simple way to construct a basis on trapezoidal elements that satisfies the necessary continuity requirements and has small support is to apply a local transformation that maps each trapezoid onto a rectangle. The inverse of this transformation on T_i^n is

$$\begin{aligned} x &= x_i^n + (x_{i+1}^n - x_i^n)\left(\frac{\xi+1}{2}\right) + (x_i^{n+1} - x_i^n)\tau + (x_{i+1}^{n+1} - x_i^{n+1} - x_{i+1}^n + x_i^n)\left(\frac{\xi+1}{2}\right)\tau, \\ t &= t_n + (t_{n+1} - t_n)\tau. \end{aligned} \tag{2.8}$$

It maps the rectangle

$$R = \{(\xi, \tau) | -1 \leqq \xi \leqq 1, 0 \leqq \tau \leqq 1\} \tag{2.9}$$

in the (ξ, τ)-plane onto T_i^n in the (x, y)-plane.

We choose this basis so that $\phi_i(x, t)$ is a function of ξ only on T_j^n. To be specific, we currently allow $\phi_i(x, t)$ to be either a piecewise linear or a piecewise Hermite cubic polynomial in ξ on T_j^n.

For piecewise linear approximations, we construct a basis in terms of the canonical basis function

(2.10) $$\hat{\phi}(\xi) = \frac{1-\xi}{2}, \qquad -1 \leqq \xi \leqq 1$$

by defining

(2.11) $$\phi_i(x, t) = \begin{cases} \hat{\phi}(\xi), & (x, t) \in T_i^n, \\ \hat{\phi}(-\xi), & (x, t) \in T_{i-1}^n, \quad i = 1, 2, \cdots, K = N, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the dimension of the trial space $\mathcal{U}_K$ is $K = N$. Along the line $x_j(\tau)$ joining x_j^n and x_j^{n+1}, we have

(2.12) $$\phi_i(x_j(\tau), t(\tau)) = \delta_{ij}, \qquad 0 \leqq \tau \leqq 1,$$

where δ_{ij} is the Kronecker delta. Using (2.3), this implies that

(2.13) $$\mathbf{c}_i(t) = \mathbf{U}_i(\tau) := \mathbf{U}(x_i(\tau), t(\tau)).$$

Thus, since only ϕ_i and ϕ_{i+1} are nonzero on T_i^n, we have

(2.14) $$\mathbf{U}(x, t) = \mathbf{U}_i(\tau)\hat{\phi}(\xi) + \mathbf{U}_{i+1}(\tau)\hat{\phi}(-\xi), \qquad (x, t) \in T_i^n.$$

For piecewise cubic Hermite approximations, we construct a basis in terms of the two canonical basis functions

(2.15) $$\hat{\psi}(\xi) = \tfrac{1}{4}(1-\xi)^2(2+\xi), \qquad \hat{\chi}(\xi) = \tfrac{1}{4}(1+\xi)(1-\xi)^2, \qquad 1 \leqq \xi \leqq 1$$

by defining

(2.16a) $$\phi_{2i-1}(x, t) = \begin{cases} \hat{\psi}(\xi), & (x, t) \in T_i^n \\ \hat{\psi}(-\xi), & (x, t) \in T_{i-1}^n, \quad i = 1, 2, \cdots, N, \\ 0, & \text{otherwise.} \end{cases}$$

(2.16b) $$\phi_{2i}(x, t) = \begin{cases} \hat{\chi}(\xi), & (x, t) \in T_i^n, \\ -\hat{\chi}(-\xi), & (x, t) \in T_{i-1}^n, \quad i = 1, 2, \cdots, N, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the dimension K of the trial space is $2N$.

We note that $\phi_{2i-1}(x, t) \in C^1(S_n)$ with

(2.17a, b) $$\phi_{2i-1}(x_j(\tau), t(\tau)) = \delta_{ij}, \qquad \phi_{2i-1_x}(x_j(\tau), t(\tau)) = 0, \qquad 0 \leqq \tau \leqq 1,$$

but $\phi_{2i}(x, t) \in C^0(S_n)$ with

(2.17c, d) $$\phi_{2i}(x_j(\tau), t(\tau)) = 0, \qquad \phi_{2i_x}(x_j(\tau), t(\tau)) = \frac{\delta_{ij}}{x_\xi(\tau)}, \qquad 0 \leqq \tau \leqq 1.$$

The function $x_\xi(\tau)$ is easily computed from (2.8) as

(2.18) $$x_\xi(\tau) = \tfrac{1}{2}(x_{j+1}^n - x_j^n) + \tfrac{1}{2}(x_{j+1}^{n+1} - x_j^{n+1} - x_{j+1}^n + x_j^n)\tau \quad \text{if } (x, t) \in T_j^n.$$

Thus, $x_\xi(\tau)$ is different on each trapezoid (unless the mesh is uniform and rectangular) and $\phi_{2i_x}(x, t)$ jumps as x crosses $x_i(\tau)$. However, using (2.3) and (2.17), we can make $\mathbf{U}(x, t)$ of class $C^1(S_n)$ by selecting

$$\mathbf{c}_{2i-1}(t) = \mathbf{U}_i(\tau) := \mathbf{U}(x_i(\tau), t(\tau)), \tag{2.19a}$$

$$\mathbf{c}_{2i}(t) = x_\xi(\tau)\mathbf{U}_{x_i}(\tau) := x_\xi(\tau)\mathbf{U}_x(x_i(\tau), t(\tau)). \tag{2.19b}$$

Thus, on T_i^n we have

$$\mathbf{U}(x, t) = \mathbf{U}_i(\tau)\hat{\psi}(\xi) + \mathbf{U}_{i+1}(\tau)\hat{\psi}(-\xi) + \mathbf{U}_{x_i}(\tau)x_\xi(\tau)\hat{\chi}(\xi) - \mathbf{U}_{x_{i+1}}(\tau)x_\xi(\tau)\hat{\chi}(-\xi). \tag{2.20}$$

2.2. Numerical integration. Ignoring the boundary conditions for the moment, we choose $v = \phi_j(x, t)$ according to either (2.11) or (2.16) and use (2.8) to transform (2.7) to

$$\mathbf{F}(\mathbf{U}, \phi_j) = \sum_{i=1}^{N-1} \mathbf{I}_i(\mathbf{U}, \phi_j) - \mathbf{I}_B(\mathbf{U}, \phi_j) = 0, \qquad j = 1, 2, \cdots, K, \tag{2.21}$$

where

$$\begin{aligned}\mathbf{I}_i(\mathbf{U}, v) = \int_0^1 \int_{-1}^1 \{&-\mathbf{U}v_\xi\xi_t + \mathbf{f}(x, t, \mathbf{U}, \mathbf{U}_\xi\xi_x)v \\ &+ \mathbf{D}(x, t, \mathbf{U})\mathbf{U}_\xi v_\xi \xi_x^2\}|J|\, d\xi\, d\tau + \int_{-1}^1 \mathbf{U}vx_\xi\, d\xi\Big|_{\tau=0}^1,\end{aligned} \tag{2.22a}$$

$$\mathbf{I}_B(\mathbf{U}, v) = \int_0^1 \mathbf{D}(x, t, \mathbf{U})\mathbf{U}_\xi v\xi_x t_\tau\, d\tau\Big|_{x=0,\xi=-1}^{x=a,\xi=1}. \tag{2.22b}$$

The functions ξ_t, ξ_x, x_ξ, t_τ and $|J|$, the Jacobian of the transformation, can be computed from (2.8).

In order to complete the specification of our numerical method, we need to select quadrature rules for evaluating the integrals in (2.22). We use the trapezoidal rule to evaluate the τ integrals and a three point Gauss–Legendre rule (cf. Abramowitz and Stegun [1, Chap. 25]) to evaluate the ξ integrals. The latter was chosen because it is known (cf. Strang and Fix [38]) to have the same order discretization error as our finite element method, with cubic approximations and the exact integration of (2.22). At present, we also use the three point Gauss–Legendre rule for linear approximations although it is more accurate than necessary in this case and, therefore, somewhat inefficient.

Upon use of the above mentioned quadrature rules, the equations (2.21) become

$$\hat{\mathbf{F}}(\mathbf{U}, \phi_j) = \sum_{i=1}^{N-1} \hat{\mathbf{I}}_i(\mathbf{U}, \phi_j) - \hat{\mathbf{I}}_B(\mathbf{U}, \phi_j) = 0, \qquad j = 1, 2, \cdots, K, \tag{2.23}$$

where $\hat{\mathbf{I}}_i$ and $\hat{\mathbf{I}}_B$ denote the approximations of (2.22) that are obtained by numerical integration.

2.3. Initial and boundary conditions. Solution technique. The solution $\mathbf{U}$ is determined on S_n by solving (2.23) together with the initial and boundary conditions (2.4) and (2.6), respectively. We satisfy the initial conditions (and implicitly define the interpolation operator $\mathbf{P}$ of (2.4)) by requiring

$$\mathbf{U}_i^0 = \mathbf{u}^0(x_i), \qquad i = 1, 2, \cdots, N, \tag{2.24a}$$

for both linear and cubic approximations, and additionally

$$\mathbf{U}_{x_i}^0 = \mathbf{u}_x^0(x_i), \qquad i = 1, 2, \cdots, N \tag{2.24b}$$

for cubic approximations. Here

$$\mathbf{U}_i^n := \mathbf{U}(x_i^n, t_n), \qquad \mathbf{U}_{x_i}^n := \mathbf{U}_x(x_i^n, t_n). \tag{2.25}$$

We obtain the approximate boundary conditions (2.6) by substituting (2.3) into (1.3), integrating the resulting equation from t_n to t_{n+1} and evaluating the integrals by the trapezoidal rule. Each boundary condition is associated with a particular partial differential equation in the vector system. The test space $\mathcal{V}_K$ is modified by setting the test functions ϕ_1 and ϕ_N (for linear approximations) or ϕ_{2N-1} (for cubic approximations) equal to zero for those partial differential equations associated with boundary conditions. This has the effect of replacing the Galerkin approximation of a partial differential equation at either $x=0$ or $x=a$ by its corresponding approximate boundary condition.

The system (2.6), (2.23) is a nonlinear algebraic system for determining $\mathbf{U}_i^{n+1}$, $i=1,2,\cdots,N$, for linear approximations or $\mathbf{U}_i^{n+1}$, $\mathbf{U}_{x_i}^{n+1}$, $i=1,2,\cdots,N$, for cubic approximations given the same information at $t=t_n$. We solve this nonlinear system by Newton's method, which requires the computation of the Jacobian of the vector $[\mathbf{F}(\mathbf{U},\phi_1), \mathbf{F}(\mathbf{U},\phi_2),\cdots,\mathbf{F}(\mathbf{U},\phi_K)]^T$ with respect to $[\mathbf{U}_1^{n+1},\mathbf{U}_2^{n+1},\cdots,\mathbf{U}_N^{n+1}]^T$ for linear approximations or $[\mathbf{U}_1^{n+1},\mathbf{U}_{x_1}^{n+1},\mathbf{U}_2^{n+1},\mathbf{U}_{x_2}^{n+1},\cdots,\mathbf{U}_N^{n+1},\mathbf{U}_{x_N}^{n+1}]^T$ for cubic approximations. The Jacobian will be block tridiagonal because of the local nature of ϕ_i. The elements in the ith block of rows will be the $M\times M$ matrices

$$\frac{\partial \mathbf{F}(\mathbf{U},\phi_i)}{\partial \mathbf{U}_j^{n+1}}, \qquad j=i-1,i,i+1, \tag{2.26a}$$

for linear approximations and the $2M\times 2M$ matrices

$$\begin{bmatrix} \dfrac{\partial \mathbf{F}(\mathbf{U},\phi_{2i-1})}{\partial \mathbf{U}_j^{n+1}} & \dfrac{\partial \mathbf{F}(\mathbf{U},\phi_{2i-1})}{\partial \mathbf{U}_{x_j}^{n+1}} \\ \dfrac{\partial \mathbf{F}(\mathbf{U},\phi_{2i})}{\partial \mathbf{U}_j^{n+1}} & \dfrac{\partial F(\mathbf{U},\phi_{2i})}{\partial \mathbf{U}_{x_j}^{n+1}} \end{bmatrix}, \qquad j=i-1,i,i+1, \tag{2.26b}$$

for cubic approximations. The elements of (2.26) are obtained from (2.22, 23) in a relatively straightforward manner, but their computation requires users of our code to supply subroutines that define $\mathbf{f}_\mathbf{u}(x,t,\mathbf{u},\mathbf{u}_x)$, $\mathbf{f}_{\mathbf{u}_x}(x,t,\mathbf{u},\mathbf{u}_x)$ and $\mathbf{D}_\mathbf{u}(x,t,\mathbf{u})$. Subroutines that define $\mathbf{f}(x,t,\mathbf{u},\mathbf{u}_x)$ and $\mathbf{D}(x,t,\mathbf{u})$ must, of course, also be supplied. We calculate and factor the Jacobian once per time step and use $\mathbf{U}(x,t_n)$ as an initial guess for $\mathbf{U}(x,t_{n+1})$. The linearized Newton system is solved by an efficient block tridiagonal algorithm that uses pivoting both within and outside of blocks (cf. Davis [13]). Generally, two iterations are performed per time step.

3. Adaptive mesh selection strategy.

In § 2 we developed a finite element method to obtain numerical solutions to systems of partial differential equations on nonuniform trapezoidal grids. In this section we construct an algorithm to select a grid at $t=t_{n+1}$ so that the L_2-norm of the error at t_{n+1} is approximately minimized. This algorithm builds upon the work of de Boor [14], Lawson [29] and Jupp [26] on variable knot spline interpolation.

For most of this section we will be discussing approximations at a single time level, say $t=t_n$, so whenever there is no possibility of confusion we omit the n superscript on x_i^n and $\mathbf{U}_i^n$ and suppress the t dependence when writing $\mathbf{u}(x,t)$. We also present the development for scalar functions u and indicate the extensions to vector functions in § 3.2.

It is well known (cf. [12], [38], [39]) that the errors in finite element-Galerkin methods for problems like (1.1–3) satisfy estimates of the form

$$\|u-U\|_{L_2} \leqq C\|u-\mathbf{P}U\|_{L_2}, \tag{3.1}$$

where $\mathbf{P}U \in \mathcal{U}_K$ interpolates u. Thus, the error in the solution of the partial differential equation is bounded by an interpolation error. The following result (cf., e.g., Pereyra and Sewell [34]) indicates how to minimize this interpolation error for piecewise polynomial interpolants.

LEMMA. *Let* $\Pi_N := \{0 = x_1 < x_2 < \cdots < x_N = a\}$ *be a partition of* $[0, a]$ *into* $N-1$ *subintervals, and let* $u(x) \in C^{l+1}[0, a]$. *The piecewise polynomial of degree* l *on* (x_i, x_{i+1}), $i = 1, 2, \cdots, N-1$, *that interpolates to* u *on* Π_N *has minimal* L_2*-error when the knots* x_i, $i = 2, 3, \cdots, N-1$, *are chosen such that*

$$h_i^{l+1}|u^{(l+1)}(\xi_i)| = E, \qquad i = 1, 2, \cdots, N-1, \tag{3.2}$$

where $u^{(l)}$ *is the* lth *derivative of* u *with respect to* x, $\xi_i \in (x_i, x_{i+1})$, E *is a constant and*

$$h_i = x_{i+1} - x_i. \tag{3.3}$$

The lemma states that the interpolation error is minimized by selecting the partition in such a way that the quantity $h_i^{l+1}|u^{(l+1)}(\xi_i)|$ is equidistributed. Considerable success has been achieved by using this result to implement adaptive grid algorithms for two-point boundary value problems (cf. Lentini and Pereyra [30], Ascher, Christiansen and Russell [2] or Russell and Christiansen [37]). Nevertheless, some practical difficulties still remain, and we discuss these and our solutions to them below.

Rather than work with (3.2) directly, we follow Lawson [29] and Jupp [26] and express (3.2) in the form

$$p_i := h_{i+1}/h_i = [|u^{(l+1)}(\xi_i)/u^{(l+1)}(\xi_{i+1})|]^{1/(l+1)}, \qquad i = 1, 2, \cdots, N-2, \tag{3.4}$$

where $l = 1$ for piecewise linear and $l = 3$ for piecewise cubic approximations.

In addition to (3.4), we impose the constraint that

$$h_1 + h_2 + \cdots + h_{N-1} = x_N - x_1. \tag{3.5}$$

This can be expressed in terms of the p_i's by defining

$$z := 1 + (p_1) + (p_1 p_2) + (p_1 p_2 p_3) + \cdots + (p_1 p_2 p_3 \cdots P_{N-2}) \tag{3.6}$$

and observing that

$$z = \frac{h_1 + h_2 + h_3 + \cdots + h_{N-1}}{h_1} = \frac{x_N - x_1}{h_1}. \tag{3.7}$$

Equations (3.3), (3.4), (3.6), (3.7) permit us to determine h_i, $i = 1, 2, \cdots, N-1$, and x_i, $i = 1, 2, \cdots, N$, in terms of $u^{(l+1)}$ without an explicit determination of E.

Of course, $u^{(l+1)}$ is unknown and must be approximated by $U^{(l+1)}$. The finite element procedure provides us with an approximate solution U and, for cubics, an approximate first derivative U_x. However, (3.4) requires a knowledge of second derivatives for linear approximations and fourth derivatives for cubic approximations. This situation typically arises in adaptive mesh algorithms, and it is usually resolved by using finite difference approximations for the necessary higher derivatives.

De Boor [14] used finite difference approximations to choose mesh points for the solution of two-point boundary value problems by assuming that the $(l+1)$st derivative was constant on each subinterval. We modify this scheme slightly by assuming that $U^{(l+1)}$ is linear on each subinterval and takes on the following values at the nodes:

$$U^{(l+1)}(x_i)=\begin{cases}\dfrac{\Delta U^{(l)}_{3/2}}{h_2+h_1}, & i=1,\\ \dfrac{2\Delta U^{(l)}_{3/2}}{h_2+h_1}, & i=2,\\ \dfrac{\Delta U^{(l)}_{i-3/2}}{h_{i-1}+h_{i-2}}+\dfrac{\Delta U^{(l)}_{i-1/2}}{h_i+h_{i-1}}, & i=3,4,\cdots,N-1,\\ \dfrac{2\Delta U^{(l)}_{N-3/2}}{h_{N-1}+h_{N-2}}, & i=N,\end{cases} \tag{3.8a}$$

where

$$\Delta U_i^{(l)} := (U_{i+1}^{(l)}-U_i^{(l)}). \tag{3.8b}$$

We use the approximation

$$U'_{i-1/2}=\frac{U_i-U_{i-1}}{h_{i-1}} \tag{3.9a}$$

for linear polynomials and

$$U'''_{i-1/2}=\frac{12(U_{i-1}-U_i)}{h^3_{i-1}}+\frac{6(U_{x_{i-1}}+U_{x_i})}{h^2_{i-1}} \tag{3.9b}$$

for cubic polynomials.

We note that p_i becomes infinite or indeterminate when $u^{(l+1)}(\xi_i)=0$ (cf. (3.4)); hence, we can expect numerical difficulties when $u^{(l+1)}(x)$ is small on any subinterval. Indeed, numerical experiments have shown that the mesh becomes very sensitive to small perturbations whenever $U^{(l+1)}(x)$ is of the same order of magnitude as the discretization error in the computed solution U.

We combat this problem by imposing a lower bound on $|U^{(l+1)}(x)|$. Thus, we let $\Delta t_n=(t_{n+1}-t_n)$ and $h=a/N$ denote the current time step and the average mesh spacing, respectively, and for linear approximations, we calculate $|U''(x_i)|$ as the maximum of the value computed by (3.8), (3.9) and $\max(\Delta t^2/h^2, h^2)$, while for cubic approximations, we calculate $|U^{(iv)}(x_i)|$ as the maximum of the value computed by (3.8, 9) and $12\max(\Delta t/h, h^2)+6\max(\Delta t^4/h^3, h^2)$. These limits were determined empirically. They are small enough so that they do not affect the mesh adaption procedure when $U^{(l+1)}(x)$ is not small but large enough to avoid the numerical difficulties caused by vanishing values of $U^{(l+1)}(x)$. Observe that if $U^{(l+1)}(x)$ is uniformly small on $[0,a]$ our limits assure that the solution of (3.3, 4, 6, 7) is a uniform mesh, as it should be in this case.

The discussion, thus far, has concerned the computation of an optimal grid at a time level t_n where the solution U^n has already been computed. We would also like to estimate an optimal grid at time level t_{n+1} prior to computing the solution there. This can be done by extrapolating the optimal grids computed at a number of previous time levels to t_{n+1}. It was somewhat surprising that numerical experiments seemed to favor zero order extrapolation; i.e., the optimal grid computed at time level t_n is used at time level t_{n+1}. Multilevel extrapolation consistently overestimated the distance that a mesh point should move in one time step and then overcorrected this error in the next time step. In some cases this caused the mesh to oscillate wildly when in fact the solution changed very little. When we simply extrapolated the optimal mesh determined at the previous time level, it tended to follow the solution even when rapidly moving fronts were present.

It is easy to show that the mesh selection strategy (3.3, 4, 5, 6) maintains the knot ordering so that no two mesh points can cross. It does not, however, prohibit severely distorted trapezoidal elements. Ciarlet and Raviart [12] and Babuska and Aziz [3] have studied the effect of element distortion on the accuracy of the finite element method. They have shown that the error obtained when computing on trapezoidal elements is a multiple of the error obtained when computing on rectangular elements. The multiplicative factor is proportional to a power of the magnitude of the derivatives of the transformation (2.8). Therefore, we must control the magnitude of these derivatives in order to maintain acceptable accuracy. We let

$$h_i^n = x_{i+1}^n - x_i^n, \quad \Delta t_n = t_{n+1} - t_n, \quad \tan \omega_i = \frac{x_i^{n+1} - x_i^n}{\Delta t_n}. \tag{3.10}$$

Hence, ω_i is the angle between the line $x_i(t)$ and the positive t axis.

Differentiating (2.8) and using (3.10), we find

$$\begin{aligned} x_\xi &= \frac{h_i^n + \tau(h_i^{n+1} - h_i^n)}{2}, \\ x_\tau &= \Delta t_n\left[\tan \omega_i + \frac{(\tan \omega_{i+1} - \tan \omega_i)(\xi + 1)}{2}\right], \\ t_\xi &= 0, \qquad t_\tau = \Delta t_n. \end{aligned} \tag{3.11}$$

Since the magnitudes of h_i^n and h_i^{n+1} are controlled by the bounds that we imposed on $|U^{(l+1)}|$ and Δt_n is prescribed, we can limit the magnitude of the derivatives in (3.11) by controlling the growth of $|\tan \omega_i|$. We found that the condition

$$\max_{1 \leq i \leq N} |\omega_i| \leq \frac{3\pi}{8} \tag{3.12}$$

worked well in practice.

3.1. Mesh selection algorithm. In this section we discuss some details of a mesh selection algorithm based on the discussion of the previous section. The first algorithm uses the finite element solution $U(x, t_n)$, calculated on the mesh x_i^n, $i = 1, 2, \cdots, N$. and (3.3, 4, 6, 7) to find a new mesh at $t = t_n$ that satisfies the optimality condition (3.2). This is the mesh that should have been used to calculate $U(x, t_n)$. Instead, we use it in the second algorithm to estimate an optimal mesh at $t = t_{n+1}$.

The difficulty in solving (3.3), (3.4), (3.6), (3.7) for the optimal mesh is that these equations are nonlinear and must be solved iteratively. We use a relaxation scheme that is similar to one which has been analyzed by Isaacson and Keller [24, Chap. 3]. They give necessary convergence criteria, but, we chose not to incorporate these into our algorithm because they require too much additional computation. The following algorithm, which calculates the relaxation parameter heuristically, has not failed to converge in any of our tests.

1. Set the relaxation parameter $\Omega := 1$, and let $x_i^{(0)} := x_i^n$, $i = 1, 2, \cdots, N$, be an initial guess for the optimal mesh. Calculate

$$z^{(0)} := (x_N^{(0)} - x_1^{(0)})/(x_2^{(0)} - x_1^{(0)}),$$

$$z^{(1)} := z + 2\varepsilon,$$

$$\nu := 1,$$

where ε is a convergence tolerance.

```
2. Compute
       $U^{(l+1)}(x_i^{(0)}), \quad i=1,2,\cdots,N,$
   using (3.8), (3.9).
3. **While** $|z^{(\nu)}-z^{(\nu+1)}|>\varepsilon$ **or** $\nu \leqq \nu_{\max}$ **do**
    4. Calculate $U^{(l+1)}(x_i^{(\nu-1)}), i=1,2,\cdots,N$, by linear interpolation of
       $U^{(l+1)}(x_i^{(0)}), i=1,2,\cdots,N$. Calculate
           $p_i^{(\nu)} := |U^{(l+1)}(x_{i+1}^{(\nu-1)})/U^{(l+1)}(x_{i+2}^{(\nu-1)})|^{1/(l+1)}, \quad i=1,2,\cdots,N-2$
       and
           $\hat{z}^{(\nu)} := 1+(p_1^{(\nu)})+(p_1^{(\nu)}p_2^{(\nu)})+\cdots+(p_1^{(\nu)}p_2^{(\nu)}\cdots p_{N-2}^{(\nu)}).$
    5. **If** $\nu>1$ **then**
           **If** $|\hat{z}^{(\nu)}-z^{(\nu-1)}| \geqq |z^{(\nu-1)}-z^{(\nu-2)}|$ **then** $\Omega := \Omega/2$.
    6. Calculate
           $h_1^{(\nu)} := (x_N^{(\nu-1)}-x_1^{(\nu-1)})/\hat{z}^{(\nu)},$
           $x_1^{(\nu)} := \hat{x}_1^{(\nu)} := x_1^{(\nu-1)},$
           $x_N^{(\nu)} := \hat{x}_N^{(\nu)} := x_N^{(\nu-1)},$
           $\left.\begin{aligned} \hat{x}_{i+1}^{(\nu)} &:= \hat{x}_i^{(\nu)}+h_i^{(\nu)} \\ h_{i+1}^{(\nu)} &:= h_i^{(\nu)}p_i^{(\nu)} \\ x_{i+1}^{(\nu)} &:= \Omega\hat{x}_{i+1}^{(\nu)}+(1-\Omega)x_{i+1}^{(\nu-1)} \end{aligned}\right\}, \quad i=1,2,\cdots,N-2,$
           $z^{(\nu)} := (x_N^{(\nu)}-x_1^{(\nu)})/(x_2^{(\nu)}-x_1^{(\nu)}),$
    7. $\nu := \nu+1$.
```

For vector systems we need only to change the definition of $p_i^{(\nu)}$ used in step 3. We used

$$p_i^{(\nu)} := \sum_{j=1}^{M} |U_j^{(l+1)}(x_{i+1}^{(\nu-1)})/U_j^{(l+1)}(x_{i+2})|^{1/(l+1)},$$

where U_j is the jth component of $\mathbf{U}$.

After we compute a convergent mesh, $\hat{x}_i^{n+1}=x_i^{(\nu)}$, $i=1,2,\cdots,N$, we perform the following:

```
1. Compute
       $\Delta x_{\max} = \max_{2\leqq i\leqq N-1} |\hat{x}_i^{n+1}-x_i^n|.$
2. **If** $\Delta x_{\max} \leqq \Delta t_n \tan(3\pi/8)$,
       **then** $\Delta x_{\text{fix}} := \Delta x_{\max}$
       **else** $\Delta x_{\text{fix}} := \Delta t_n \tan(3\pi/8)$.
3. Compute corrected mesh $\mathbf{x}^{n+1}$ as
       $x_i^{n+1} := x_i^n+(\hat{x}_i^{n+1}-x_i^n)\Delta x_{\max}/\Delta x_{\text{fix}}, \quad i=2,3,\cdots,N-1.$
```

Steps 2–3 prevent the elements from becoming too distorted.

The algorithms contain several approximations and heuristic procedures. Derivatives are estimated by differences and are assumed to vary linearly between mesh points. Zero order extrapolation was used to predict optimal grids at subsequent time levels. Grids were restrained to prevent severe element distortion. Even with these approximations, the mesh selection algorithms performed satisfactorily on all test examples that we considered. In addition, we note that Rheinboldt [35] has shown that an order Δ error in the placement of the optimal mesh only produces an order Δ^2 change in the computed solution. Thus, it suffices to be only close to the optimal mesh in order to reap its benefits.

4. Computation results. In this section we examine the performance of our method on four problems which are graded in difficulty such that each one exercises an additional facet of the method. The following norms are used to evaluate the performance of our method on examples where exact solutions are known.

$$\|e(t)\|_\infty := \max_{1\le i\le N} |\mathbf{e}(x_i, t)|_\infty := \max_{1\le i\le N} |\mathbf{u}(x_i, t) - \mathbf{U}(x_i, t)|_\infty, \tag{4.1a}$$

$$\|e(t)\|_{L_2}^2 := \sum_{i=1}^{N-1} \frac{h_i}{2}(|\mathbf{e}(x_i, t)|_\infty^2 + |\mathbf{e}(x_{i+1}, t)|_\infty^2), \tag{4.1b}$$

where

$$|\mathbf{v}|_\infty := \max_{1\le k\le M} |v_k|. \tag{4.1c}$$

Example 1.

$$\begin{aligned} u_t &= \left(\frac{1}{\pi}\right)^2 u_{xx}, \qquad 0<x<1, \quad t>0, \\ u(x, 0) &= \sin \pi x, \qquad u(0, t) = u(1, t) = 0. \end{aligned} \tag{4.2}$$

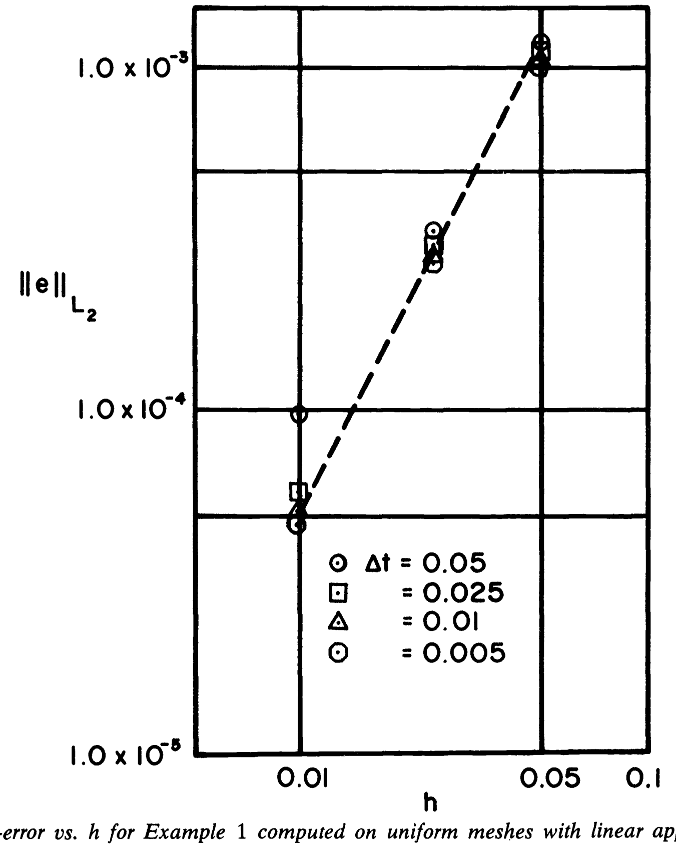


FIG. 2. *L_2-error vs. h for Example* 1 *computed on uniform meshes with linear approximations. The dotted line connects points for which $\Delta t = h$.*

The exact solution is

$$u(x, t) = e^{-t} \sin \pi x.$$

Analysis presented in [13] indicates that the finite element method described in § 2 would have L_2-error of $O(h^2) + O(\Delta t^2)$, with linear elements and $O(h^4) + O(\Delta t^2)$ with cubic elements on a uniform spacial mesh of width h and a uniform time step of duration Δt. We created this simple constant coefficient example to verify that these errors are actually attained. Figures 2 and 3 present plots of the L_2-error at $t = 1$ as a function of h for linear and cubic approximations, respectively.

The analysis of [13] predicts that the points on Fig. 2 for which $\Delta t = h$, and the points in Fig. 3, for which $\Delta t = h^2$, should lie on straight lines having slopes 2 and 4, respectively. These lines are shown confirming that the theoretical error bounds are actually attained.

Example 2.

(4.3a) $$u_t = \sigma u_{xx} + f(x), \qquad 0 < x < 1, \quad t > 0, \quad \sigma > 0.$$

The initial conditions, boundary conditions and source f are chosen so that the exact solution is

(4.3b) $$u(x, t) = \tanh (r_1(x - 1) + r_2 t).$$

The solution (4.3b) is a wave that travels in the negative x-direction when r_1 and r_2 are positive. The values r_1 and r_2 determine the steepness of the wave and its speed

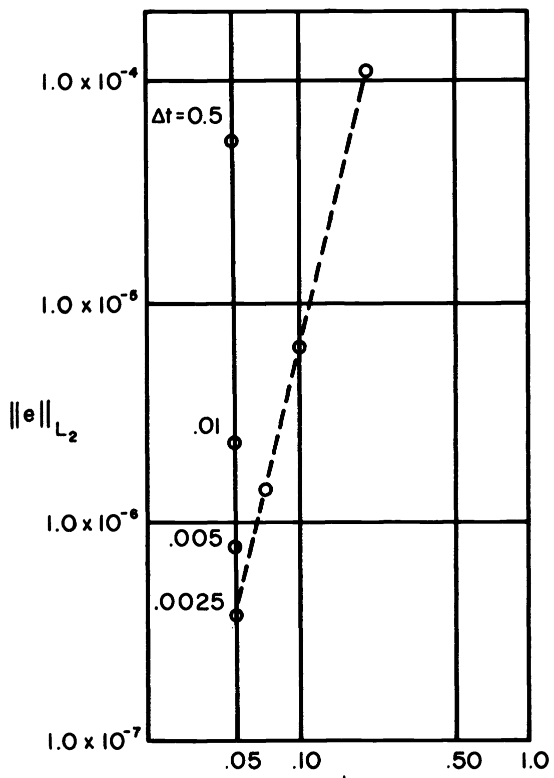


FIG. 3. *L_2-error vs. h for Example 1 computed on uniform meshes with cubic approximations. The dotted line connects points for which $\Delta t = h^2$.*

of propagation. Thus, the problem can be made more or less difficult by adjusting r_1 and r_2.

We created this problem to study the effectiveness of our adaptive mesh algorithm at concentrating grid points in transition regions, following moving fronts and reducing errors below those of uniform grid calculations.

We first solve problem (4.2) with $r_1 = r_2 = 5$, uniform time steps of $\Delta t = 0.01$, 10 elements per time step and linear approximations. The mesh computed by our adaptive mesh algorithm is shown in Fig. 4. The grid points are concentrated in the region of

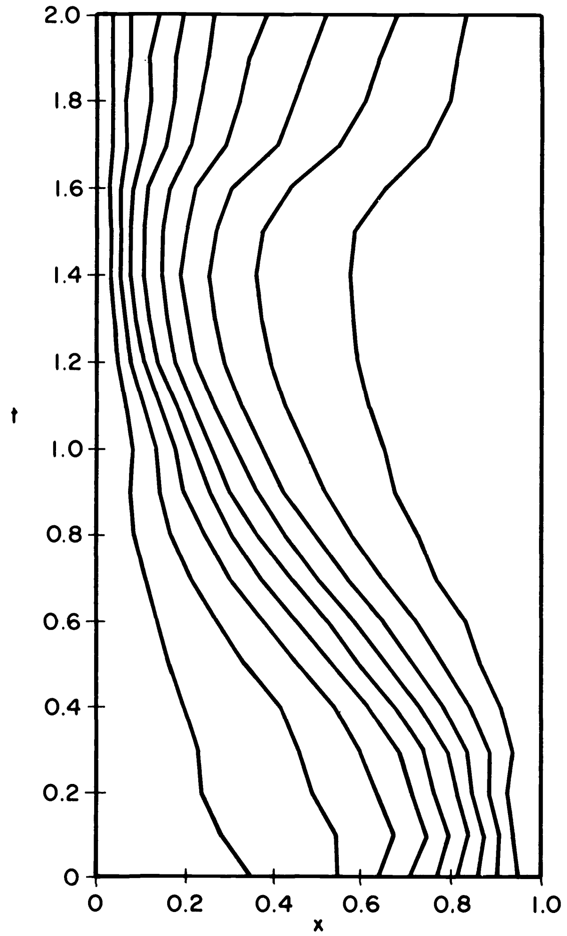


FIG. 4. *Mesh selected for Example 2 with $r_1 = r_2 = 5$, uniform time steps of $\Delta t = 0.1$, $N = 10$ and linear approximations.*

maximum curvature and move to the left with the wave. As the wave front passes out of the domain and u_{xx} becomes small, the grid points move toward a uniform distribution. It is clear that the grid adapts to the solution and follows its progress.

As a somewhat more difficult problem, we solve (4.3a) with initial conditions, boundary conditions and forcing function chosen so that the solution is given by (4.3b) with $r_1 = r_2 = 100$. The wave front is much steeper than in the previous test of (4.3).

In Table 1 we present a comparison at $t = 1$ of the results of computation using linear approximations on a variety of uniform and variably spaced meshes. These results are somewhat disappointing. At best, the mesh moving scheme improves the accuracy of the solution only slightly. The improvement is greater when Δt is small,

TABLE 1
Results of computations at $t = 1$ for Example 2 with $r_1 = r_2 = 100$ using linear approximations on uniform and variably spaced grids.

N	Δt	Uniform spacing		Variable spacing	
		$\|e\|_{L2}$	$\|e\|_\infty$	$\|e\|_{L2}$	$\|e\|_\infty$
10	0.1	0.168	0.137	0.459	1.346
	0.05	1.107	1.708	0.492	0.949
	0.01	0.146	0.254	0.121	0.340
20	0.1	0.365	1.391	0.567	1.00
	0.05	0.177	0.392	0.155	0.746
	0.01	0.768 E−1	0.226	0.166 E−1	0.870 E−1
40	0.025	0.367 E−1	0.697 E−1	0.348 E−1	0.565 E−1
	0.01	0.342 E−1	0.158	0.106 E−1	0.105
100	0.01	0.701 E−2	0.703 E−2	0.493 E−2	0.158 E−1
				0.144 E−2	0.275 E−2

and in some cases, when Δt is large, the uniform mesh is more accurate. A closer examination explains these results and reveals something about the nature of this mesh moving scheme.

Table 1 shows that the solution of this problem was not computed accurately with either a uniform or a variable mesh. This can be explained by examining the time evolution of the solution at a fixed value of x, say x^*. The solution is approximately given by -1 until the time when the wave reaches the point x^*. It then jumps suddenly to a value near 1. If the time step Δt is too large to resolve this transition, we would expect large errors in the vicinity of the wave. The solid curve in Fig. 5 confirms this

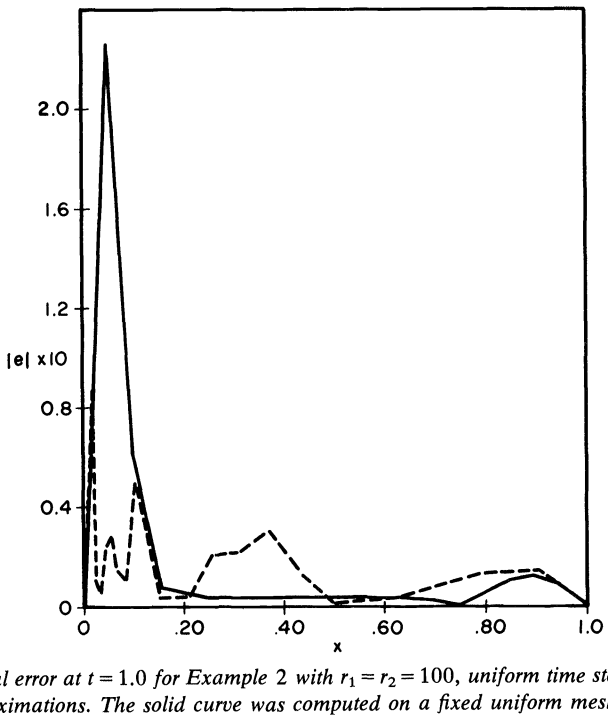


FIG 5. *Local error at $t = 1.0$ for Example 2 with $r_1 = r_2 = 100$, uniform time steps of $\Delta t = 0.01$, $N = 20$ and linear approximations. The solid curve was computed on a fixed uniform mesh, the broken curve on a variable mesh.*

prediction. The mesh selection procedure misinterprets these errors as being part of the solution and places too many points in the region outside of the wave front. Thus, a suboptimal mesh is selected and the expected decrease in the error is not obtained. When Δt becomes small enough to adequately resolve the passing wave, the mesh selection procedure does improve the accuracy of the solution (cf. Table 1).

This points out the need for an algorithm to adaptively refine time steps in the vicinity of severe temporal gradients. Such a procedure was used by Berger et al. [5] to solve hyperbolic partial differential equations, and we are currently studying its suitability for our code.

Table 2 summarizes the results of computations performed on the same problem using cubic approximations. In these cases the time steps Δt were small enough to resolve the transition of the solution and the cubic approximations were accurate enough to provide us with reasonable estimates of the derivatives. As a result, the variable mesh scheme improved the solution significantly.

TABLE 2

Results of computations at $t = 1$ for Example 2 with $r_1 = r_2 = 100$ using cubic approximations on uniform and variably spaced grids.

N	Δt	Uniform spacing		Variable spacing	
		$\|e\|_{L_2}$	$\|e\|_\infty$	$\|e\|_{L_2}$	$\|e\|_\infty$
10	0.01	0.607 E−1	0.801 E−1	0.130 E−1	0.232 E−1
14	0.005	0.319 E−1	0.257 E−1	0.332 E−2	0.602 E−2
20	0.01	0.214 E−1	0.394 E−1	0.167 E−1	0.951 E−1
	0.005	0.185 E−1	0.309 E−1	0.483 E−2	0.950 E−2
	0.0025	0.138 E−1	0.405 E−2	0.353 E−3	0.170 E−2

Figures 5 and 6 for linear and cubic approximations, respectively, show that the mesh selection algorithm tends to distribute the local error evenly over the domain, and thus, as indicated in § 3, approximately minimizes the error in L_2.

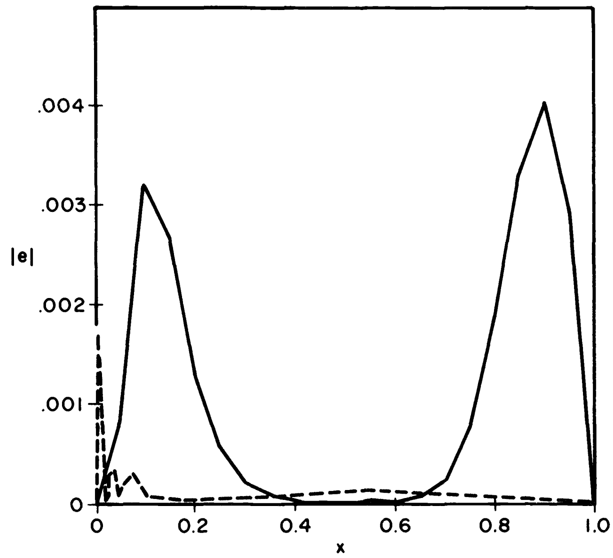


FIG. 6. *Local error at $t = 1.0$ for Example 2 with $r_1 = r_2 = 100$, uniform time steps of $\Delta t = 0.0025$, $N = 20$ and cubic approximations. The solid curve was computed on a fixed uniform mesh, the broken curve on a variable mesh.*

Example 3. (Burgers' equation).

$$u_t = -uu_x + \varepsilon u_{xx}, \qquad 0 < x < 1, \quad t > 0, \tag{4.4}$$
$$u(x, 0) = \sin \pi x, \qquad u(0, t) = u(1, t) = 0,$$

and $\varepsilon = 5 \times 10^{-3}$.

It is well known that the solution to this problem is a wave that steepens and moves to the right until a shock layer forms at $x = 1$. After a time of $O(1/\varepsilon)$, the wave dissipates and the solution decays to zero. Figures 7 and 8 show the results of computations on this problem using linear approximations on a uniform mesh and a variable mesh with a constant time step of $\Delta t = 0.1$ and 10 elements per time step. The results in Fig. 7 are typical of finite difference or finite element calculations for this problem (cf., e.g., Chin et al. [10]). Spurious oscillations develop in the computed solution unless the mesh width is of the same order as the width of shock layer, which is $O(\sqrt{\varepsilon})$ for this example. The variable mesh results in Fig. 8 largely suppress these oscillations by automatically concentrating the mesh in the shock region as the wave steepens.

When Example 3 is solved using cubic approximations on a uniform mesh, we find that the solution U_i^n at the nodes is computed accurately; however, there are large errors in the slope of the solution $U_{x_i}^n$ at the nodes when the mesh is not suitably fine in the shock region. This effect is exhibited in Fig. 9 where the solution at $t = 0.6$ is shown for a calculation performed with $\Delta t = 0.1$ and $N = 10$. Equations (2.16, 18, 19, 20) were used to calculate the solution between mesh points.

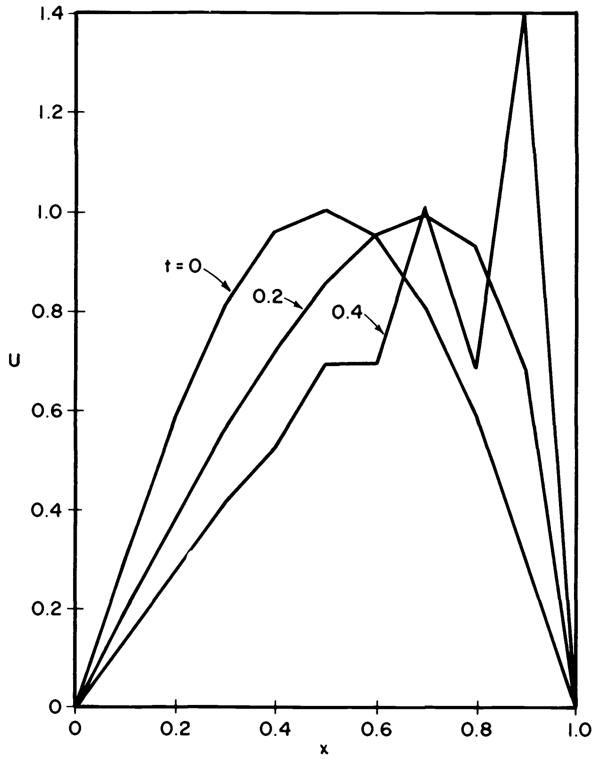


FIG. 7. *Solution of Example* 3 *for various values of t using linear approximations on a uniform mesh with* $\Delta t = 0.1$ *and* $N = 10$.

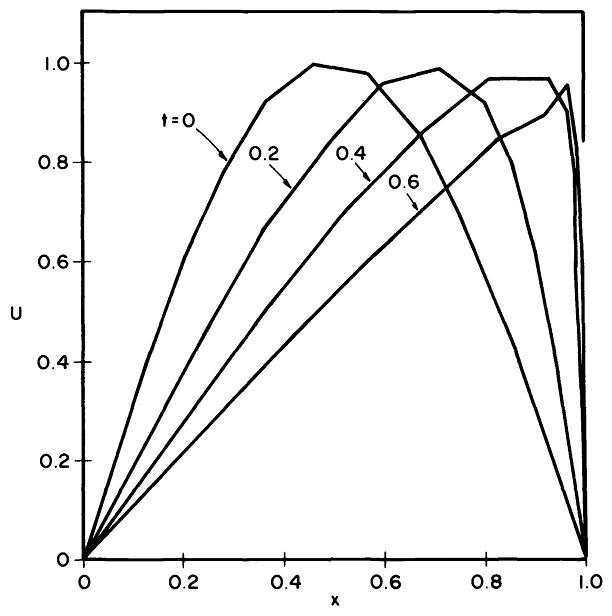


FIG. 8. *Solution of Example* 3 *for various values of t using linear approximations, uniform time steps of* $\Delta t = 0.1$ *and a variable mesh with* $N = 10$ *elements per time step.*

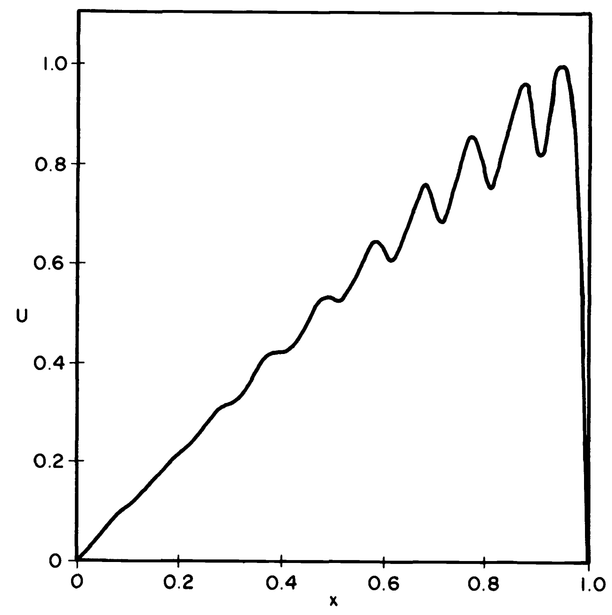


FIG. 9. *Solution of Example* 3 *at* $t = 0.6$ *using cubic approximations on a uniform mesh with* $\Delta t = 0.01$ *and* $N = 10$.

One possible explanation of this behavior was proposed by Miller and Miller [32], but they do not explain why the large errors in the slopes do not feed back and cause large errors in the function values.

Once again, these problems are corrected when the mesh adapts with the solution. Figure 10 shows the results of a similar computation using cubic approximations on a variable mesh. Both the function values and slope values are computed accurately at the nodes.

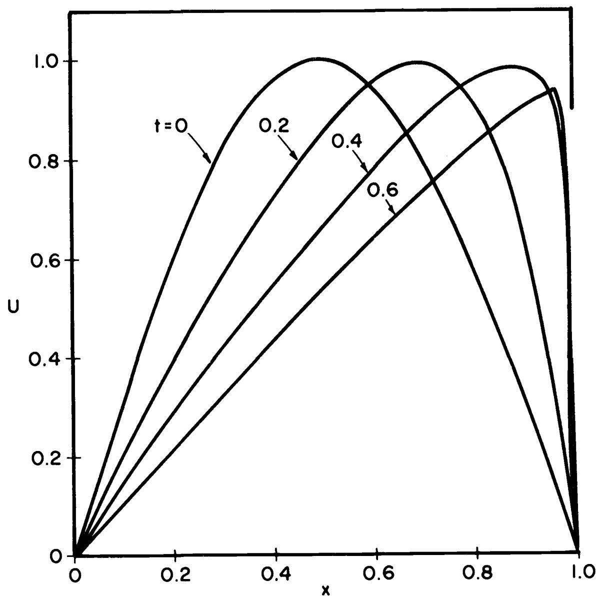


FIG. 10. *Solution of Example 3 for various values of t using cubic approximations, uniform time steps of $\Delta t = 0.01$ and a variable mesh with $N = 10$ elements per time step.*

Example 4.

$$b_t = [\mu(s)b_x]_x - [b\chi(s)s_x]_x, \qquad s_t = -k(s)b, \qquad 0 < x < 5, \quad t > 0. \tag{4.5}$$

This two-component nonlinear system was studied by Keller and Odell [28], [33] as a model for the chemotactic motion of bacteria. The quantity $b(x, t)$ denotes the bacterial density, and $s(x, t)$ denotes the concentration of the critical substrate (bacterial food). If the functions μ, χ and k satisfy conditions derived by Keller and Odell [28], equations (4.5) have traveling wave solutions. These solutions have been computed by Odell and Keller [33] and are interpreted as traveling bands of bacteria. For our study, we choose $k(s) = 1$, $\mu(s) = \mu_0$ and $\chi(s) = \delta_0/s$, where μ_0 and δ_0 are constants. The initial conditions are shown in Fig. 11a and the boundary conditions are

$$b(0, t) = b(5, t) = 0, \qquad s(0, t) = 1. \tag{4.6}$$

We solved this problem for $\delta_0/\mu_0 = 2$ using cubic approximations, uniform time steps of $\Delta t = 0.005$ and 50 elements per time step. The computed solutions at $t = 0$, 0.1, 0.5 and 1.0 are shown in Figs. 11a, b, c and d, respectively. The method places the majority of the mesh points in the regions of the wave fronts and follows the bacterial motion. The results indicate that our adaptive mesh algorithm may be also used for vector systems of equations.

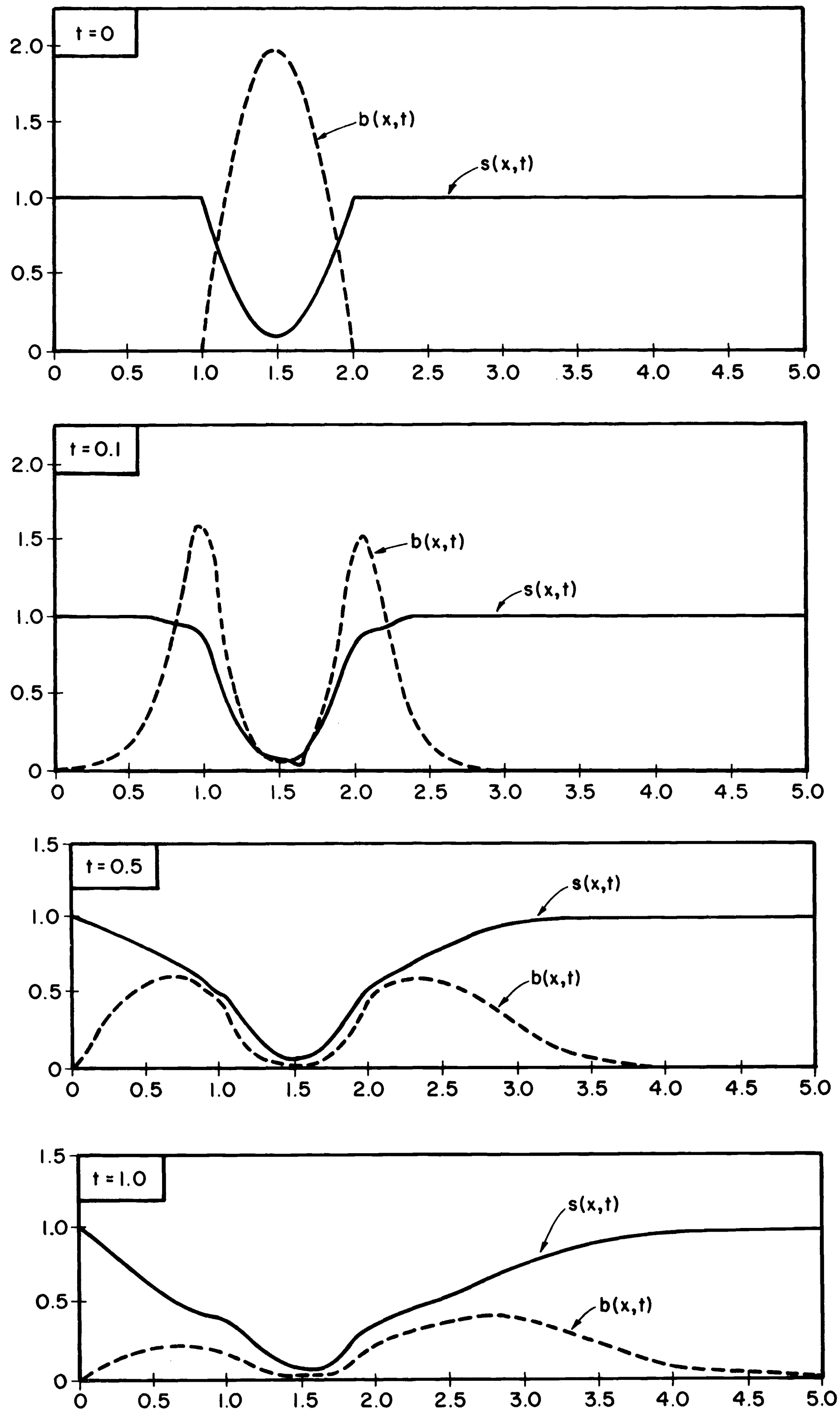


FIG. 11. *Computational results for Example* 4 *with* $\delta_0/\mu_0 = 2.0$ *using uniform time steps of* $\Delta t = 0.005$, $N = 50$ *and cubic approximations at* $t = 0$, 0.1, 0.5 *and* 1.0.

5. Discussion and conclusions. The computations presented in the last section show that it is possible to construct an accurate and stable adaptive grid finite element method for nonlinear systems of partial differential equations and that such techniques offer advantages over fixed grid techniques. In particular, we have shown that the error estimates obtained by Davis [13] are actually realized in practice and that the adaptive mesh algorithm correctly concentrates the mesh in a sharp transition and is able to follow moving fronts. Examples 3 and 4 of § 4 indicate that our method is also useful for nonlinear equations and vector systems of equations.

In the present study we used piecewise polynomial functions for both the trial and test spaces. However, work of Flaherty and Mathon [17], Heinrich et al. [21] and Hemker [22] indicates that exponential and "upwinded" polynomial functions may give superior test spaces for singularly perturbed problems. In addition, recent work of Chin and Krasny [11] indicates that there may be more efficient iterative procedures for equidistributing the mesh. We plan to test these potential improvements shortly.

All of our calculations were performed with a constant time step. Examples 3 and 4 of § 4 indicate that it would be most desirable to be able to vary the time step during the calculation. Our code presently allows for this, but as yet, we have not implemented an algorithm to adaptively alter the time step. We also plan to add this feature to our code shortly.

Other areas for future study include free boundary problems and higher dimensional problems. The present work has shown that it is possible to construct a practical adaptive grid finite element method. Future work must refine this method and apply it and test it on a greater variety of problems.

Acknowledgment. The authors would like to thank Dr. R. C. Y. Chin for his helpful comments and suggestions.

REFERENCES

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

[2] U. ASCHER, J. CHRISTIANSEN AND R. D. RUSSELL, *Collocation software for boundary value ODE's*, ACM Trans. Math. Software, 7 (1981), pp. 209–222.

[3] I. BABUSKA AND A. K. AZIZ, *On the angle condition in the finite element method*, SIAM J. Numer. Anal., 13 (1976), pp. 214–226.

[4] G. K. BATCHELOR, *An Introduction to Fluid Dynamics*, Cambridge University Press, Cambridge, 1970.

[5] M. BERGER, W. GROPP AND J. OLIGER, *Grid generation for time dependent problems: Criteria and methods*, Numerical Grid Generation Techniques, NASA Conference Publ. 2166, NASA Langley Research Center, Hampton, VA, October 1980, pp. 181–188.

[6] R. BONNEROT AND P. JAMET, *Numerical computation of the free boundary for the two-dimensional Stefan problem by space-time finite elements*, J. Comp. Phys., 25 (1977), pp. 161–181.

[7] A. BRANDT, *Multilevel adaptive technique (MLAT) for fast numerical solution to boundary value problems*, Lecture Notes in Physics 18, Springer-Verlag, New York, 1973, pp. 82–89.

[8] G. F. CAREY, *A mesh refinement scheme for finite element computations*, Comput. Meth. Appl. Mech. Engrg., 7 (1976), pp. 93–105.

[9] B. CHILDS, et al., eds., *Codes for Boundary Value Problems in Ordinary Differential Equations: Proceedings of a Working Conference, May* 14–17, 1978, Lecture Notes in Computer Science 76, Springer-Verlag, New York, 1979.

[10] R. C. Y. CHIN, G. W. HEDSTROM AND K. E. KARLSSON, *A simplified Galerkin method for hyperbolic equations*, Math. Comp., 33 (1979), pp. 647–658.

[11] R. C. Y. CHIN AND R. KRASNY, *A numerical method for stiff two-point boundary value problems*, to appear.

[12] P. G. CIARLET AND P. A. RAVIART, *Interpolation theory over curved elements with applications to finite element methods*, Comput. Meth. Appl. Mech. Engrg., 1 (1972), pp. 217–249.

[13] S. F. DAVIS, *An adaptive grid finite element method for initial-boundary value problems*, Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, NY, 1980.

[14] C. DE BOOR, *Good approximation by splines with variable knots* II, Conf. on the Numerical Solutions of Differential Equations, Lecture Notes in Mathematics 363, Springer-Verlag, New York, 1973.

[15] ———, *A Practical Guide to Splines*, Appl. Math. Sciences, 27, Springer-Verlag, New York, 1978.

[16] P. C. FIFE, *Singular perturbation and wave front techniques in reaction-diffusion problems*, SIAM-AMS Proceedings, 10 (1975), pp. 23–49.

[17] J. E. FLAHERTY AND W. MATHON, *Collocation with polynomial and tension splines for singularly perturbed boundary value problems*, this Journal, 1 (1980), pp. 260–289.

[18] A. FRIEDMAN, *The Stefan problem in several space variables*, Trans. Amer. Math. Soc., 133 (1968), pp. 51–87.

[19] R. J. GALINAS, S. K. DOSS AND K. MILLER, *The moving finite element method: Application to general partial differential equations with multiple large gradients*, J. Comp. Phys., 40 (1981), pp. 202–249.

[20] W. D. GROPP, *A test of moving mesh refinement for 2-D scalar hyperbolic problems*, this Journal, 1 (1980), pp. 191–197.

[21] J. C. HEINRICH, P. S. HUYAKORN, O. C. ZIENKIEWICZ AND A. R. MITCHELL, *An upwind finite element scheme for two-dimensional convective transport equations*, Int. J. Numer. Meths. Engrg., 11 (1977), pp. 131–143.

[22] P. W. HEMKER, *A numerical study of stiff two-point boundary problems*, Ph.D Dissertation, Mathematisch Centrum, Amsterdam, 1977.

[23] F. HOPPENSTEADT, *Mathematical Theories of Population: Demographics, Genetics and Epidemics*, CBMS Regional Conference Series in Applied Mathematics 20, Society for Industrial and Applied Mathematics, Philadelphia, 1975.

[24] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.

[25] P. JAMET AND R. BONNEROT, *Numerical solution of the Eulerian equations of compressible flow by a finite element method which follows the free boundary and interfaces*, J. Comp. Phys., 18 (1975), pp. 21–45.

[26] D. L. JUPP, *Approximation to data by splines with free knots*, SIAM J. Numer. Anal., 15 (1978), pp. 328–343.

[27] A. K. KAPILA, *Reactive-diffusive systems with Arrhenius kinetics: Dynamics of ignition*, SIAM J. Appl. Math., 39 (1980), pp. 21–36.

[28] E. F. KELLER AND G. M. ODELL, *Necessary and sufficient conditions for chemotactic bands*, Math. Biosci., 27 (1975), pp. 309–317.

[29] C. L. LAWSON, *Segmented rational minmax approximations, characteristic properties and computational methods*, J. P. L. Tech. Rept., pp. 32–57, 1963.

[30] M. LENTINI AND V. PEREYRA, *An adaptive finite difference solver for nonlinear two-point boundary problems with mild boundary layers*, SIAM J. Numer. Anal., 14 (1977), pp. 91–111.

[31] J. N. LYNESS AND J. J. KAGANOVE, *Comments on the nature of automatic quadrature routines*, ACM Trans. Math. Software, 2 (1976), pp. 65–81.

[32a] K. MILLER AND R. MILLER, *Moving finite elements*, I, SIAM J. Numer. Anal., 18 (1981), pp. 1019–1032.

[32b] K. MILLER, *Moving finite elements*. II, SIAM J. Numer. Anal., 18 (1981), pp. 1033–1057.

[33] G. M. ODELL AND E. F. KELLER, *Traveling bands of chemotactic bacteria revisited*, J. Theoret. Biol., 56 (1976), pp. 243–247.

[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] W. C. RHEINBOLDT, *Adaptive methods in numerical analysis*, Invited lecture, SIAM 1979 Spring Meeting, June 11–13, Toronto, Canada.

[36] J. R. RICE, *A meta algorithm for adaptive quadrature*, J. Assoc. Comput. Mach., 22 (1975), pp. 61–82.

[37] R. D. RUSSELL AND J. CHRISTENSEN, *Adaptive mesh strategies for solving boundary value problems*, SIAM J. Numer. Anal., 15 (1978), pp. 59–80.

[38] G. STRANG AND G. J. FIX, *An analysis of the finite element method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[39] M. F. WHEELER, *A priori, L_2-error estimates for Galerkin approximations to parabolic partial differential equations*, SIAM J. Numer. Anal. 10 (1973), pp. 723–759.

[40] A. B. WHITE, JR., *On the numerical solution of initial/boundary value problems*, SIAM J. Numer. Anal. 19 (1982), to appear.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 1, March 1982

A SPLINE LEAST SQUARES METHOD FOR NUMERICAL PARAMETER ESTIMATION IN DIFFERENTIAL EQUATIONS*

J. M. VARAH†

Abstract. In this paper, we describe a straightforward least squares approach to the problem of finding numerical values for parameters occurring in differential equations so that the solution best fits some observed data. The method consists of first fitting the given data by least squares using cubic spline functions with knots chosen interactively, and then finding the paramters by least squares solution of the differential equation sampled at a set of points. We illustrate the method by four problems from chemical and biological modeling.

Key words. parameter estimation, spline fitting, differential equations

1. Introduction. The general problem can be stated as follows: we are given a system of ordinary differential equations

$$y_1' = f_1(t, \mathbf{y}, \mathbf{p}),\quad \vdots \quad y_n' = f_n(t, \mathbf{y}, \mathbf{p}), \tag{1.1}$$

where $\mathbf{p} = (p_1, \cdots, p_m)$ are m (real) parameters whose numerical values are unknown. Also, the solution vector $\mathbf{y}(t) = (y_1(t), \cdots, y_n(t))$ has been measured at certain data points $\{t_i, i = 1, \cdots, N\}$. The problem is to find reasonable values for $\mathbf{p}$ so that the solution of (1.1) with these parameter values, and suitably chosen initial conditions, fits the given data.

As a specific example, consider the Lotka/Volterra predator–prey model (see, e.g., Clark (1976, p. 194))

$$y_1' = p_1 y_1 - p_2 y_1 y_2, \qquad y_2' = p_3 y_1 y_2 - p_4 y_2.$$

Here $y_1(t)$ measures prey population, $y_2(t)$ predator population, and the $\{p_i\}$ are positive constants dealing with birth, death and interaction rates. Typically, there are measured values for $y_1(t)$, $y_2(t)$ at certain times $t = t_i$, and we are asked to provide reasonable values for the $\{p_i\}$ so that the solution $(y_1(t), y_2(t))$ approximates the data. Of course, there is a certain amount of error inherent in the data, so we cannot expect to fit the data perfectly. Furthermore, there may or may not be exact initial conditions given; the first point $(y_1(t_1), y_2(t_1))$ may be just as much in error as the other points.

Such problems arise frequently in various areas, for example, in chemical reaction equations, and in the modeling of biological and ecological processes. In this paper, we describe a simple approach to the problem, discuss its merits relative to other methods which have been proposed, and illustrate the method on four specific examples.

2. Method of solution. Recently, the most popular method for solving this problem has been an initial value technique: initial estimates of the parameters are made, and (1.1) is integrated using these parameters and some (possibly given) set of initial conditions. Then the least squares deviation of the solution at the data points is measured, and this is treated as a function of the parameters, which one tries to

* Received by the editors November 25, 1980, and in revised form June 24, 1981.

† Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada V6T 1W5.

minimize over the space of parameter values. This approach is described in Bard (1974), van Domselaar and Hemker (1975) and Benson (1979). Although it can work well, we feel it has some serious drawbacks:

(a) The solution of (1.1) may be very sensitive to the initial conditions, thus making it difficult to integrate the equations. This sensitivity will be worsened in cases where the initial conditions are not known accurately; often the data error is at least 10%.

(b) The technique requires guessing the parameter values; if these are not known reasonably well in advance, again it may be difficult to integrate the equations, and the behavior of the solution may be totally different from that obtained using "good" parameters.

(c) There is a large amount of computational work involved; each new set of parameters requires a full-scale integration of the equations, possibly with a special method (e.g., if the initial value system is stiff). This all means that the computer programs set up to solve such a problem are by necessity long and complex.

(d) When the parameters occur *linearly* in (1.1), as in the Lotka/Volterra model, the method does not simplify: an iterative procedure still ensues.

We would like to propose a simple, straightforward method which, we feel, overcomes all of these drawbacks.

(1) First, fit the given data by least squares using cubic spline functions. That is, for each component $j=1,\cdots,n$, construct a cubic spline $s_j(t)$ with fixed knots $\{t_k^*\}$, $k=1,\cdots,q$, choosing the spline coefficients to minimize the least squares deviation at the data points. This technique is described in de Boor (1979, Chapt. 14) and we give some further details in the next section, as it is useful for data fitting in general. The number and position of the knots $\{t_k^*\}$ are adjusted adaptively, preferably using interactive graphics, until a "good" spline fit is obtained. We *assume* that this can be done: that is, that the user is a good judge of whether a given spline fit represents the data properly.

(2) When these spline fits have been found (so that the data has in effect been *smoothed*), we then find parameters to minimize the least squares deviation in the differential equation system (1.1) measured at some set of *sample points* $\{\hat{t}_i\}$, $i=1,\cdots,M$. That is, we find $\mathbf{p}$ to

$$\min_{\mathbf{p}} R_D^2(\mathbf{p})=\sum_{j=1}^{n}\sum_{i=1}^{M}[s_j'(\hat{t}_i)-f_j(\hat{t}_i,\mathbf{s},\mathbf{p})]^2. \tag{2.1}$$

In particular, notice that when the parameters $\mathbf{p}$ appear linearly in (1.1), this is a linear least squares problem, which can be solved directly by setting up the overdetermined system of nM equations in m variables ($\mathbf{p}$) and solving this by a QR factorization or by using the normal equations. Moreover, no initial value solver is needed and no specific initial conditions are required. Thus the amount of computation, and the complexity of the program needed, are much less than for the initial value method described earlier. We should add that this technique is not new: a similar method (using a different data fitting technique in (1)) was proposed by Swartz and Bremerman (1975), and other similar methods were probably proposed earlier.

We should add that, strictly speaking, the wrong least squares sum is being minimized: instead of R_D in (2.1), we are really interested in minimizing the integrated residual

$$R_I^2(\mathbf{p})=\sum_{j=1}^{n}\sum_{i=1}^{M}(y_j(t_i)-\bar{y}_{ij}(\mathbf{p}))^2, \tag{2.2}$$

where $\bar{y}_{ij}(\mathbf{p})$ is the numerical approximation to $y_j(t_i)$ obtained by integrating the DE system (1.1) with parameters $\mathbf{p}$. However, to do so would mean we would have to integrate the DE system for each parameter choice, and this is precisely what we are trying to avoid. Our purpose here is to present an efficient algorithm which produces *reasonable* values for the parameters. Of course, the parameters found should be checked, a posteriori, by integrating the DE system with these values to check the residual R_I. We have done this in the examples. In all the examples we tried, the DE residual (R_D) and the integrated residual (R_I) were minimized at roughly the same parameter values, but this is indeed open to question. However, we expect that with enough sample points and enough data points, this will be the case. In Mezaki, Draper and Johnson (1973), an example is given where the use of two different minimizing functions produces two different minima, but these functions are not so closely related as R_D and R_I.

3. Least squares cubic splines. To simplify the notation, assume we have only one y-component, so our data are $\{(t_i, y_i), i=1,\cdots,N\}$. Also let $a \leqq t_1 \leqq t_2 \leqq \cdots \leqq t_N \leqq b$. We wish to approximate this by a cubic spline $s(t)$ with knots $\{t_i^*\}$, $a < t_1^* \leqq t_2^* \leqq \cdots \leqq t_q^* < b$. Thus the spline $s(t)$ is made up of different cubic polynomials in each interval (a, t_1^*), (t_1^*, t_2^*), $\cdots$, (t_q^*, b), matched at the knots so that $s''(t)$ is continuous throughout (a, b). Since each cubic polynomial has four coefficients, and the above requires three continuity conditions per knot, we are left with $(q+4)$ coefficients to determine by least squares solution of the data equations $s(t_i) = y_i$, $i = 1, \cdots, N$.

Of course, one can solve directly by expressing each cubic polynomial in powers of t, matching the continuity conditions at the knots and solving the least squares problem for the other coefficients. However, it is much easier (technically, if not conceptually) to use a B-spline basis for the cubic splines. Each cubic B-spline $B_i^{(4)}(t)$ is uniquely defined by 5 successive knots, and is positive inside and zero outside this range. They can be easily generated by the recurrence relation (see de Boor (1979, p. 131)):

$$B_i^{(1)}(t) = \begin{cases} 1, & t_i^* \leqq t \leqq t_{i+1}^*, \\ 0, & \text{otherwise}, \end{cases}$$

$$B_i^{(k)}(t) = \frac{t - t_i^*}{t_{i+k-1}^* - t_i^*} B_i^{(k-1)}(t) + \frac{t_{i+k}^* - t}{t_{i+k}^* - t_{i+1}^*} B_{i+1}^{(k-1)}(t), \qquad k = 2, 3, 4. \tag{3.1}$$

Notice that this generates, in order, the B-spline of degree 1, 2 and 3 over the appropriate set of knots. To make this complete, the endpoints a and b must be included as 4-fold multiple knots. Since there are q interior knots, this defines $(q+4)$ B-splines, which then form a *basis* for all cubic splines over these knots in the interval (a, b). To find the best least squares spline, we solve the overdetermined linear system

$$\sum_{j=1}^{q+4} a_j B_j^{(4)}(t_i) \cong y_i, \qquad i = 1, \cdots, N.$$

This is the data fitting technique we advocate here. The knots $\{t_k^*\}$ must be chosen fairly carefully in order to get a reasonable fit of the data with not too many knots. It is best to do this interactively, using a graphics terminal if possible. We should also remark that the knots may be multiple; a double knot, for example, allows the second derivative of the spline to be discontinuous. This can be helpful in fitting data which change abruptly.

Clearly it is important that the derivative $s'(t)$ be a reasonable approximation to the rate of change of the given data. As is well known, this can be a tricky business, and it is for this reason that we have used cubic splines. Our experience indicates that cubic splines give as good a derivative as can be expected from the data. As an indication of this, when we tried the method on the example of Anderssen and Bloomfield (1974), we obtained the same accuracy as they did for their Fourier or regularization method. Cubic smoothing splines are also a possibility; see de Boor (1979, p. 235).

4. Details of the method. To succeed, our technique should be used in an interactive environment. When the data are first fitted with a cubic spline, it is very important that a graph of the fit be plotted. For a given knot set $\{t_k^*\}$, we may obtain a small least squares residual at the data points, and yet the spline may deviate considerably from the "expected curve" between the data points.

With an interactive plot, the knots $\{t_k^*\}$ can be adjusted to get a better "visual" or overall plot. Similarly, although one could automate the choice of knots by minimizing the least squares residual at the data points over all possible knot sets $\{t_k^*\}$, such a choice will not necessarily produce the "best" fit, because of possible large oscillations between data points and because the final curve we are after is not the spline fit, but a solution curve of the differential equation. This is borne out in our examples in § 6.

Similarly, the choice of sample points $\{\hat{t}_i\}$ is somewhat arbitrary, and should be done interactively. Enough points should be chosen that the behavior of the solution is adequately represented, and it is important to place sample points "where the action is", that is, where the solution is changing rapidly. It has been our experience that a *reasonable* selection of sample points $\{\hat{t}_i\}$ and knots $\{t_k^*\}$ is what is important, not their exact placement; that is, the method is fairly robust.

After the choice of sample points, the next step is the solution of (2.1). As we have mentioned, this is a linear least squares problem if the parameters appear linearly in (1.1) and, if so, the problem can be solved directly using normal equations or a *QR* factorization (which is, of course, required for the spline fit earlier). If the parameters appear nonlinearly, however, there are many techniques, algorithms and programs available. We have contented ourselves with using a simple direct search algorithm for nonlinear minimization, which has the advantage of not requiring any partial derivative $\partial f_i/\partial p_j$. However, much more sophisticated techniques are available for such nonlinear least squares problems and should probably be used, since these partial derivatives are required in any case if we want to obtain confidence intervals for the parameters by solving the sensitivity equations (see (5.1) below). Particular methods are Levenberg–Marquardt (which is available in various implementations) and the algorithm of Dennis–Gay–Welsch (1979), which was designed for large residual problems. For a survey of such methods, see Nazareth (1980).

There is also an intermediate case which often arises: some of the parameters can appear linearly and some nonlinearly in (1.1). In this case one can use the idea of separability or variable projection (see Golub and Pereyra (1973) or Ruhe and Wedin (1980)), in which the linear parameters are implicitly solved for, the resulting (fully) nonlinear least squares problem is solved for the nonlinear parameters, and then the linear parameters are obtained using their representation in terms of the nonlinear parameters. Since this reduces the size of the nonlinear least squares problem to be solved, it is worthwhile.

Once the parameters $\mathbf{p}$ have been found, their validity should be checked by integrating the system (1.1) using these values. As we indicated earlier, this can present difficulties, particularly if exact initial conditions are not given, as is the case in our first two examples in § 6. It is important to realize that this causes real difficulties with the initial value techniques mentioned in the introduction: the inexact initial conditions must be included as parameters, thus increasing the dimension of the parameter space. Moreover, the initial value problem may not have a solution for some of the initial conditions used as parameter values during the course of the algorithm.

Here we are in a better situation: we have our parameter estimate ($\mathbf{p}^*$ say) and we are only concerned with estimating the integrated residual $R_I(\mathbf{p}^*)$ properly. We could merely use the given values $y_j(t_1)$, $j = 1, \cdots, n$, as initial conditions and integrate once; however, we feel that in some cases it is better to allow the initial values to vary slightly, and choose those values which minimize the integrated residual. This process is, or course, itself a linear or nonlinear least squares problem, involving integrations of the DE system at each step, but there are only n parameters involved and we have good estimates of them, so that only a few iterations are required for the minimization.

We used this technique for the first two examples in § 6, and the corresponding values of $R_I(\mathbf{p}^*)$ did drop substantially. For the other examples, however, the variation was insignificant.

Finally, we mention a special difficulty which arises in the third example in § 6, wherein data are only given for *some* of the components. This causes great hardship for any method, including ours, since there is no data to fit the spline to (for some components)! In our example, we can finesse the difficulty by converting the 2×2 system into a single equation of second order involving only that variable for which we have data measurements. Notice that this means we must also use second derivative estimates of the fitted spline curve (and (2.1) changes accordingly). In general, this only works if the nondata variables can be explicitly solved for, and even then necessitates approximating higher derivatives of the data variables, possibly using higher degree splines.

5. The sensitivity equations. After having obtained estimates of the parameters $\mathbf{p}$ and the solution vector $\mathbf{y}(t)$, one can obtain estimates of the sensitivity and accuracy of the parameters. Define $Z_{ij} = \partial y_i / \partial p_j$, $i = 1, \cdots, n$, $j = 1, \cdots, m$. Differentiating (1.1), it is easy to see that Z satisfies the first order linear equation

$$Z' = G(t, \mathbf{y}, \mathbf{p}) + J(t, \mathbf{y}, \mathbf{p})Z, \tag{5.1}$$

where $G_{ij} = \partial f_i / \partial p_j$ and $J_{ij} = \partial f_i / \partial y_j$. These are the sensitivity equations, and one can obtain estimates of the sensitivity of the solution to changes in the parameters (i.e., $Z(t)$) by integrating (5.1) using the computed values for $\mathbf{p}$ and $\mathbf{y}$. This is quite well known (see Bard (1974) for example), although it is not clear to this author what initial conditions should be used for (5.1). If *exact* initial conditions are specified, then $Z_{ij} = 0$ is appropriate and seems to always be used; however, if the initial conditions are not known exactly, then it is not clear that $Z_{ij} = 0$ should be used. For this reason, we have not attempted to compute $Z(t)$ in our examples.

Once $Z(t)$ has been computed, *confidence intervals* for the parameters can be obtained in the usual way, by assuming that the least squares function

$$\Phi(\mathbf{p}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{n} (y_j(t_i) - y_{ij})^2$$

is locally quadratic near the minimum $\mathbf{p}^*$. Then if the uncertainty or noise level in ϕ is ε^2, the confidence intervals are of the form

$$|p_i - p_i^*| \leqq \varepsilon \sqrt{H_{ii}^{-1}},$$

where $H = H(\mathbf{p}^*, y)$ is the Hessian matrix at the minimum. This Hessian consists of two terms:

$$H_{kl} = \sum_{i=1}^{N} \sum_{j=1}^{n} Z_{jl}(t_i) Z_{jk}(t_i) + \sum_{i=1}^{N} \sum_{j=1}^{n} (y_j(t_i) - y_{ij}) \frac{\partial^2 y_j(t_i)}{\partial p_k \, \partial p_l}.$$

Usually the second term is considered negligible so that H can be computed directly from $Z(t)$: $H = \sum_{i=1}^{N} Z^T(t_i) Z(t_i)$. This is reasonable in cases where the residual is fairly small, or the problem is (nearly) linear in $\mathbf{p}$. In other cases, however, the second term can materially affect H, and it is probably wise to use a nonlinear least squares routine which computes an approximate Hessian as the minimization proceeds (again, see Nazareth (1980)). Near-singularity of the Hessian can be caused by (for example) nearly linearly dependent parameters, or insufficient data to separate the parameters. In any case, this indicates the problem is poorly conditioned and should be revised.

6. Numerical Examples.

A. *Barnes' problem* (see van Domselaar and Hemker (1975)—referred to as VDH).

$$y_1' = p_1 y_1 - p_2 y_1 y_2, \qquad y_2' = p_2 y_1 y_2 - p_3 y_2.$$

TABLE 1

t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
y_1	1.0	1.1	1.3	1.1	0.9	0.7	0.5	0.6	0.7	0.8	1.0
y_2	0.3	0.35	0.4	0.5	0.5	0.4	0.3	0.25	0.25	0.3	0.35

This problem as originally given represented chemical reaction equations; however, it is also the well known Lotka/Volterra predator–prey model in ecology (see, e.g., Clark (1976, p. 194)). The solution components to this system are oscillatory in nature and out of phase with each other. Moreover, the data are only accurate to about 10%, so it is clear that we should not try too hard to fit the data closely.

Table 2 displays a sample of results obtained, using up to four knots in the spline approximation.

TABLE 2

Knot positions	Spline residuals	DE residual (# sample points)	Parameters found	Integrated residual	Initial conditions
3.0	0.16, 0.11	1.3 (20)	0.85, 2.13, 1.91	0.35	1.02, 0.25
3.0	0.16, 0.11	1.7 (40)	0.80, 2.06, 1.86	0.36	1.05, 0.26
1.5, 3.0	0.14, 0.04	1.0 (20)	0.85, 2.20, 2.04	0.38	1.02, 0.24
1.5, 3.0	0.14, 0.04	1.5 (40)	0.83, 2.17, 2.01	0.37	1.04, 0.24
*0.4, 2.5	0.10, 0.09	3.0 (40)	0.62, 1.73, 1.60	0.44	1.16, 0.29
0.9, 2.1, 3.6	0.11, 0.04	1.6 (40)	0.80, 2.11, 1.94	0.36	1.05, 0.24
1.0, 2.0, 3.0, 4.0	0.09, 0.02	1.6 (40)	0.85, 2.21, 2.02	0.365	1.02, 0.24
*0.14, 0.97, 3.2, 3.9	0.06, 0.01	40.0 (40)	−0.01, 2.19, 1.47	1.9	1.02, 0.24

The VDH values were (0.86, 2.07, 1.81) with $R_I = 0.40$. Knot selection was made visually, except for the cases marked with an asterisk; these knots were found by minimizing the least squares deviation of the spline fit for the first component. Notice that although these gave better spline fits, the corresponding parameter values were very poor. In some sense, we are trying "too hard" to fit the data, and produce a curve which is not close to an integral curve of the differential equations.

This problem is linear in $\mathbf{p}$, so the minimization (2.1) is a linear least squares problem. Its residual is given in the third column; we used twenty (or forty) equally spaced sample points. In each case, the parameter values were checked by integrating the system from the first data point ($t = 0$) and varying the initial condition so as to obtain the smallest least squares deviation in the integrated residual. These results are given in the last two columns. We also give plots of the first one knot spline fit and the corresponding integration in Figs. 1 and 2 (for the first component y_1) and for both four knot cases in Figs. 3–4 and 5–6.

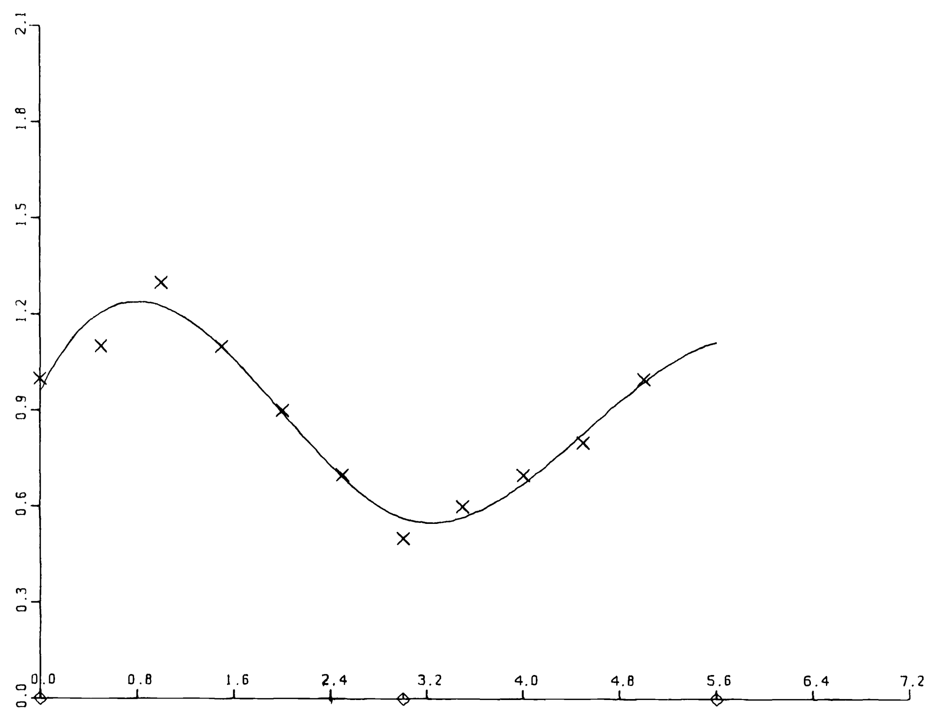


FIG. 1. *Example A—One-knot spline fit.*

The Hessian matrix for this problem is not ill-conditioned, so in that sense the problem is well-conditioned. The rather large variation in parameter values obtained (for about the same residual) is due to the inaccuracy in the data, and the correspondingly large residual (relative to the size of the data).

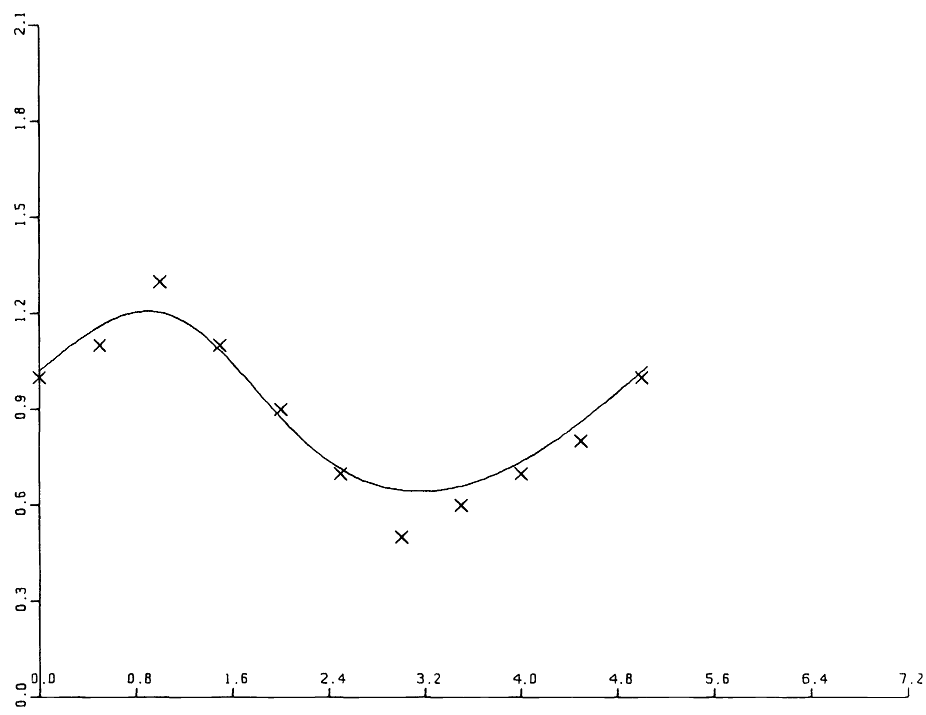


FIG. 2. *Example* A—*Integration using one-knot spline fit.*

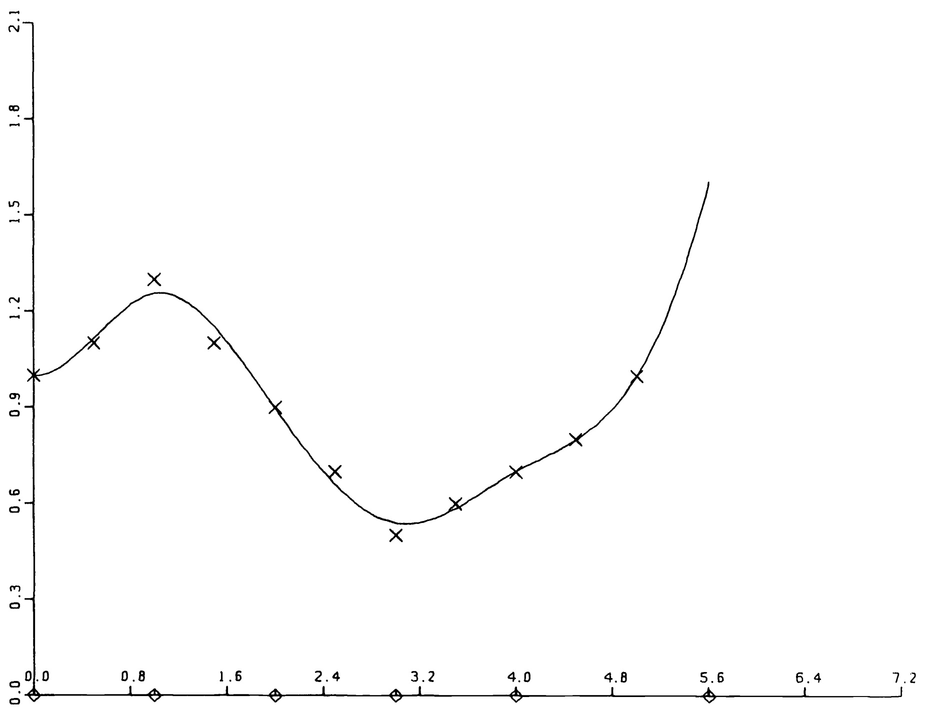


FIG. 3. *Example* A—*Visual four-knot spline fit.*

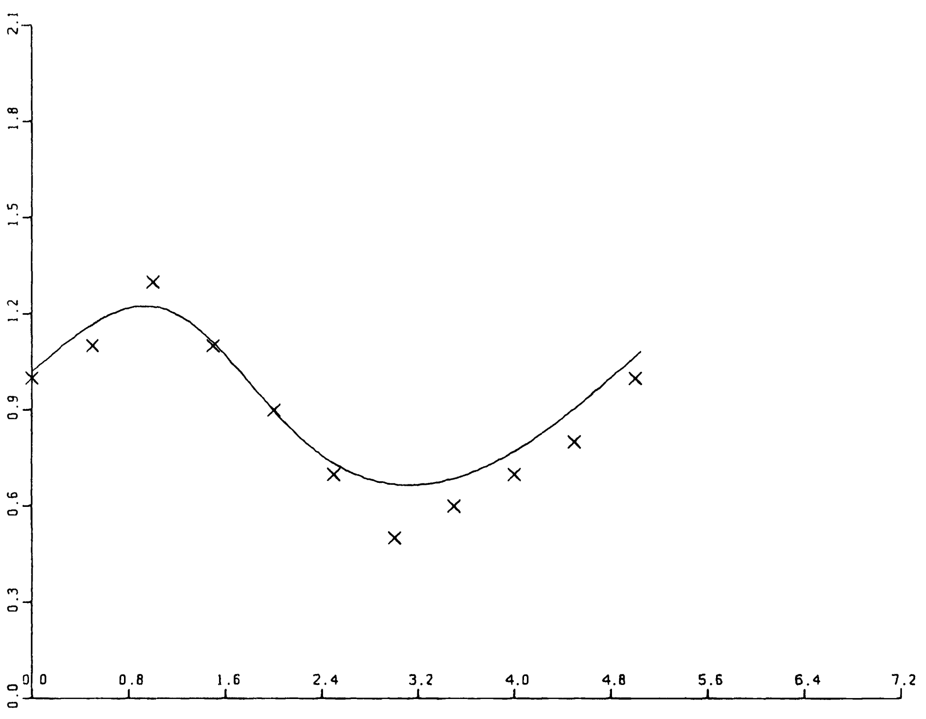


FIG. 4. *Example A—Integration using visual four-knot spline fit.*

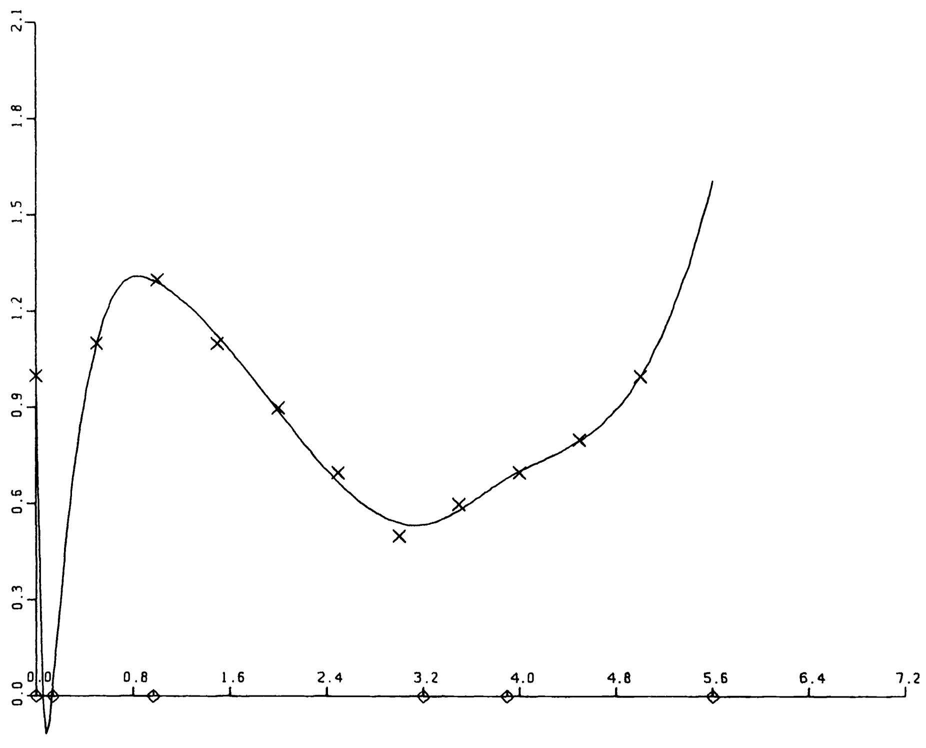


FIG. 5. *Example A—Optimal four-knot spline fit.*

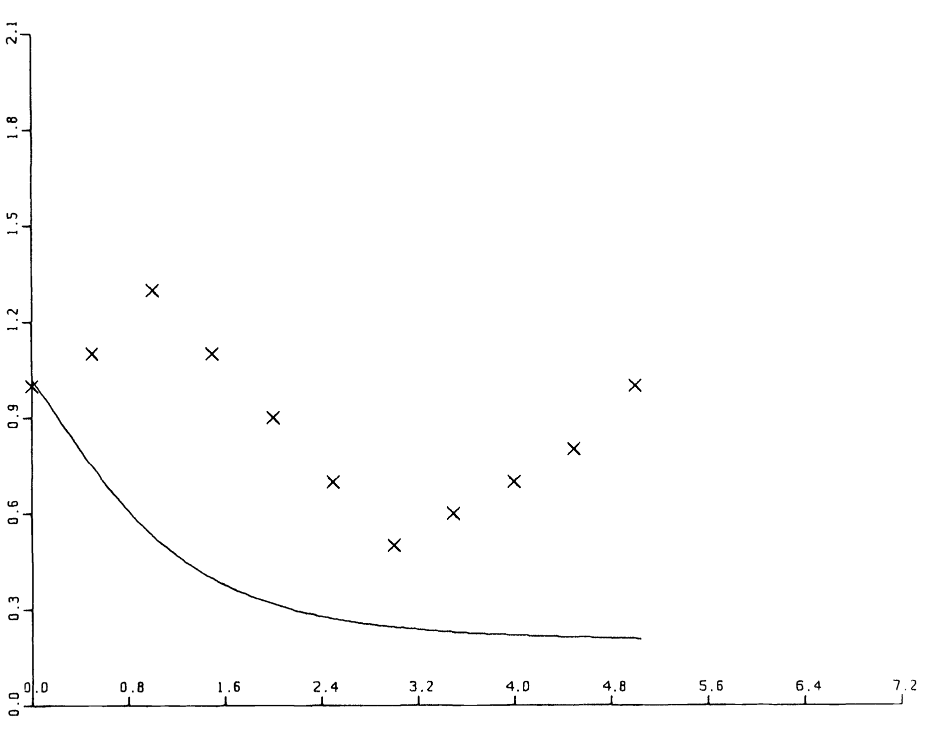


FIG. 6. *Example* A—*Integration using optimal four-knot spline fit.*

B. *Bellman's problem* (Bellman et al. (1967)).

$$y_1' = p_1(126.2 - y_1)(91.9 - y_1)^2 - p_2 y_1^2.$$

TABLE 3

t	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
y_1	0.0	1.4	6.3	10.4	14.2	17.6	21.4	23.0
t	10.0	12.0	15.0	20.0	25.0	30.0	40.0	
y_1	27.0	30.4	34.4	38.8	41.6	43.5	45.3	

This problem arises from a chemical reaction, and is also treated in van Domselaar and Hemker (1975). This is somewhat easier to solve than problem A, and we give results in Table 4.

TABLE 4.

Knot positions	Spline residuals	DE residual (# sample points)	Parameters found	Integrated residual	Best initial condition
20.2	2.7	0.86 (15)	$0.46\times10^{-5}, 0.27\times10^{-3}$	3.9	−1.10
20.2	2.7	0.76 (20)	$0.46\times10^{-5}, 0.30\times10^{-3}$	4.0	−0.98
20.2	2.7	0.98 (40)	$0.47\times10^{-5}, 0.31\times10^{-3}$	3.7	−1.49
5.0, 15.0	1.6	3.0 (20)	$0.40\times10^{-5}, 0.15\times10^{-3}$	6.7	0.89
5.0, 15.0	1.6	3.1 (40)	$0.41\times10^{-5}, 0.23\times10^{-3}$	6.4	0.33

Notice that we get a better result using only one knot, and this is at least indicated by the relative size of the DE residual in column 3. In Figs. 7–10, we give the best results for both one and two knots. Again the Hessian matrix was not ill-conditioned; however, in this case the data were fitted very well, so the variation in the parameters is much less than in Example A. The VDH values were $(0.45\times10^{-5}, 0.27\times10^{-3})$ with $R_I=4.7$.

C. *Enzyme effusion problem* (van Domselaar and Hemker (1975)).

$$y_1' = p_1(27.8-y_1)+\frac{p_4}{2.6}(y_2-y_1)+\frac{4991}{t\sqrt{2\pi}}\exp\left(-0.5\left(\frac{\log(t)-p_2}{p_3}\right)^2\right),$$

$$y_2' = \frac{p_4}{2.7}(y_1-y_2).$$

TABLE 5

t	0.1	2.5	3.8	7.0	10.9	15.0	18.2	21.3	22.9	24.9
y_1	27.8	20.0	23.5	63.6	267.5	427.8	339.7	331.9	243.5	212.0
t	26.8	30.1	34.1	37.8	42.4	44.4	47.9	53.1	59.0	65.1
y_1	164.1	112.7	88.1	76.2	62.3	58.7	41.9	40.2	31.3	30.0
t	73.1	81.1	91.2	101.9	115.4	138.7	163.2	186.7		
y_1	30.6	23.5	24.8	26.1	33.3	17.8	16.8	16.8		

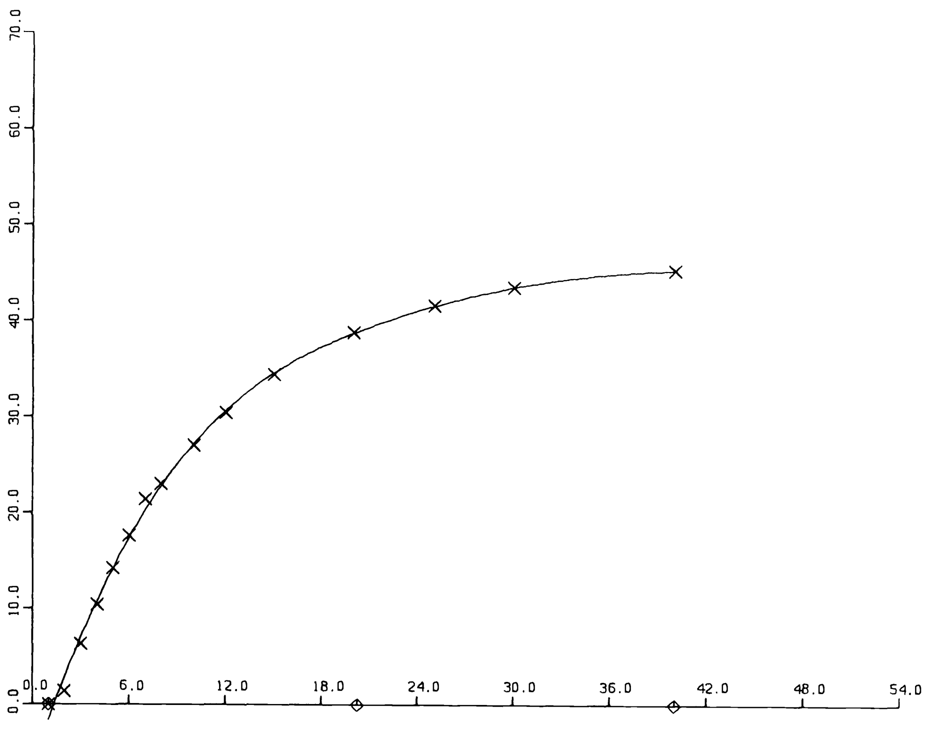


FIG. 7. *Example* B—*One-knot spline fit.*

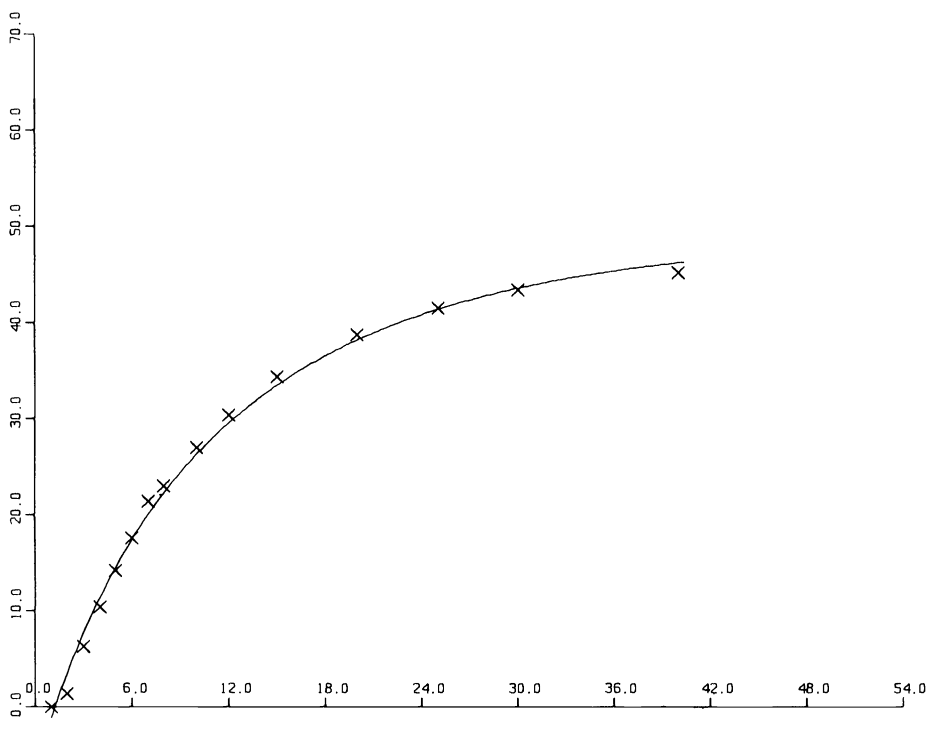


FIG. 8. *Example* B—*Integration using one-knot spline fit.*

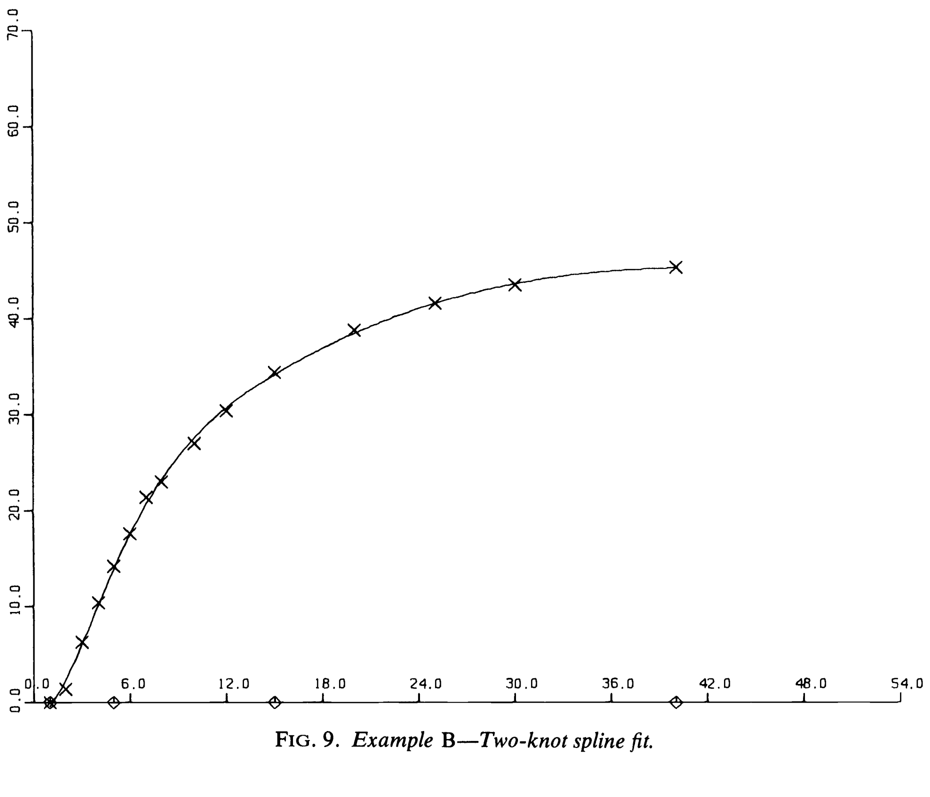


FIG. 9. *Example* B—*Two-knot spline fit.*

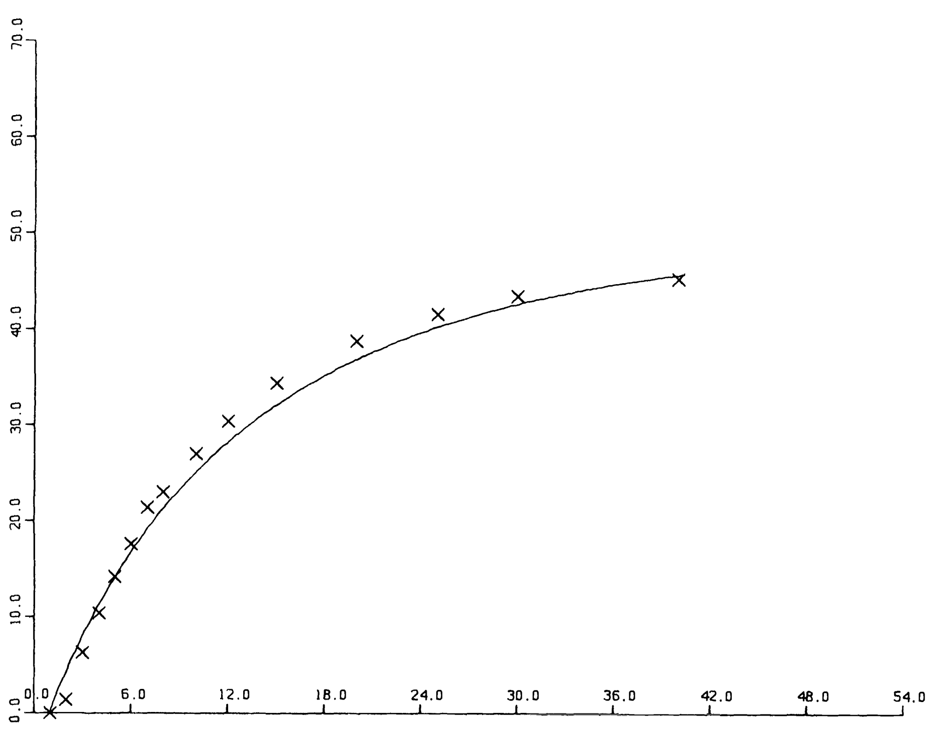


FIG. 10. *Example* B—*Integration using two-knot spline fit.*

This problem represents the modeling of enzyme concentrations in the blood, inside and outside the heart, over a period of time. A complication here is that observations are only available on y_1. We remedy this (as indicated earlier) by solving the first equation for y_2, differentiating and substituting in the second equation to get a single second order equation for y_1.

This was more difficult to solve than the first two, both because of the nonlinear parameters and because of the difficulty in obtaining a good spline fit to the data. Results were as shown in Table 6.

TABLE 6

Knot positions	Spline residual	DE residual (# sample points)	Parameters found	Integrated residual
8.0, 11.0, 23.0, 43.0	64	7.8 (28)	0.326, 2.674, 0.40, 0.198	94
8.0, 11.0, 23.0, 43.0	64	15.5 (40)	0.257, 2.62, 0.364, 0.29	70
9.2, 11.25, 22.7, 42.8	62	5.3 (28)	0.36, 2.72, 0.40, 0.04	111
9.2, 11.25, 22.7, 42.8	62	15.7 (40)	0.266, 2.65, 0.353, 0.228	64
8.0, 12.0, 18.0, 24.0, 43.0	60	6.6 (28)	0.278, 2.673, 0.378, 0.193	64
8.0, 12.0, 18.0, 24.0, 43.0	60	12.3 (40)	0.251, 2.61, 0.348, 0.327	67
11.3, 12.1, 15.0, 29.1, 39.4	54	7.9 (28)	0.017, 2.63, 0.281, 0.277	1968
11.3, 12.1, 15.0, 29.1, 39.4	54	58.0 (40)	0.192, 2.52, 0.231, 0.487	136

For each knot set, we used 28 sample points (=data points) and 40 sample points (skewed to represent the function better). The first knot set represents the best we could do with four knots chosen visually. We then tried optimizing the knot locations by minimizing the least squares deviation at the data points as a function of the four knots. This gave the second knot set which, with 40 sample points, gave the best result. We plot this spline fit and result in Figs. 11 and 12. The third set of five knots was again chosen visually and gave equally good results (notice, however, that the last parameter has changed appreciably without affecting the residual). We again optimized the knots, but this final knot set was not as successful (although still reasonable). We plot this for comparison in Figs. 13 and 14. The VDH values were (0.27, 2.65, 0.364, 0.21) with $R_I = 64$.

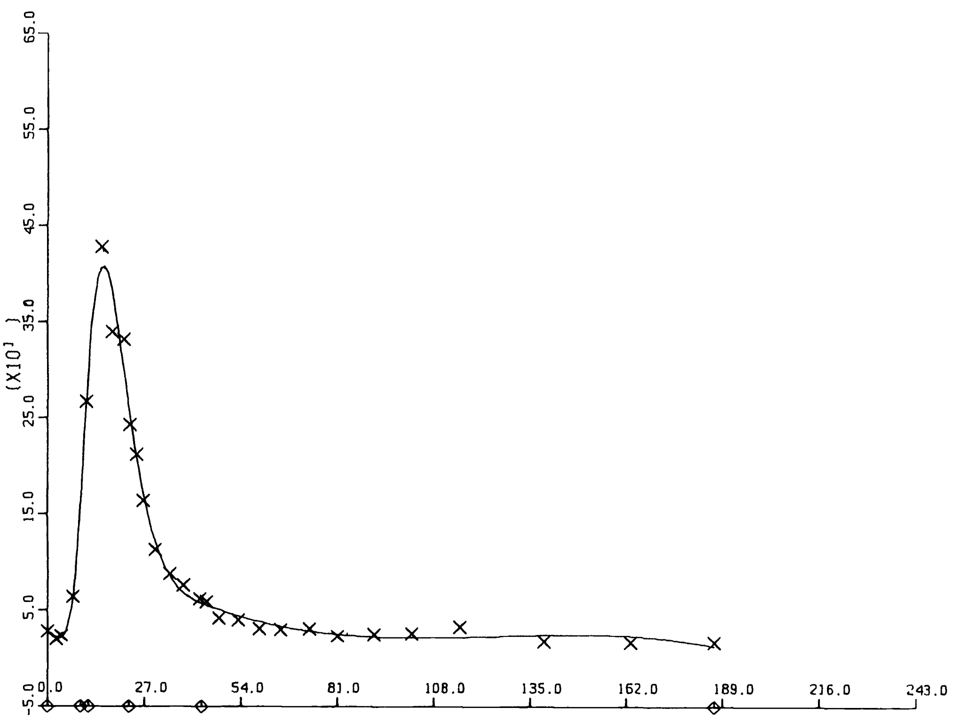


FIG. 11. *Example C—Optimal four-knot spline fit.*

D. *Blood ethanol problem* (Ralston et al. (1979)).

Finally we present an example with very sensitive parameters. This is (Ralston et al. (1979, Example 3.3)) where the blood ethanol concentration was measured over a period of time, with intravenous ethanol injected for an initial period. Michaelis–Menton kinetics are assumed, giving the model

$$y' = \begin{cases} p_1 \dfrac{p_2 y}{p_3 + y}, & t \leqq 2, \\ -\dfrac{p_2 y}{p_3 + y}, & t > 2. \end{cases}$$

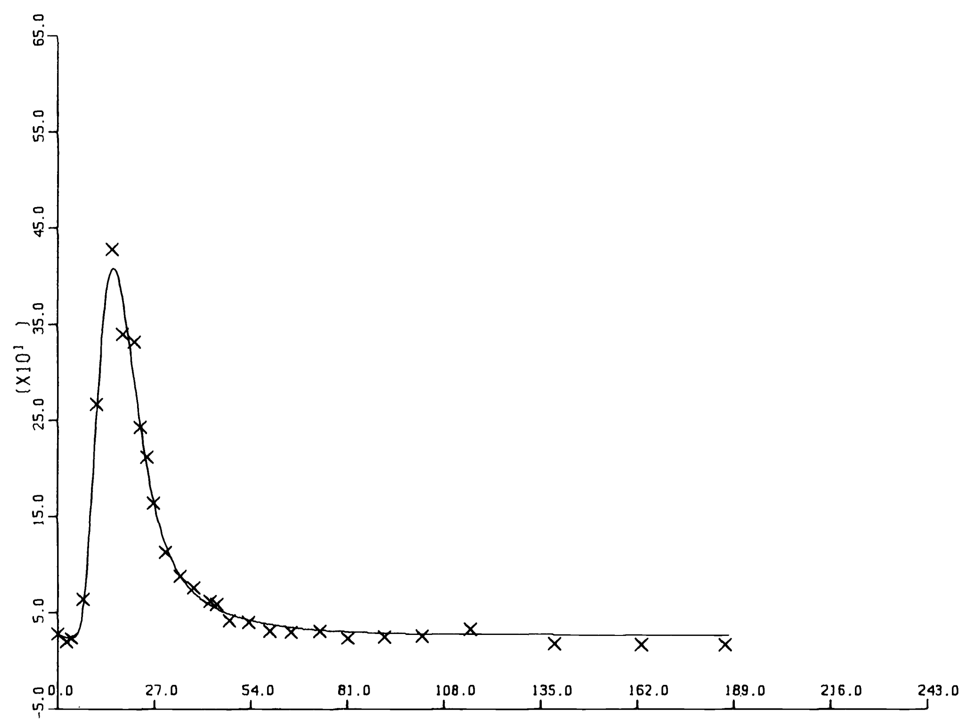


FIG. 12. *Example C—Integration using optimal four-knot spline fit.*

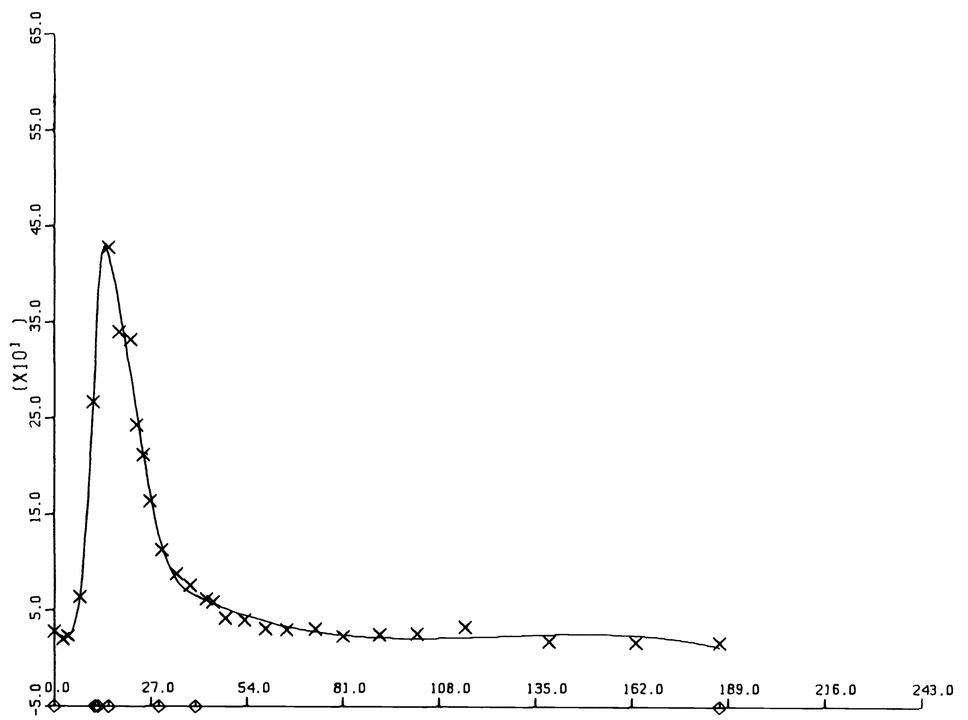


FIG. 13. *Example C—Optimal five-knot spline fit.*

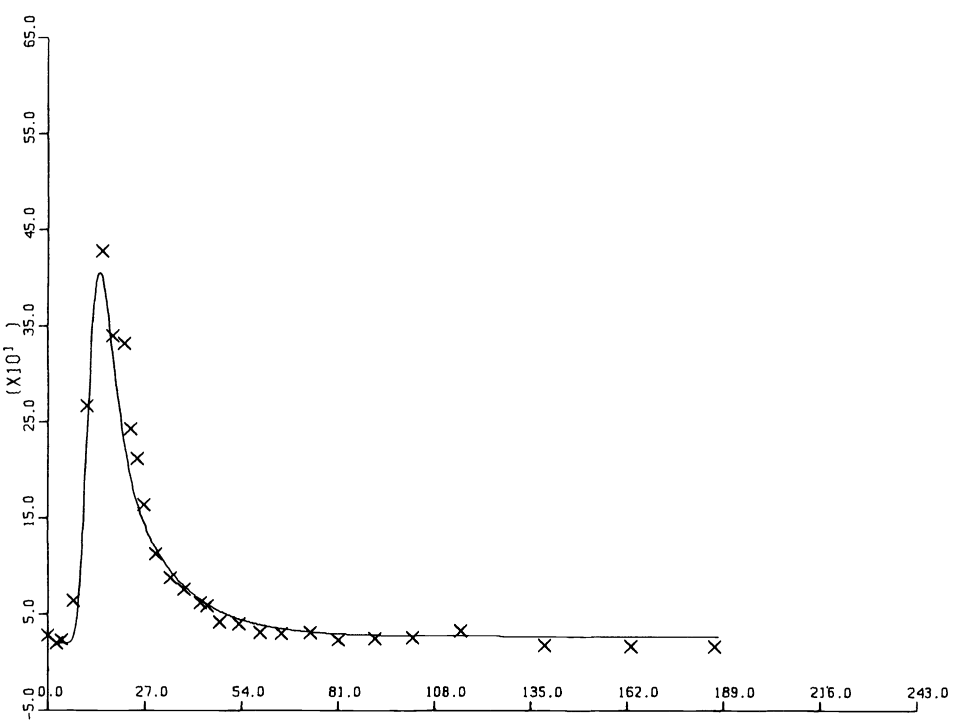


FIG. 14. *Example C—Integration using optimal five-knot spline fit.*

The data are shown in Table 7.

TABLE 7

t	0.0	0.083	0.250	0.500	0.750	1.0	1.5	2.0	2.083	2.167		
y	0.0	0.10	0.23	0.37	0.47	0.52	0.69	0.81	0.79	0.72		
t	2.25	2.50	2.75	3.0	3.5	4.0	4.5	5.0	5.25	5.5		
y	0.78	0.66	0.59	0.59	0.55	0.48	0.41	0.32	0.29	0.29		
t	5.75	6.0	6.26	6.50	6.75	7.0	7.25	7.5	7.75	8.0	8.25	8.5
y	0.23	0.17	0.13	0.10	0.06	0.049	0.037	0.024	0.017	0.011	0.0053	0.0023

The authors minimized a weighted least squares residual by an initial value technique. We prefer to consider the nonweighted case, and can compare the results from our technique with those obtained from direct integration, since we can integrate the DE explicitly:

$$\text{for } t \leqq 2, \quad \text{solve } p_1(e^x - 1) - p_2 x = \frac{(p_1 - p_2)^2}{p_3} t \quad \text{for } x$$

$$\text{and set } y(t) = \frac{p_1 p_3 (e^x - 1)}{p_1 - p_2};$$

$$\text{for } t > 2, \quad \text{solve } p_3 x + y(2)(e^x - 1) = (2-t)p_2 \quad \text{for } x$$
$$\text{and set } y(t) = y(2)e^x.$$

Both of these equations for x are nonlinear, of the form $e^x = a + bx$, and can be solved by Newton's method (although for $t < 2$ a good starting approximation is needed as there is an extraneous root). Thus for any t, we can compute $y(t)$, and thus the least squares function

$$f^2(\mathbf{p}) = \sum_1^n (y_i - y(t_i; \mathbf{p}))^2.$$

We minimized this directly as a function of $\mathbf{p}$, obtaining a minimum of $f = 0.30$ at $\mathbf{p}^* = (0.557, 0.221, 0.151)$.

For our spline technique, it is clear from the model that the solution is only C^0 at $t = 2$, so it is natural to use a spline fit with a triple knot at this point. However, although the spline did fit the data well even with no additional knots (see Fig. 15), the parameters obtained by minimizing the DE residual R_D were not close to the values above, yet gave much the same residual. Typical values were (with forty sample points) $\mathbf{p} = (0.58, 0.78, 1.54)$ with $R_I = 0.32$. We plot this solution in Fig. 16 and give the direct integration result in Fig. 17. Notice how flat the least squares surface must be: p_2 and p_3 have changed enormously, with little change in R_I.

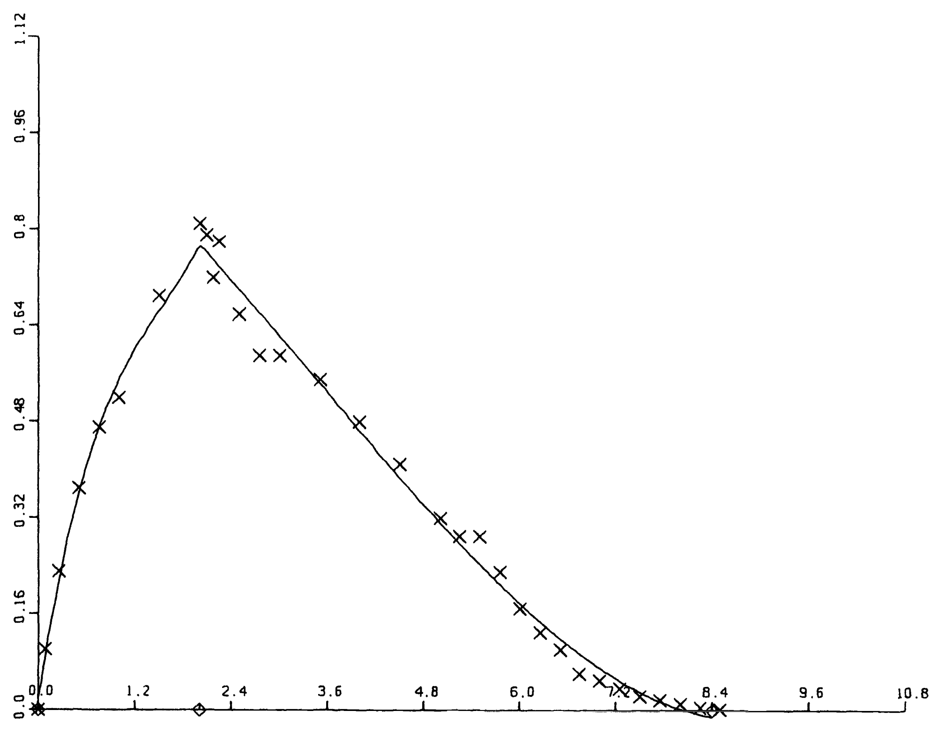


FIG. 15. *Example D—Triple knot spline fit.*

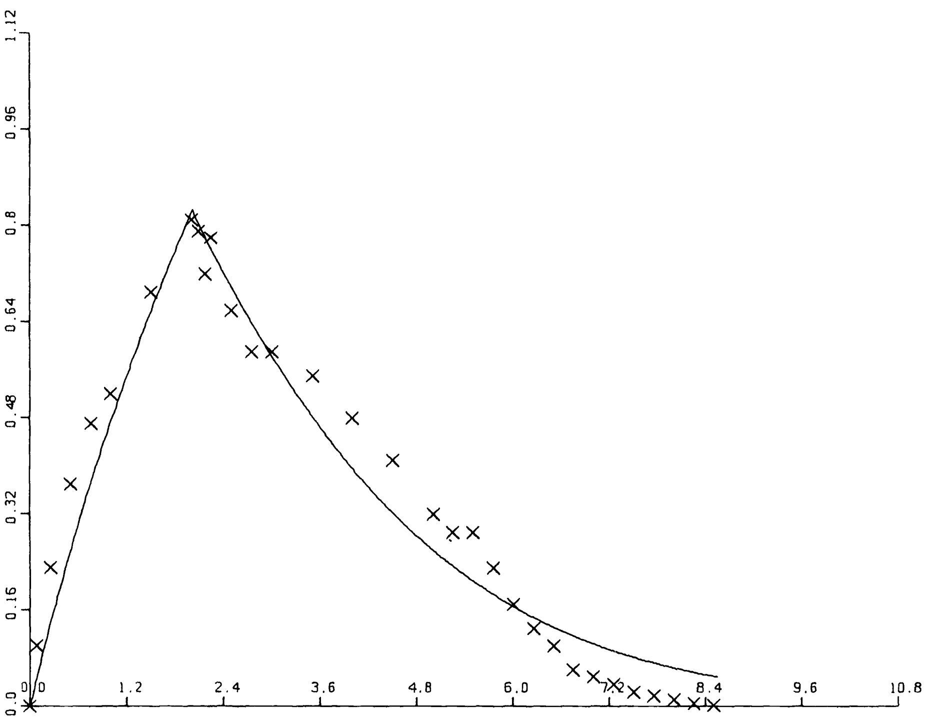


FIG. 16. *Example* D—*Integration using triple knot spline fit.*

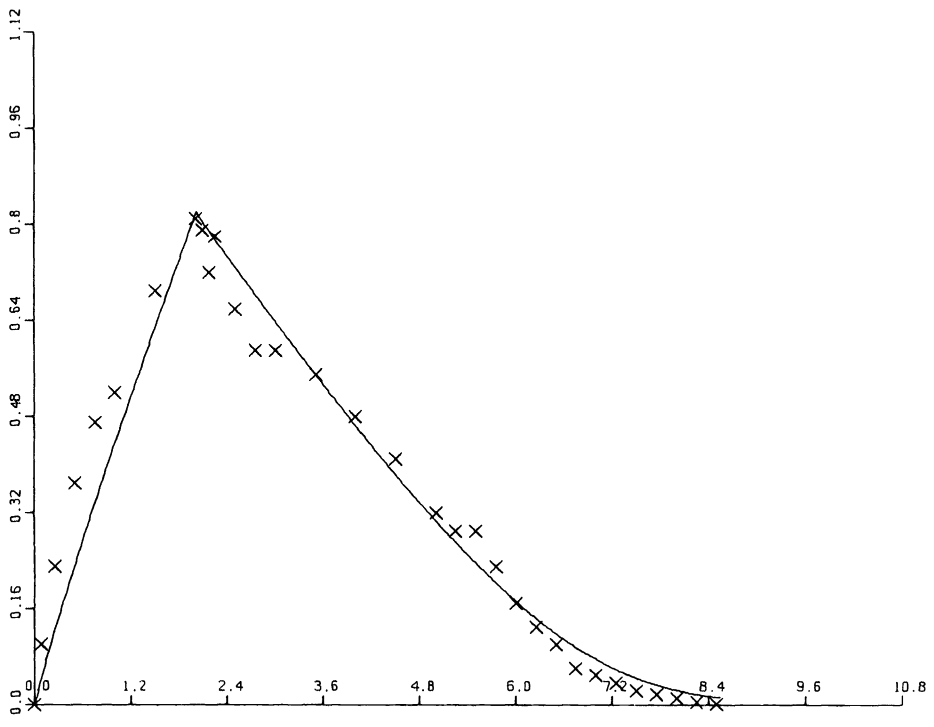


FIG. 17. *Example* D—*Integration using direct minimization.*

REFERENCES

R. S. ANDERSSEN AND P. BLOOMFIELD (1974), *Numerical differentiation procedures for non-exact data*, Numer. Math., 22, pp. 157–182.

Y. BARD (1974), *Nonlinear Parameter Estimation*, Academic Press, New York.

R. BELLMAN *et al.* (1967), *Quasilinearization and the estimation of chemical rate constants from raw kinetic data*, Math. Biosci., 1, pp. 71–76.

M. BENSON (1979), *Parameter fitting in dynamic models*, Ecol. Mod., 6, pp. 97–115.

C. CLARK (1976), *Mathematical Bioeconomics*, Wiley-Interscience, New York.

C. DE BOOR (1979), *A Practical Guide to Splines*, Springer-Verlag, New York.

J. E. DENNIS, D. M. GAY AND R. E. WELSCH (1979), *An adaptive nonlinear least squares algorithm*, MRC Tech. Rep. 2010, Mathematics Research Center, Univ. of Wisconsin, Madison.

G. H. GOLUB AND V. PEREYRA (1973), *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*, SIAM J. Numer. Anal., 10, pp. 413–432.

R. MEZAKI, N. R. DRAPER AND R. A. JOHNSON (1973), *On the violation of assumptions in nonlinear least squares by interchange of response and predictor variables*, I & EC Fundamentals, 12, p. 251.

L. NAZARETH (1980), *Some recent approaches to solving large residual nonlinear least squares problems*, SIAM Rev., 22, pp. 1–11.

M. L. RALSTON, R. I. JENNRICH, P. F. SAMPSON AND F. K. UNO (1979), *Fitting models defined by differential equations*, Proc. 12th Annual Symposium on the Interface of Computer Science and Statistics, Univ. of Waterloo, Ontario, J. Gentleman, ed.

A. RUHE AND P. A. WEDIN (1980), *Algorithms for separable nonlinear least squares problems*, SIAM Rev., 22, pp. 318–337.

J. SWARTZ AND H. BREMERMAN (1975), *Discussion of parameter estimation in biological modelling: Algorithms for estimation and evaluation of the estimates*, J. Math. Biol., 1, pp. 241–257.

B. VAN DOMSELAAR AND P. W. HEMKER (1975), *Nonlinear parameter estimation in initial value problems*, Rep. NW18/75, Mathematical Centrum, Amsterdam.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 1, March 1982

EXTREMAL POLYNOMIALS WITH APPLICATION TO RICHARDSON ITERATION FOR INDEFINITE LINEAR SYSTEMS*

CARL DE BOOR† AND JOHN R. RICE‡

Abstract. The application of Richardson iteration to a symmetric, but indefinite linear system requires certain parameters which can be determined from the zeros in the error of a certain best polynomial approximant on some set S known to contain the spectrum of the coefficient matrix. It is pointed out that this error can also be obtained as a multiple of the extremal polynomial for the linear functional $p \mapsto p(0)$, and this leads to an efficient Remes type algorithm for its determination. A program incorporating this algorithm for the case that S consists of two or more intervals bracketing zero is available.

Key words. Richardson iteration, symmetric indefinite, Remes, norm preserving extension, norm calculation for linear functional, Chebyshev polynomial

1. The iteration problem.

Consider the linear system of equations $A\mathbf{x}=\mathbf{b}$, and the iteration

$$\mathbf{x}^{n+1}=\mathbf{x}^n-\alpha_n(A\mathbf{x}^n-\mathbf{b}).$$

With $\mathbf{e}^n := \mathbf{x}^n-\mathbf{x}$ the error in the nth iterate, we have

$$\mathbf{e}^n=(1-\alpha_{n-1}A)\mathbf{e}^{n-1}=\cdots=\prod_{j=1}^{n}(1-\alpha_{j-1}A)\mathbf{e}^0=Q_n(A)\mathbf{e}^0,$$

where Q_n is the polynomial of degree n which vanishes at $1/\alpha_0,\cdots,1/\alpha_{n-1}$ and is 1 at 0. This is *Richardson's* (first order) *iteration*, with iteration parameters α_j. If the spectrum of A is known to lie in some compact set S, then a standard analysis suggests that one should choose the parameters α_j so as to minimize

$$\|Q_n\|_S := \max_{s\in S}|Q_n(s)|.$$

The resulting polynomial P_n is then the error in the *best Chebyshev approximation on S to* 1 *from* $\{\sum_{j=1}^n \beta_j t^j\}$. If S is an interval not containing the origin (hence A is known to be definite), then it is well known that a renormalization of P_n to make the coefficient of t^n equal to 1 gives T_n, the Chebyshev polynomial for the interval S. For this case, the three-term recurrence relation for the Chebyshev polynomials may be employed to build up x^n without the use of the zeros of P_n. This has the advantage that the iterates x^i so generated along the way are themselves using P_i. This method is known as the *Chebyshev semi-iterative method.* This variation requires more memory (3 vectors rather than 2 are used) and more computation per step (since more vectors are combined per step). The *conjugative gradient method* is a further variation which, with some more work per iteration, removes the dependence on the interval S; the mere knowledge that such an interval exists suffices to show that the error produced at the nth step is of the form $P_n\mathbf{e}^0$, with P_n the error in a best approximation to 1 on the spectrum of A itself.

The conjugate gradient method may run into difficulties when A, though symmetric and invertible, is not definite. See Paige and Saunders [1975] for a detailed

* Received by the editors August 8, 1980, and in revised form June 11, 1981. This work was sponsored by the U.S. Army under contract DAAG29-80-C-0041.

† Mathematics Research Center, University of Wisconsin, Madison, Wisconsin 53706.

‡ Division of Mathematical Sciences, Purdue University, West Lafayette, Indiana 47907. The work of this author was supported in part by the National Science Foundation under Grant MCS 77-01408.

discussion and some remedies. For this reason, Richardson iteration becomes an attractive alternative in this case. We now have the spectrum of A contained in two intervals, with the origin between them. Akhiezer [1928] determined the Chebyshev polynomials for two such intervals of *equal length*, and Lebedev [1969] extended this technique to a set S consisting of an arbitrary number of intervals of equal length and applied his result to iteration. See Anderssen and Golub [1972] for a translation of Lebedev's paper and further discussions, particularly on the important subject of the order in which best to use the α_i's.

Specifically, let $S=[a,b]\cup[c,d]$. For certain values of a, b, c and d, Atlestam [1977] has obtained a representation of the Chebyshev polynomials for S, of the following form: Let

$$Q(t) := \cos(m(\pi+I(t))),$$

with

$$I(t) := \int_a^t (u-r)p(u)\,du, \qquad r := \int_b^c up(u)\,du\Big/\int_b^c p(u)\,du$$

and

$$p(u) := ((u-a)(u-b)(u-c)(d-u))^{-1/2}.$$

If there are integers m and k so that $I(b)=k\pi/m$, then Q is a polynomial of degree m proportional to the Chebyshev polynomial for S. Atlestam further shows that, for any interval pair S, the Chebyshev polynomial is of this form but for a slightly different pair of intervals, and this difference goes to zero as the degree goes to infinity. Her arguments can be used to show that in the same way, for any interval pair S bracketing the origin, the best polynomial P_n is of the above form, but for a slightly different interval pair. These results can be used to obtain sharp asymptotic results on the degree of convergence of the iteration method, but it is not clear how useful the representation is for obtaining the necessary iteration parameters.

In the present paper, we give what we feel is a more useful formulation of the mathematical problem underlying the determination of the parameters; well-known results then establish existence and uniqueness of the solution of this problem and characterize it. In particular, we are led to a Remes type algorithm for the determination of P_n, whose zeros can then be determined efficiently by the Modified Regula Falsi. For the particular case that S is an interval pair, we present some numerical results to illustrate the nature of the parameters and the convergence rate of the corresponding iteration.

2. The extremal polynomial. The papers mentioned above all use Chebyshev polynomials in some essential way, so we first note that, in general, the required polynomial P_n is unrelated to the Chebyshev polynomial T_n for S. This is seen in the analysis of Atlestam [1977] or, more directly, from the fact shown below that P_n alternates one less time on S than does T_n.

To recall, the Chebyshev polynomial T_n for the compact set S is the polynomial of the form $t^n+\sum_{j=0}^{n-1}\beta_j t^j$ which is as small as possible on S. In other words, T_n is the error in the best approximation on S to t^n from $\{\sum_0^{n-1}\beta_j t^j\}$. By contrast, we are interested in the polynomial P_n which is the error in the best approximation on S to 1 from $\{\sum_1^n \beta_j t^j\}$.

We now reformulate this problem as follows. Let λ be the linear functional on π_n (:= the polynomials of degree n or less) whose value at p is $p(0)$. In symbols,

$$\lambda : \pi_n \to \mathbb{R} : p \mapsto p(0).$$

An *extremal for* λ is any polynomial of norm 1 at which λ takes on its norm, i.e., any $p \in \pi_n$ with $\|p\|_S = 1$ and $\lambda p = \|\lambda\|$. Here

$$\|p\| = \|p\|_S := \max_{s \in S} |p(s)|$$

and

$$\|\lambda\| := \max_{p \in \pi_n} \frac{\lambda p}{\|p\|_S} = \frac{1}{\min\{\|p\|_S : p \in \pi_n, p(0) = 1\}}.$$

This shows that the polynomial P_n which is of minimum norm on S and satisfies $P_n(0) = 1$ is a multiple of an extremal p^* for λ, i.e., $P_n = p^*/p^*(0)$.

The standard approach to the construction of extremals is via norm preserving extensions, i.e., via a so-called canonical representation for λ (see, e.g., Rivlin [1974, pp. 82 ff.]). Such a *canonical representation for* λ consists of $n+1$ points $t_1 < t_2 < \cdots < t_{n+1}$ in S and corresponding coefficients $\alpha_1, \alpha_2, \cdots, \alpha_{n+1}$, so that

$$\lambda p = \sum_{j=1}^{n+1} \alpha_j p(t_j), \quad \text{all } p \in \pi_n$$

and

$$\|\lambda\| = \sum_{j=1}^{n+1} |\alpha_j|.$$

In other words, writing $[t]$ for the linear functional of evaluation at t, such a canonical representation provides us with an extension

$$\sum_{j=1}^{n+1} \alpha_j [t_j]$$

of λ from π_n to all of $C(S)$:= Banach space of continuous functions on S, and this extension has the same norm (on $C(S)$) as does λ (on π_n).

We will give a constructive proof later on of the existence of such a canonical representation for our particular λ. Taking this for granted (or referring for it to Rivlin [1974]), we note that, for the Lagrange polynomials l_j given by

$$l_j(t) := \prod_{\substack{i=1 \\ i \neq j}}^{n+1} \frac{t - t_i}{t_j - t_i}, \qquad j = 1, \cdots, n+1$$

we must then have

$$l_j(0) = \lambda l_j = \sum_{i=1}^{n+1} \alpha_i l_j(t_i) = \alpha_j.$$

This implies that

$$\alpha_j = \prod_{\substack{i=1 \\ i \neq j}}^{n+1} \frac{-t_i}{t_j - t_i}, \quad \text{all } j;$$

hence all coefficients are nonzero if, as we assume, 0 does not lie in S. Further,

$$\alpha_j\alpha_{j+1} \geqq 0 \quad \text{iff} \quad t_j \leqq 0 \leqq t_{j+1}.$$

If now p^* is an extremal for λ, then we have

$$\|\lambda\| = \lambda p^* = \sum_{j=1}^{n+1} \alpha_j p^*(t_j) \leqq \sum_{j=1}^{n+1} |\alpha_j|\,|p^*(t_j)| \leqq \left(\sum_{j=1}^{n+1} |\alpha_j|\right)\|p^*\| = \|\lambda\|,$$

so equality must hold throughout this relationship. In particular,

$$\operatorname{sign}(\alpha_j)p^*(t_j) = \|p^*\| = 1, \quad \text{all } j.$$

This pins down p^* uniquely. Explicitly,

$$p^* = \sum_{j=1}^{n+1} \operatorname{sign}(l_j(0))l_j$$

for any canonical representation $\sum \alpha_j[t_j]$ of λ. Moreover, if p is a polynomial of norm 1 of the form $\sum_{j=1}^{n+1} \operatorname{sign}(l_j(0))l_j$ for some points $t_1, \cdots, t_{n+1}$ in S, then $p = p^*$.

If now 0 lies to one side of $[t_1, t_{n+1}]$, then it follows that p^* *alternates* on $t_1, \cdots, t_{n+1}$; hence p^* is necessarily a multiple of the Chebyshev polynomial for $S \cap [t_1, t_{n+1}]$. Further, $|p^*(t)| > 1$ for t not in $[t_1, t_{n+1}]$. We conclude that in the case of particular interest to us, namely when 0 is in the convex hull of S, there must be some k for which

$$t_k < 0 < t_{k+1}.$$

This shows our assertion at the beginning of this section that, in general, P_n need only alternate on n points in S. Further, for $t_k < t < t_{k+1}$,

$$p^*(t) = \sum_{j=1}^{n+1} (\operatorname{sign}(l_j(0)))l_j(t) = \sum (\operatorname{sign}(l_j(t)))l_j(t) = \sum |l_j(t)| > |\sum l_j(t)| = 1$$

if $n > 1$. Consequently, then,

$$t_k = b := \max S \cap (-\infty, 0], \qquad t_{k+1} = c := \min S \cap [0, \infty).$$

We gather these various facts in the following theorem, for the record.

THEOREM 1. *Assume that S is compact and does not contain* 0, *and $n > 1$. Further, let $\sum_{j=1}^{n+1} \alpha_j[t_j]$ be a canonical representation for $\lambda : p \mapsto p(0)$. Then*

(a) $$\alpha_j = \prod_{i \neq j} (-t_i/(t_j - t_i)), j = 1, \cdots, n+1.$$

(b) *λ has a unique extremal, p^*, and this extremal satisfies*

$$p^* = \sum_{j=1}^{n+1} \operatorname{sign}(l_j(0))l_j \quad \textit{with } l_j(t) = \prod_{i \neq j} \frac{t - t_i}{t_j - t_i}, \quad \textit{all } j.$$

If, in addition, 0 *is in the convex hull of S, then*

(c) *$t_k < 0 < t_{k+1}$ for some k. Further,*

$$p^*(t_j) = \begin{cases} (-1)^{k-j}, & j = 1, \cdots, k, \\ (-1)^{j+1-k}, & j = k+1, \cdots, n+1, \end{cases}$$

and $p^*(t) > 1$ for $t_k < t < t_{k+1}$; hence $t_k = \max S \cap [-\infty, 0]$ and $t_{k+1} = \min S \cap [0, \infty)$.

We pointed out earlier that $P_n = p^*/p^*(0)$ could also be obtained as the error in the Chebyshev approximation to 1 from the subspace $\{\sum_1^n \beta_j t^j\}$. This subspace forms a Haar set on S as long as S does not contain 0. We could therefore have obtained the above characterization from general criteria such as Kolmogorov's criterion, but the derivation would not have been any simpler. We note that, while P_n is in general not (a multiple of) the Chebyshev polynomial for S, it is always a multiple of a Zolotarev polynomial since its alternations over n points characterize it as the error in a best approximation to $\beta_n t^n + \beta_{n-1} t^{n-1}$ from π_{n-2}.

3. Remes algorithm for the extremal polynomial. We begin with a statement of the algorithm. In it, we use the abbreviations

$$b := \max S \cap (-\infty, 0], \qquad c := \min S \cap [0, \infty)$$

introduced earlier.

Remes algorithm for the extremal polynomial.

```
1. Choose $\mathbf{t} = \{t_j\}_{j=1}^{n+1}$ in $S$, strictly increasing, and with $t_k = b$, $t_{k+1} = c$ for some $k$.
2. Set $p := \sum_{j=1}^{n+1} \operatorname{sign}(l_j(0)) l_j$, with $l_j(t) := \prod_{i \neq j} (t - t_i)/(t_j - t_i)$, all $j$.
3. Set $t_0 := \min S$, $t_{n+2} := \max S$ and construct $\mathbf{s}$ by
   $s_j := \begin{cases} t_j & \text{for } j = k, k+1, \\ \text{the first of the possibly two maxima of } p(t_j)p \text{ in } [t_{j-1}, t_{j+1}] \cap S \\ \quad \text{for } j = 1, \cdots, k-1, k+2, \cdots, n+1. \end{cases}$
4. Choose $\tilde{\mathbf{t}}$ from $\mathbf{s}$ as follows:
   (a) if $p(t_0)p(t_1) < -1$, then $\tilde{\mathbf{t}} := (t_0, s_1, \cdots, s_n)$, and increase $k$ by 1;
   (b) if $p(t_{n+2})p(t_{n+1}) < -1$, then $\tilde{\mathbf{t}} := (s_2, \cdots, s_{n+1}, t_{n+2})$, and decrease $k$ by 1;
   (c) otherwise, $\tilde{\mathbf{t}} := \mathbf{s}$.
5. Set $\mathbf{t} := \tilde{\mathbf{t}}$.
6. Iterate steps 2 through 5.
```

THEOREM 2. *The sequence of polynomials produced by the above Remes algorithm converges to p^*.*

Proof. We first note that the algorithm is well defined at step 4 in that only one of the alternatives (a) and (b) is possible. Indeed, if both (a) and (b) were to occur, then p would alternate on $t_0, \cdots, t_k, t_{k+2}, \cdots, t_{n+2}$, and this is not possible for a polynomial of degree n or less.

As to the convergence, denote by $\tilde{p}$ the polynomial obtained from p after one iteration, i.e., the polynomial constructed from the sequence $\tilde{\mathbf{t}}$ obtained at step 4. We claim that $1 < \tilde{p}(t) \leqq p(t)$ for any t in (b, c) and that strict inequality holds here unless $\tilde{\mathbf{t}} = \mathbf{t}$. The first inequality we already observed earlier (for p^*). As to the second, we have

$$(-1)^{j-k}\tilde{p}(\tilde{t}_j) = 1 \leqq (-1)^{j-k} p(\tilde{t}_j) \qquad \text{for } j = 1, 2, \cdots, k,$$

$$(-1)^{j-k-1}\tilde{p}(\tilde{t}_j) = 1 \leqq (-1)^{j-k-1} p(\tilde{t}_j) \quad \text{for } j = k+1, \cdots, n+1$$

by construction. This implies that, for $b < t < c$,

$$p(t) - \tilde{p}(t) = \sum_{j=1}^{n+1} (p(\tilde{t}_j) - \tilde{p}(\tilde{t}_j))\tilde{l}_j(t) = \sum |p(\tilde{t}_j) - \tilde{p}(\tilde{t}_j)|\, |\tilde{l}_j(t)| \geqq 0,$$

and equality occurs only if $p = \tilde{p}$ on the $n+1$ points $\tilde{t}_1, \cdots, \tilde{t}_{n+1}$, i.e., only if $p = \tilde{p}$.

We conclude that the sequence generated by the Remes algorithm decreases monotonically on (b, c), yet is bounded below there. Hence it converges, uniformly on any finite interval, to some polynomial p^∞. Since the map T in π_n given by

$$T : p \mapsto \tilde{p}$$

is not everywhere continuous (a fact pointed out to us by Joe Grcar and Paul Saylor), we cannot use the standard argument which would now finish with the observation that p^∞ must be a fixed point of T, hence equal to p^*. Instead, we verify directly that $p^\infty = p^*$, as follows.

We have to prove that, for some $t_1^\infty < \cdots < t_{n+1}^\infty$, all in S,

$$p^\infty = \sum_{j=1}^{n+1} (l_j^\infty(0)) l_j^\infty, \quad \text{with } l_j^\infty(t) := \prod_{i \neq j} \frac{t - t_i^\infty}{t_j^\infty - t_i^\infty}, \quad \text{all } j$$

and that

$$\|p^\infty\| \leqq 1.$$

The first is easily seen to hold since p^∞ is the uniform limit of polynomials of the same form (by step 2 of the algorithm). In particular, $t_1^\infty < \cdots < t_{n+1}^\infty$ could be chosen as limit points of the sequences $t_1 < \cdots < t_{n+1}$ used in steps 2–4 of the algorithm.

As to the second, we prove that the assumption $\|p^\infty\| > 1$ leads to a contradiction, as follows. To begin with, we claim that $\tilde{t}_1 = t_0$ or $\tilde{t}_{n+1} = t_{n+2}$. This is obvious in case of alternative (a) or (b) in step 4. But if, under alternative (c), $\tilde{t}_1 > t_0$ and $\tilde{t}_{n+1} < t_{n+2}$, then $\Delta p(u_j)\, \Delta p(u_{j+1}) \leqq 0$, $j = 1, \cdots, n$, for the strictly increasing sequence

$$(u_j) := (t_0, s_1, \cdots, s_k, s_{k+2}, \cdots, s_{n+1}, t_{n+2}),$$

which would imply that $p' = 0$, hence $n = 1$, a contradiction.

Thus we are free to assume that either $t_1 = t_0$, or else $t_{n+1} = t_{n+2}$. Further, since p converges to p^∞, the difference $p - \tilde{p}$ must go to zero, hence $|p(\tilde{t}_j)| \to |\tilde{p}(\tilde{t}_j)| = 1$. The assumption $\|p^\infty\| > 1$ then implies that, for some $s \in S$ and some $e > 1$, eventually every p satisfies the inequalities

$$\max_j |p(\tilde{t}_j)| < e \leqq |p(s)|.$$

We now consider various cases.

Case $p(s)p(t_j) > 0$ for some $j \in \{1, \cdots, n+1\} \backslash \{k, k+1\}$ and with $t_{j-1} \leqq s \leqq t_j$ or $t_j \leqq s \leqq t_{j+1}$. Then $|p(s_j)| \geqq |p(s)| > |p(\tilde{t}_i)|$ for all i; hence either $j = n+1$ with alternative (a) or $j = 1$ with alternative (b) must hold. Assume without loss of generality the former. Then $t_0 < t_1$; therefore $t_{n+1} = t_{n+2}$ and now $\Delta p(u_j) \Delta p(u_{j+1}) \leqq 0$, $j = 1, \cdots, n$, for the strictly increasing sequence

$$(u_j) := (t_0, \cdots, t_k, s_{k+2}, \cdots, s_{n+1} = s, t_{n+1}),$$

a contradiction.

Case $p(s) > 0$ and $t_{k-1} \leqq s \leqq t_k$. Then $\Delta p(u_j) \Delta p(u_{j+1}) \leqq 0$ for $j = 1, \cdots, n$, for the strictly increasing sequence

$$(u_j) := (t_1, \cdots, t_{k-1}, s, t_k, \cdots, t_{n+1}),$$

a contradiction.

Case $p(s) > 0$ and $t_k \leqq s \leqq t_{k+1}$ is treated analogously.

Case $p(s)p(t_1) < 0$ and $t_0 \leqq s < t_1$. If $p(t_0)p(t_1) \leqq p(s)p(t_1)$, then

$$p(t_0)p(t_1) \leqq p(s)p(t_1) < -e < -1;$$

hence alternative (a) is chosen, which implies that $|\tilde{p}(t_0)| = 1$ while $|p(t_0)| \geqq e > 1$. Since p converges, this cannot happen eventually. Therefore, eventually $p(t_0)p(t_1) > p(s)p(t_1)$. But then $\Delta p(u_j) \Delta p(u_{j+1}) \leqq 0$, $j = 1, \cdots, n$, for the strictly increasing sequence

$$(u_j) := (t_0, s, t_1, \cdots, t_k, t_{k+2}, \cdots, t_{n+1}),$$

a contradiction. Finally,

Case $p(s)p(t_{n+1}) < 0$ and $t_{n+1} < s \leqq t_{n+2}$ is treated analogously.

4. Efficient computation of the parameters. We were led to study this problem by the work of Roloff [1979], where estimates of the parameters are provided. A result of Roloff's states that, for sufficiently large n, the zeros of P_n are approximately distributed in S in a proportion which is independent of n. One might hope that this proportion is determined by measure, e.g., a subinterval of S containing $\frac{1}{10}$ the length of S contains about $\frac{1}{10}$ of the zeros of P_n. We use this to obtain the initial guess (in step 1) for the Remes algorithm, but we also note that this approximate distribution of zeros of P_n is not especially good. Rather, there is also a tendency for the zeros to be distributed equally among the intervals which make up S and the actual distribution resulting from these conflicting tendencies is not easily predicted.

The Lagrange basis for π_n is especially suited for the efficient and stable implementation of the Remes algorithm because one can obtain the polynomial p associated with the current point sequence $\mathbf{t}$ without any computation, because the basis is well conditioned near the optimal $\mathbf{t}$, and because, in the end, the zeros of $P_n = p^*/p^*(0)$ are particularly easily obtained from this form.

For efficiency in evaluating p away from $\mathbf{t}$ one should express p as

$$p(t) = \prod_{j=1}^{n+1} (t-t_j) \sum_{j=1}^{n+1} \frac{\alpha_j}{t-t_j},$$

where

$$\alpha_j := \frac{p(t_j)}{\prod_{i\neq j}(t-t_i)}$$

would be calculated once and for all. Also, the derivative of p at t_m is efficiently computed by

$$p'(t_m) = \prod_{j\neq m} (t_m-t_j) \sum_{j\neq m} \frac{\alpha_m+\alpha_j}{t_m-t_j}.$$

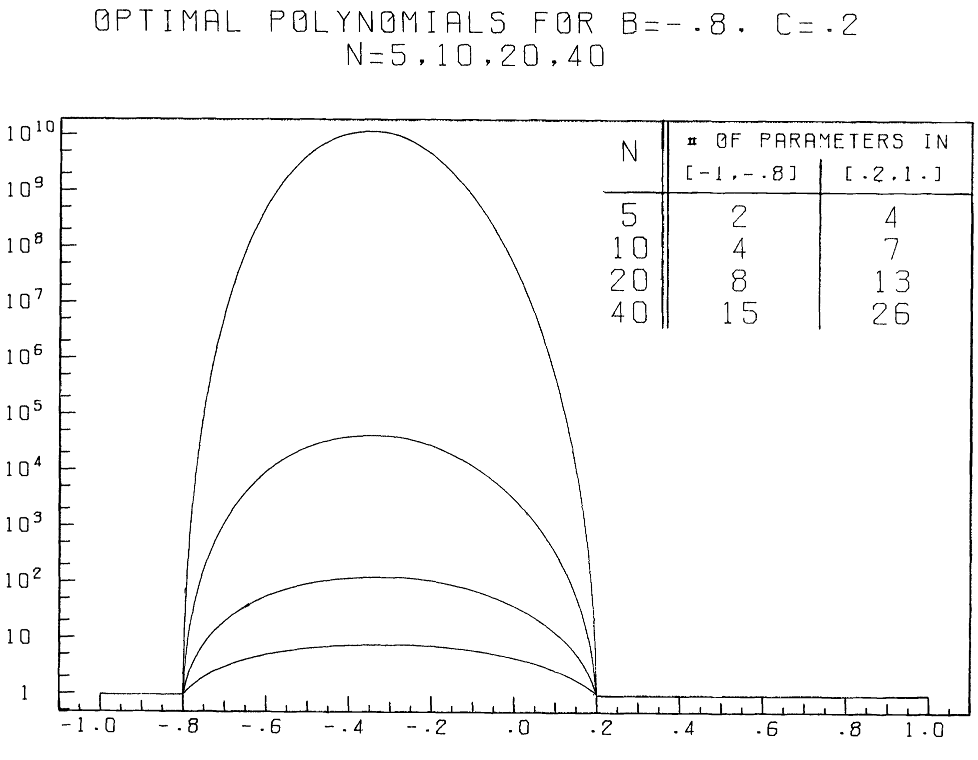
The interior local extrema of p are estimated by parabolic interpolation. As a first step, the unique extremum x^*, say, of the parabola matching p at t_j, t_j and x is found, with $x = t_{j-1}$ or t_{j+1} depending on the sign of $p'(t_j)$. The unique extremum of the parabola matching p at t_j, t_j, and x^* is then taken as the suitable approximation to the desired local extremum of p. This is a version of the standard technique for locating local extrema for use in the Remes algorithm; it is sufficiently accurate for quadratic convergence of the algorithm. Note that in our particular situation there is no need to make a global search for extrema as we know exactly where all the extrema must be.

Once the extremal polynomial p^* is found sufficiently accurately, then its zeros are found by the Modified Regula Falsi. The zeros are already bracketed by the t_j, except for one which is outside the interval $[t_1, t_{n+1}]$. This one may be to the left or right of $[t_1, t_{n+1}]$ and may actually be at infinity. We make the transformation $x = 1/t$ and then apply the same method to $[x_1, x_{n+1}]$.

Maehly's second method (see Maehly [1963]) is an alternative method for computing P_n. Its attraction is that it operates directly on the representation of p as $\prod_{j=1}^{n+1} (1-y_jt)$ and thus does not need the second phase, the computations of the zeros. We judge this approach to be less efficient overall because of the added work of solving for the parameters z_j of the new polynomial each time $\mathbf{t}$ is replaced by $\tilde{\mathbf{t}}$. This work involves the solution of $n+1$ simultaneous equations with a full coefficient matrix. We have not tried this approach; see Dunham [1966] for some remarks concerning improved convergence of this method.

5. Properties of the parameters and convergence rate of the iteration. An examination of particular examples of the extremal polynomials shows that they do not, in general, belong to orthogonal polynomial families and that they are not generated by a three-term recurrence relation. It follows from a classical result of Fekete (see Widom [1969]) that $\|P_n\|^{1/n}$ converges as n tends to infinity. Consequently, convergence of Richardson's first order method with these parameters is geometric. Of course, in practice, one is not likely to use the parameters from $P_n, P_{n+1}, P_{n+2}, \cdots$ in sequence, but is likely to use the parameters for a fixed n cyclically. One still obtains geometric convergence, but examples (such as seen below) show that n should be rather large in order to exploit the method's potential fully.

In order to judge the convergence rates one could expect, we present in Fig. 1 the graphs of p^* on the interval $[b, c]$ for the case $S=[-1, -.8]\cup[.2, 1]$ and for various values of n. Recall that $P_n = p^*/p^*(0)$, hence $\|P_n\| = 1/p^*(0)$. Further, it is worthwhile at this point to realize that a linear change in the independent variable leaves P_n essentially unchanged. In other words, if this particular S is obtained from some interval pair $S^\# = [a^\#, b^\#]\cup[c^\#, d^\#]$ by the linear change $t = f(t^\#)$, so that $-1 = f(a^\#)$, $-.8 = f(b^\#)$, etc., then the polynomial $P_n^\#$ for $S^\#$ is simply $P_n \cdot f$. In particular, then $\|P_n^\#\| = 1/p^*(f(0))$, thus $1/p^*(t)$ runs through the possible values of such $\|P_n^\#\|$ as t runs between $-.8$ and $.2$. Thus as one moves from $b = -.8$ to $c = .2$, along one of the curves for fixed n, one sees the effect on the achievable error reduction of the location of the origin between the two intervals comprising an interval pair. Note that the rate of convergence becomes 1 and the linear system becomes (possibly) singular as $f(0)$ approaches b or c.



N	# OF PARAMETERS IN	
	[-1,-.8]	[.2,1.]
5	2	4
10	4	7
20	8	13
40	15	26

FIG. 1

Figure 2 shows the dependence of the rate of convergence on the relative sizes of the two intervals which make up S. A contour plot is given of the maximum possible rate of convergence as b and c vary while $a=-1$ and $d=1$ remain fixed. This maximum rate occurs at the point where p^* is at a maximum (between b and c) which depends on b and c. This rate approaches 1 as $c-b$ approaches 0 and becomes quite fast as b and c approach -1 and -1, respectively.

Figure 3 indicates the effect of near singularity of the linear system on the rate of convergence. Again for S an interval pair and for 10 parameters, we plot contours of the logarithm of the slope of p^* at $t=c$. The larger this slope, the less the rate of convergence is degraded by the origin being close to c.

Both Figs. 2 and 3 exhibit somewhat erratic behavior due to the fact that the number of t_j's in each interval is a discrete function of b and c. This suggests that it

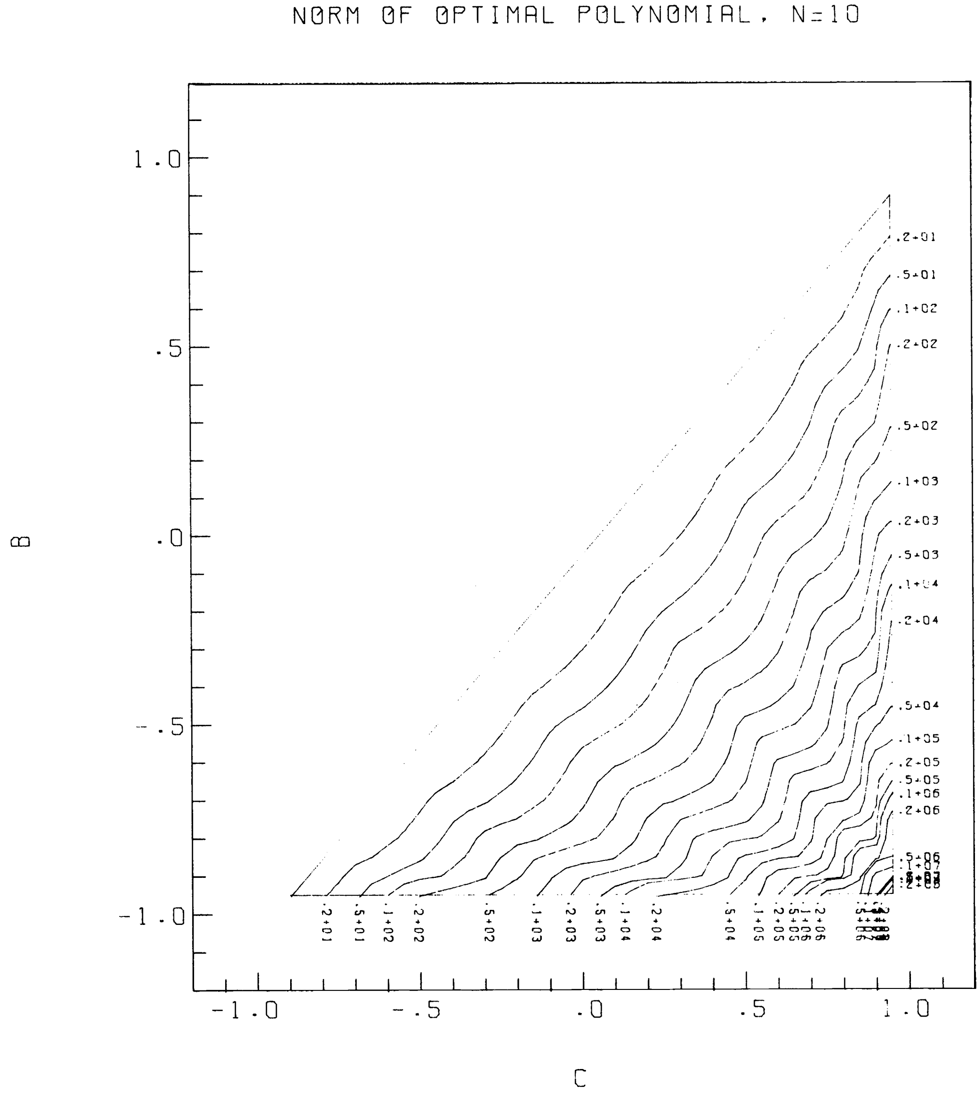


FIG. 2

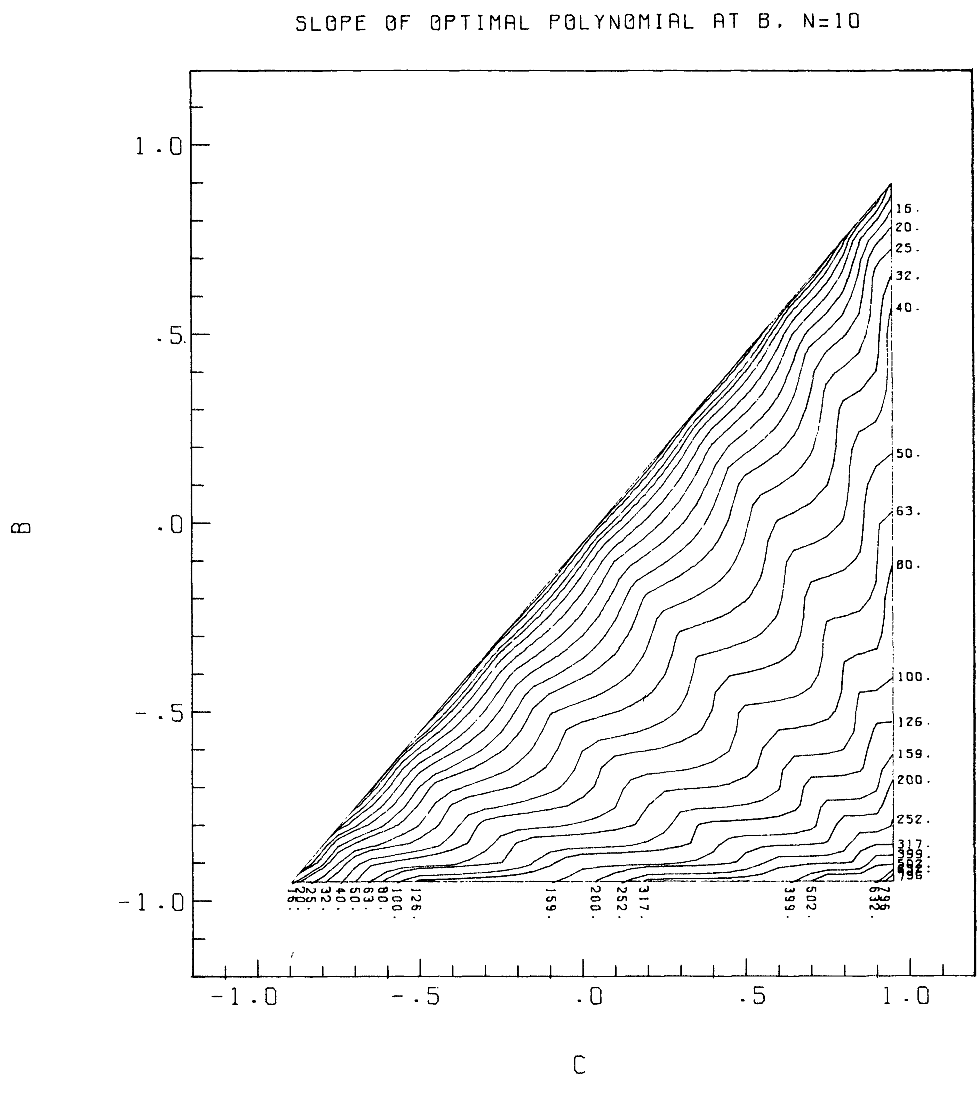


FIG. 3

would be quite difficult to obtain accurate and simple approximation formulae for the parameter distribution.

Acknowledgments. We are pleased to thank Gene H. Golub for providing us with a copy of Atlestam's report and for several helpful discussions. Further, we thank a referee for pointing out an oversight and thank Joe Grcar and Paul Saylor for providing us with a counterexample of our (former) assertion that the map T in § 3 is continuous. Finally, we thank Frederick Sauer for carrying out the computations on which the figures are based. A copy of his program, which realizes the Remes algorithm given in § 3, may be obtained by writing to him at the Mathematics Research Center.

REFERENCES

N. I. AKHIESER [1928], *Über einige Funktionen, die in gegebenen Intervallen am wenigsten von Null abweichen*, Izv. Kazanskovo Fiz.-Mat. Obshchestva.

R. S. ANDERSSEN AND G. H. GOLUB [1972], *Richardson's non-stationary matrix iterative procedure*, Rep. STAN-CS-72-304, Computer Science Dept., Stanford Univ., Stanford, CA.

B. ATLESTAM [1977], *Tschebycheff-polynomials for sets consisting of two disjoint intervals with application to convergence estimates for the conjugate gradient method*, Res. Rep. 77.06R, Dept. Computer Sciences, Chalmers University of Technology and the University of Göteborg, Sweden.

C. B. DUNHAM [1966], *Convergence problems in Maehly's second method*. II, J. Assoc. Comput. Mach., 13, pp. 108–113.

V. I. LEBEDEV [1969], *An iteration method for the solution of operator equations with their spectrum lying on several intervals*, Zh. Vych. Mat. i Fiz., 9, pp. 1247–1252; an English translation has appeared in R. S. Anderssen and G. H. Golub [1972].

H. J. MAEHLY [1963], *Methods for fitting rational approximations. Parts* II *and* III, J. Assoc. Comput. Mach., 10, pp. 257–277.

C. C. PAIGE AND M. A. SAUNDERS [1975], *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 pp. 617–629.

T. J. RIVLIN [1974], *The Chebyshev Polynomials*, John Wiley, New York.

R. R. ROLOFF [1979], *Iterative solutions of matrix equations for symmetric matrices possessing positive and negative eigenvalues*, Ph.D. thesis, Dept. Computer Science, Univ. Illinois at Urbana-Champaign, Urbana, IL.

H. WIDOM [1969], *Extremal polynomials associated with a system of curves in the complex plane*, Adv. Math., 3, pp. 127–232.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 1, March 1982

SOLVING SYMMETRIC-DEFINITE QUADRATIC λ-MATRIX PROBLEMS WITHOUT FACTORIZATION*

DAVID S. SCOTT† AND ROBERT C. WARD‡

Abstract. Algorithms are presented for computing some of the eigenvalues and their associated eigenvectors of the quadratic λ-matrix $M\lambda^2+C\lambda+K$. M, C and K are assumed to have special symmetry-type properties which insure that theory analogous to the standard symmetric eigenproblem exists. The algorithms are based on a generalization of the Rayleigh quotient and the Lanczos method for computing eigenpairs of standard symmetric eigenproblems. Monotone quadratic convergence of the basic method is proved. Test examples are presented.

Key words. eigenvalues, eigenvectors, symmetric definite quadratic λ-matrices, Lanczos algorithm

1. Introduction. Quadratic λ-matrix problems consist of determining scalars λ, called eigenvalues, and corresponding $n\times 1$ nonzero vectors x, called eigenvectors, such that the equation

$$(M\lambda^2+C\lambda+K)x=0 \tag{1}$$

is satisfied, where M, C and K are given $n\times n$ matrices. In addition, we assume that M, C and K are real symmetric or Hermitian, M is definite (either positive or negative definite) and the eigenvalues of (1) are real and can be divided into two equal disjoint sets, $\mathcal{P}$ and $\mathcal{S}$, with the following properties:

P1) If $\lambda_i\in\mathcal{P}$ and $\lambda_j\in\mathcal{S}$, then $\lambda_i>\lambda_j$.

P2) If $\lambda_i\in\mathcal{P}(\mathcal{S})$ and x_i is its associated eigenvector, then λ_i is the larger (smaller) root of the quadratic equation $(x_i^*Mx_i)\lambda^2+(x_i^*Cx_i)\lambda+(x_i^*Kx_i)=0$.

The eigenvalues in $\mathcal{P}$ will be called primary eigenvalues, and those in $\mathcal{S}$ will be called secondary. Their eigenvectors will be referenced similarly.

In this paper we present a basic method and algorithms for computing eigenpairs of (1) when factorization of M, C, K or any combination of them is either impossible or undesirable, such as might be the case if M, C and K are extremely large and sparse. (See Scott [9] for alternatives when factorization is possible, and Ruhe [7] or Lancaster [4] for a discussion of λ-matrices in a more general setting.)

Problems of this nature occur in several application areas; we will briefly discuss two of them. Lancaster [3] states that the determination of sinusoidal solutions to the equations of motion for vibrating systems which are heavily damped results in such a quadratic λ-matrix problem. In these overdamped systems M, C and K are real symmetric, M and C are positive definite, K is nonnegative definite, and the overdamping condition

$$(y^*Cy)^2-4(y^*My)(y^*Ky)>0$$

is satisfied for all vectors $y\neq 0$. Proof that the eigenvalues for overdamped systems are all real and obey properties P1 and P2 above can be found in Lancaster [3]. Problem (1) also arises in the dynamic analysis of spinning structures where the gyroscopic effects cannot be ignored. (See Wildheim [12] and Lancaster [3].) In

* Received by the editors February 25, 1981. This research was sponsored by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy, under contract W-7405-eng-26 with the Union Carbide Corporation.

† Department of Computer Sciences, University of Texas, Austin, Texas 78712.

‡ Computer Sciences Division, Union Carbide Corporation, Nuclear Division, Oak Ridge, Tennessee 37830.

gyroscopic systems M, C and K are real symmetric (Hermitian), M is negative definite and K is positive definite. One can easily determine that all the eigenvalues are real, that $\mathcal{P}$ and $\mathcal{S}$ are the positive and negative eigenvalues, respectively and that properties P1 and P2 are satisfied. In both overdamped and gyroscopic systems, the M matrix is usually called the mass matrix and K the stiffness matrix. For spinning structures, C is the Coriolis matrix. Thus, we have chosen the notation given in (1) rather than the more standard mathematical notation using A, B and C for the matrices.

Due to the simplicity of the properties of gyroscopic systems, our model problem for presentation and analysis of algorithms will be from this application area, except where explicitly noted otherwise. That is, we will discuss algorithms for computing eigenpairs of (1) where M, C and K are real symmetric, M is negative definite and K is positive definite.

It can be easily verified that the eigenvalue–eigenvector pair (λ, x) satisfies (1) if and only if it also satisfies

$$\left\{\begin{bmatrix} 0 & K \\ K & C \end{bmatrix} - \lambda \begin{bmatrix} K & 0 \\ 0 & -M \end{bmatrix}\right\}\begin{bmatrix} x \\ \lambda x \end{bmatrix} = 0. \tag{2}$$

Equation (2) is a generalized eigenvalue problem which will be denoted by $(A - \lambda B)z = 0$. Also, the matrix $A - \lambda B$ is frequently referred to as the linear pencil (A, B). Note that the A matrix is symmetric and the B matrix is symmetric and positive definite.

Scott [10] has developed and analyzed an algorithm for computing a few of the smallest eigenvalues of generalized eigenvalue problems without factorization. However, it would be inefficient in time and storage to directly apply his algorithm to (2) since the algorithm would be searching for an n-dimensional solution in a $2n$-dimensional space. The method presented in this paper attacks the problem directly in terms of (1).

In § 2 we discuss generalizations of the Rayleigh quotient and present one which is particularly attractive when working with quadratic λ-matrices. Some theoretical results on which the method is based are presented in § 3. The basic vector algorithm and its properties are presented in § 4, with the block algorithm presented in § 5. The last two sections contain the numerical results of some test problems and conclusions.

2. Rayleigh quotient generalizations. Given any nonzero vector z, the Rayleigh quotient for the linear pencil determines the "best" scalar using z as its associated eigenvector which most closely approximates an eigenvalue. The definition of best is based on the following theorem (see Parlett [5]):

THEOREM 1. *For any nonzero vector z and scalar σ, there is an eigenvalue λ of the linear pencil (A, B) such that*

$$|\lambda - \sigma| \leqq \frac{\|(A - \sigma B)z\|_{B^{-1}}}{\|Bz\|_{B^{-1}}}, \tag{3}$$

where $\|x\|_{B^{-1}} \equiv \sqrt{x^* B^{-1} x}$.

The Rayleigh quotient $\rho_1(z)$ minimizes the above bound over all scalars σ and is given by

$$\rho_1(z) = \frac{z^* A z}{z^* B z}. \tag{4}$$

Thus, in that sense, $(\rho_1(z), z)$ is the best approximation to an eigenpair of the linear pencil when z is given. We would like to extend this concept to the quadratic λ-matrix problem.

Lancaster [3] has rewritten (4) in terms of an equation equivalent to (2) to produce a generalization of the Rayleigh quotient for λ-matrices of arbitrary order. For quadratic λ-matrices this generalization states that given any nonzero vector x and a scalar α, the "best" approximation to an eigenvalue with x as its eigenvector is given by

$$\rho_L(\alpha, x) = \alpha - \frac{x^*(M\alpha^2 + C\alpha + K)x}{x^*(2M\alpha + C)x}.$$

Best in this generalization means minimizing a bound similar to (3) when the scalar α is used in the vector z in place of the eigenvalue λ (cf. (2)). The main disadvantage of $\rho_L(\alpha, x)$ is this lack of freedom in choosing the scalar to use in the vector z.

Another generalization of the Rayleigh quotient can be obtained by combining (2), (3) and (4) to form a new minimization problem. It will be helpful in describing this generalization and for the remainder of the paper to make the following definitions:

$$k(x) \equiv x^*Kx, \quad c(x) \equiv x^*Cx, \quad m(x) \equiv x^*Mx.$$

We know that given the vector $z(\sigma) = \left[\begin{smallmatrix} x \\ \sigma x \end{smallmatrix}\right]$, the Rayleigh quotient for the linear pencil is given by (4), which reduces to the following when considered as a function of σ only:

$$\tau(\sigma) = \frac{\sigma^2 c(x) + 2\sigma k(x)}{k(x) - \sigma^2 m(x)}.$$

Thus, a generalization of the Rayleigh quotient can be obtained by solving the following problem for σ_0 and using $\tau(\sigma_0)$:

$$\min_{\sigma} \frac{\|(A - \tau(\sigma)B)z(\sigma)\|_{B^{-1}}}{\|Bz(\sigma)\|_{B^{-1}}}.$$

The function to be minimized is a rational function with a sixth degree polynomial in σ in both the numerator and denominator. Also, some of the coefficients are dependent upon M^{-1}. Due to the factorization required to determine M^{-1} or to solve the appropriate linear system, this generalization could not be used to solve our stated problem. However, this approach was tried on some small test problems and appeared to work quite well.

Given any nonzero vector x, potential eigenvectors z of the linear pencil (A, B) would be linear combinations of the vectors $\left[\begin{smallmatrix} x \\ 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 \\ x \end{smallmatrix}\right]$. Using the Rayleigh–Ritz procedure, the "best" approximation to two eigenvalues and eigenvectors using vectors in this space can be determined. Best in this context means minimizing the Frobenius norm of the 2×2 scaled residual matrix (see Parlett [5]). The characteristic equation of the reduced linear pencil in the Rayleigh–Ritz procedure is the quadratic equation

$$\sigma^2 m(x) + \sigma c(x) + k(x) = 0. \tag{5}$$

Thus, the approximations to two eigenvalues of the quadratic λ-matrix are given by its roots, which can be easily determined by

$$\rho_2(x) = \frac{c(x) \pm \sqrt{c^2(x) - 4m(x)k(x)}}{-2m(x)}. \tag{6}$$

Some observations concerning $\rho_2(x)$ are noted below:

(i) $\rho_2(x)$ may take on two different values corresponding to the choice of sign in the quadratic formula. One value will always be positive and the other negative since $m(x) < 0$ and $k(x) > 0$ for all vectors x. If an approximation to a positive eigenvalue (i.e., primary eigenvalue in the general problem) is desired, then the + sign should be chosen, with the − sign selected to approximate a negative (secondary) eigenvalue.

(ii) If x is an eigenvector of the quadratic λ-matrix, one of the values of $\rho_2(x)$ will be an eigenvalue. If x is a double eigenvector, that is, an eigenvector for two different eigenvalues, both values of $\rho_2(x)$ will be eigenvalues.

(iii) $\rho_L(\alpha, x)$ derived by Lancaster is just one Newton step from α for computing a root of the quadratic equation (5). Since $\rho_2(x)$ computes the root directly, $\rho_2(x)$ would be expected to outperform $\rho_L(\alpha, x)$ and not suffer from the potential defects of Newton's method.

Due to the simplicity of its determination and to its properties presented above and in the next section, we will use $\rho_2(x)$ as the generalization of the Rayleigh quotient appropriate for the particular class of quadratic λ-matrices under discussion in this paper. The values $\rho_2(x)$ are identical to the primary and secondary functionals discussed by Duffin [1]. Duffin establishes a minimax characterization of $\rho_2(x)$; however, he does not present a theoretical basis for how and why the pair $(\rho_2(x), x)$ most closely approximates an eigenpair of the quadratic λ-matrix.

3. Eigensystem properties. For this section it will be convenient to denote the positive value of $\rho_2(x)$ by $\rho_2^+(x)$ and the negative value by $\rho_2^-(x)$. The matrix $M\sigma^2 + C\sigma + K$ will be denoted by $W(\sigma)$ and the quadratic polynomial $\sigma^2 m(x) + \sigma c(x) + k(x)$ by $Q_x(\sigma)$. Also, we refer to the eigenvalues of the quadratic λ-matrix by λ_i with $\lambda_{-n} \leqq \lambda_{-n+1} \leqq \cdots \leqq \lambda_{-1} < 0 < \lambda_1 \leqq \cdots \leqq \lambda_n$.

The next three theorems and two lemmas are similar to those found in Duffin [1], Lancaster [3] and Rogers [6] for overdamped systems; thus, their proofs are omitted.

THEOREM 2. *$M\lambda^2 + C\lambda + K$ is a simple λ-matrix.*

LEMMA 1. *For any $x \neq 0$, $Q_x'(\rho_2^+(x)) < 0$ and $Q_x'(\rho_2^-(x)) > 0$.*

LEMMA 2. *Let λ_i, λ_j be positive eigenvalues with eigenvectors x_i, x_j, respectively and $\lambda_i < \lambda_j$. Then x_i, x_j are linearly independent, and if x is any linear combination of x_i and x_j, then*

$$\lambda_i \leqq \rho_2^+(x) \leqq \lambda_j. \tag{7}$$

THEOREM 3. *Let $\lambda_{i_1} < \lambda_{i_2} < \cdots < \lambda_{i_m}$ be positive eigenvalues with eigenvectors $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$, respectively. Then these eigenvectors are linearly independent, and if x is any linear combination of them,*

$$\lambda_{i_1} \leqq \rho_2^+(x) \leqq \lambda_{i_m}.$$

Lemma 2 and Theorem 3 remain true if the word positive is replaced by negative and $\rho_2^+(x)$ is replaced by $\rho_2^-(x)$. Also, it can be easily shown that for every γ between λ_{i_1} and λ_{i_m} there exists a vector x which is a linear combination of the eigenvectors $x_{i_1}, x_{i_2}, \cdots, x_{i_m}$ such that $\rho_2^+(x)$ equals γ.

THEOREM 4. *The eigenvectors corresponding to the positive eigenvalues span the space consisting of all real vectors of dimension n, and so do the eigenvectors corresponding to the negative eigenvalues.*

When using vector-type iterations to converge to eigenpairs, schemes are frequently employed to control the iteration so that convergence to previously determined eigenpairs or undesirable eigenpairs is prevented. From linear pencil theory and (2), we can derive the following orthogonality relationship for eigenpairs (λ_i, x_i) of the quadratic λ-matrix, when $i \neq j$:

$$x_i^* K x_j - \lambda_i \lambda_j x_i^* M x_j = 0. \tag{8}$$

Unfortunately, the appearance of both λ_i and λ_j in this expression means that the eigenvalue to which we are trying to converge, say λ_j, must be known to prevent the vector iteration from converging to x_i. In the algorithm presented in the next section, the following theorem (Scott [9]) provides us with the ability to control our iterative process:

THEOREM 5. *The number of negative eigenvalues of $W(\sigma)\equiv M\sigma^2+C\sigma+K$ equals the number of eigenvalues of the quadratic eigenproblem between σ and* 0.

4. The basic algorithm and analysis. In § 2 we have presented a generalization of the Rayleigh quotient which determines the "best" approximation to an eigenvalue given an approximate eigenvector. If a method for computing the "best" approximation to an eigenvector given an approximate eigenvalue is determined, then the basis of an algorithm would exist. We observe that if σ is an eigenvalue of the quadratic λ-matrix with x as its eigenvector, $W(\sigma)$ has the eigenpair $(0, x)$. Recalling Theorem 5 and that convergence to the eigenvalues of smallest magnitude is desired, the appropriate eigenvector of $W(\sigma)$ to choose for our iteration corresponds to the kth most negative eigenvalue of $W(\sigma)$ when convergence to the kth smallest positive or negative eigenvalue is desired. The Lanczos algorithm (see Parlett [5]) does an effective job in computing this eigenvector when k is not large with respect to n. This leads to the following algorithm for converging to the m smallest positive eigenvalues:

```
THE QUADRATIC RAYLEIGH QUOTIENT (QRQ) ALGORITHM
 I) Set the vector $x_0$ to random numbers.
II) For $k=1, 2, \cdots, m$ do 1) and 2),
    1) For $i=1, 2, \cdots$ until convergence do a) and b),
       a) Set $\sigma_i=\rho_2^+(x_{i-1})$.
       b) Set $x_i=y_k$ where $(\theta_j, y_j)$ are eigenpairs of $W(\sigma_i)$ with $\theta_1\leqq\theta_2\leqq\cdots\leqq\theta_n$
          and $y_j$ unit-length vectors.
    2) Set $x_0$ to the $y_{k+1}$ computed in step 1b.
```

An educated guess, if known, could be used for initializing x_0 in step I.

Similar algorithms exist for converging to the other extreme positive and negative eigenvalues of (1). To converge to the m smallest negative eigenvalues, $\rho_2^+(x_{i-1})$ in step II.1b should be replaced by $\rho_2^-(x_{i-1})$. To converge to the eigenpairs associated with the largest positive or negative eigenvalues, y_k in step II.1b should be replaced by y_{n-k+1} and y_{k+1} in step II.2 by y_{n-k}.

In our analysis of the QRQ algorithm, we will concentrate on convergence to λ_1, the smallest positive eigenvalue of the quadratic λ-matrix, which may be a multiple eigenvalue. The theorems and proofs for convergence to the other most extreme eigenvalues in $\mathcal{P}$ and $\mathcal{S}$ are very similar. Comments concerning convergence to the interior eigenvalues will be given at the end of this section.

THEOREM 6. *The sequence $\{\sigma_i\}$ determined by the* QRQ *algorithm converges monotonically downward to λ_1.*

Proof. Theorem 4 states that the initial vector x_0 is a linear combination of the eigenvectors corresponding to the positive eigenvalues. Therefore, $\sigma_1\equiv\rho_2^+(x_0)\geqq\lambda_1$ by Theorem 3. For $i=1, 2, \cdots$ $W(\sigma_i)\,x_i=\theta_1 x_i$, and since x_i is unit-length,

$$Q_{x_i}(\sigma_i)=m(x_i)\sigma_i^2+c(x_i)\sigma_i+k(x_i)=\theta_1,$$

with $\theta_1\leqq 0$ by Theorem 5. Recalling that $m(x_i)<0$ and that σ_{i+1} is the positive root of $Q_{x_i}(\sigma)$, σ_{i+1} is less than or equal to σ_i with equality holding only when $\theta_1=0$, that is, only when $\sigma_i=\sigma_{i+1}=\lambda_1$. Thus, $\{\sigma_i\}$ is a decreasing sequence converging to λ_1. □

THEOREM 7. *The convergence of the sequence* $\{\sigma_i\}$ *determined by the* QRQ *algorithm to* λ_1 *is asymptotically quadratic.*

Proof. The theorem could be proved by relying on perturbation theory for quadratic polynomials; however, we shall present the straightforward proof which does not rely on the theory of equations.

Let $\lambda_1 = \sigma_i - \varepsilon$. For ε less than the smallest nonzero eigenvalue of $M\lambda_1^2 + C\lambda_1 + K$, we can determine a bounded β such that $x_i = \alpha x + \varepsilon\beta w$, where (λ_1, x) is an eigenpair of the quadratic λ-matrix; that is, x is unit-length and in the null space of $M\lambda_1^2 + C\lambda_1 + K$, and w is unit-length and orthogonal to this null space. Note that α is also bounded. We have

$$x_i^*(M\sigma_i^2 + C\sigma_i + K)x_i = \theta_1, \tag{9}$$

which implies

$$\varepsilon^2\beta^2 w^*(M\lambda_1^2 + C\lambda_1 + K)w + (\alpha x + \varepsilon\beta w)^*[(2M\lambda_1 + C)\varepsilon + M\varepsilon^2](\alpha x + \varepsilon\beta w) = \theta_1.$$

Therefore,

$$\theta_1 = \alpha^2 x^*(2M\lambda_1 + C)x\varepsilon + O(\varepsilon^2). \tag{10}$$

From (6) and (9), we have

$$\sigma_{i+1} = \sigma_i + \frac{\sqrt{c^2(x_i) - 4m(x_i)k(x_i)} - \sqrt{c^2(x_i) - 4m(x_i)[k(x_i) - \theta_1]}}{-2m(x_i)}.$$

Expanding the second term in the numerator of the above expression by a Taylor series around the first term, which is never zero by Lemma 1, yields

$$\sigma_{i+1} = \sigma_i + \frac{\theta_1}{\sqrt{c^2(x_i) - 4m(x_i)k(x_i)}} + O(\theta_1^2)$$

and, from (6),

$$\sigma_{i+1} = \sigma_i + \frac{\theta_1}{-2m(x_i)\sigma_{i+1} - c(x_i)} + O(\theta_1^2). \tag{11}$$

Now, $-2m(x_i)\sigma_{i+1} - c(x_i) = -x_i^*[2M(\lambda_1 + \delta\varepsilon) + C]x_i$ where $\delta \geqq 1$ by Theorem 6. Substituting for x_i, we have

$$-2m(x_i)\sigma_{i+1} - c(x_i) = -\alpha^2 x^*(2M\lambda_1 + C)x + O(\varepsilon). \tag{12}$$

Using (10) and (12) in (11) results in

$$\sigma_{i+1} = \sigma_i - \frac{\alpha^2 x^*(2M\lambda_1 + C)x\varepsilon + O(\varepsilon^2)}{\alpha^2 x^*(2M\lambda_1 + C)x + O(\varepsilon)} = \sigma_i - \varepsilon + O(\varepsilon^2) = \lambda_1 + O(\varepsilon^2). \qquad \square$$

Convergence of the basic algorithm to the eigenvalues other than the extreme positive and negative eigenvalues cannot be guaranteed, as the following example, similar to an example in Scott [10], illustrates:

$$\begin{bmatrix} -1 & & \\ & -6 & \\ & & -3 \end{bmatrix}\lambda^2 + \begin{bmatrix} 1 & & \\ & 3 & \\ & & 1 \end{bmatrix}x = 0.$$

The eigenvalues of the quadratic λ-matrix are ± 1, $\pm(1/\sqrt{2})$, $\pm(1/\sqrt{3})$, with eigenvectors e_1, e_2 and e_3 respectively, for both of the $\pm$ values. The vector e_i is the ith column of the identity matrix. $W(\sigma)$ is a diagonal matrix with eigenvalues $-\sigma^2 + 1$, $-6\sigma^2 + 3$

and $-3\sigma^2+1$ down the diagonal and e_1, e_2 and e_3 as their respective eigenvectors. Therefore, in the QRQ algorithm, after converging to the smallest positive eigenvalue $1/\sqrt{3}$, the initial vector x_0 for starting the iteration to converge to the second smallest positive eigenvalue would be e_1. The sequence σ_i for $i = 1, 2, \cdots$ would then alternate between 1 and $1/\sqrt{3}$ (x_i would alternate between e_1 and e_3) and would never approach the eigenvalue $1/\sqrt{2}$. This oscillation rarely happens in actual problems, and convergence to these eigenvalues can be expected. However, an extension of the basic algorithm, which incorporates a block of m vectors X_i instead of the single vector x_i, guarantees monotonic quadratic convergence to the m extreme eigenpairs as discussed in the following section.

5. The block algorithm. Comparing the above example of nonconvergence for an interior eigenvalue to the theory for an extreme eigenvalue, it is clear that it is the lack of monotonicity of the iterates that allows for the nonconvergence to interior eigenvalues. Convergence will still be asymptotically quadratic if it occurs.

To recover global convergence for interior eigenvalues it is necessary to use a block version of the algorithm. The block size must be at least as large as the distance from the desired eigenvalue to an edge of the spectrum. That is, if the third smallest positive eigenvalue is desired, then a block size of at least three is needed. In the following discussion, we will use p to denote the block size.

Instead of computing just one eigenvector of W at each step, the block algorithm computes p of them. It then uses these p vectors as columns of the $n \times p$ matrix X to compute $2p$ approximate eigenvalues and eigenvectors of the quadratic problem (1) by solving the reduced quadratic problem

$$m(X)\theta^2 + c(X)\theta + k(X), \tag{13}$$

where $m(X) = X^*MX$, $c(X) = X^*CX$ and $k(X) = X^*KX$. This reduced problem can be solved by forming the $2p \times 2p$ linear system

$$\begin{bmatrix} 0 & k(X) \\ k(X) & c(X) \end{bmatrix} - \theta \begin{bmatrix} k(X) & 0 \\ 0 & -m(X) \end{bmatrix}$$

and using a linear pencil solver. The appropriate eigenvalue of (13) is then used to form the new W for the next iteration. At the termination of the algorithm, it is necessary to form the proper linear combination of the columns of X to determine the eigenvector of (1).

We give a summary of the algorithm for computing the m smallest positive eigenvalues.

```
THE BLOCK QUADRATIC RAYLEIGH QUOTIENT (BQRQ) ALGORITHM
I) Choose $p \geqq m$ and set $X_0 = (x_1^0, x_2^0, \cdots, x_p^0)$ to random vectors.
II) For $k = 1, 2, \cdots, m$ do 1) to 3),
    1) For $i = 1, 2, \cdots$ until convergence do a) and b),
        a) Set $\sigma_i$ to the $k$th smallest positive eigenvalue of $m(X_{i-1})\theta^2 + c(X_{i-1})\theta +$
           $m(X_{i-1})$,
        b) Set the columns of $X_i$ to the eigenvectors associated with the $p$ smallest
           eigenvalues of $W(\sigma_i)$.
    2) Compute $x_k = X_i s_k$, where $s_k$ is the eigenvector of $\sigma_i$ in step 1a.
    3) Use $X_i$ as the new $X_0$.
```

The Cauchy interlace theorem assures that the iterates in the BQRQ algorithm will be monotonic. The convergence and the asymptotic quadratic convergence can then be proved exactly as in the single vector version.

6. Numerical results. A block algorithm has been coded to compute either the smallest or largest m eigenvalues in $\mathcal{P}$ or $\mathcal{S}$ for a quadratic λ-matrix with the properties discussed in the first paragraph of § 1. This code solves for the Rayleigh–Ritz values (step II.1a) via the RGG algorithm found in EISPACK (see Garbow et al. [2]), which is based on an extension of the QZ algorithm by Ward [11] and solves the inner-loop eigenproblem (step II.1b) via the block Lanczos algorithm (see Scott [8]). Several test problems have been run using a single-precision code (≈8 decimal digits) on a PDP-10 computer at the Oak Ridge National Laboratory. Sample results are presented below with only the lower part of the symmetric matrices given.

Test case 1. *Distinct eigenvalues.*

$$M=\begin{bmatrix} -10 & & & & \\ 2 & -11 & & & \\ -1 & 2 & -12 & & \\ 1 & -2 & -1 & -10 & \\ 3 & -1 & 1 & 2 & -11 \end{bmatrix}, \qquad C=\begin{bmatrix} 1 & & & & \\ 2 & 1 & & & \\ 1 & 2 & 0 & & \\ 2 & 1 & -2 & 2 & \\ 1 & 3 & -2 & 3 & 3 \end{bmatrix},$$

$$K=\begin{bmatrix} 10 & & & & \\ 2 & 9 & & & \\ -1 & 3 & 10 & & \\ 2 & -1 & 2 & 12 & \\ -2 & -2 & -1 & 1 & 10 \end{bmatrix}.$$

This test case was generated by inserting pseudorandom integers in the matrices, making sure that the diagonally dominant M and K matrices were negative definite and positive definite respectively. The eigenvalues to 3 decimal figures are −1.27, −1.08, −1.00, −.779, −.512, .502, .880, .937, 1.47 and 1.96. The code was asked to compute the three smallest positive eigenvalues using a block size of 3 by iterating until the relative change in the eigenvalue was less than the machine precision. The code was also used to compute the 3 most negative eigenvalues requesting the same accuracy. Tables 1 and 2 present the results of the separate compute runs. The number of Lanczos steps indicate the total number of such steps taken until convergence. The number of Lanczos steps taken at any particular iteration varied.

TABLE 1

Actual eigenvalues	Iterations	Lanczos steps	Error in computed eigenvalues
.502415273	3	8	4×10^{-8}
.879927281	3	8	1×10^{-8}
.936550669	2	6	3×10^{-8}

TABLE 2

Actual eigenvalues	Iterations	Lanczos steps	Error in computed eigenvalues
−1.27188510	3	7	1×10^{-7}
−1.07716772	2	6	1×10^{-7}
−1.00483822	2	6	1×10^{-7}

Test case 2. *Multiple eigenvalues.*

$$M=\begin{bmatrix}-1 & & & \\ 1 & -2 & & \\ -1 & 1 & -2 & \\ -2 & 2 & 0 & -9\end{bmatrix},\quad C=\begin{bmatrix}-3 & & & \\ 0 & -3 & & \\ -3 & 0 & -5 & \\ -6 & 0 & -4 & -19\end{bmatrix},$$

$$K=\begin{bmatrix}1 & & & \\ -1 & 5 & & \\ 1 & -1 & 2 & \\ 2 & -2 & 0 & 14\end{bmatrix}.$$

These matrices were generated by coupling two quadratic λ-matrices of order 2 and applying a symmetric transformation to the resulting block diagonal matrices. The eigenvalues are $-4-\sqrt{19}(\approx -8.36)$, $-4-3\sqrt{2}(\approx -8.24)$, -2, -2, $-4+3\sqrt{2}(\approx .243)$, $-4+\sqrt{19}(\approx .359)$, 1 and 1. Note that the problem has two multiple eigenvalues and two eigenvalues very close for both the set of positive eigenvalues and the set of negative eigenvalues. The results from the code requesting machine accuracy for computation of the two smallest negative and positive eigenvalues are given in Tables 3 and 4, respectively. A block size of two was used in both runs. Note that multiple eigenvalues caused no computational problems; in fact, the second copy of -2 was determined after only one iteration. A consequence of using the block Lanczos algorithm is that multiple eigenvalues will be determined at the rate one per iteration as this example illustrates.

TABLE 3

Actual eigenvalues	Iterations	Lanczos steps	Error in computed eigenvalues
−2.0	3	7	0.0
−2.0	1	2	1×10^{-7}

TABLE 4

Actual eigenvalues	Iterations	Lanczos steps	Error in computed eigenvalues
.242640687	5	12	1×10^{-8}
.358898944	3	8	3×10^{-8}

Test case 3. *Sparse problem.* The lower triangular part of the symmetric M matrix contains -12's down the main diagonal, 1's down the first subdiagonal and 1's down each tenth subdiagonal (i.e., tenth, twentieth, thirtieth, $\cdots$, ninetieth subdiagonals). The lower part of the symmetric C matrix contains 1's down the first subdiagonal and -1's down the first column below its second element. The symmetric K matrix contains 4's down the main diagonal and 1's down the fiftieth super- and subdiagonals. Table 5 presents the results from requesting the computation of the 5 smallest negative eigenvalues to an accuracy of 5 decimal digits. Note that the algorithm required a fairly large number of Lanczos steps to converge to the second smallest negative eigenvalue. This slow convergence is due to the cluster of eigenvalues near $-.439$; in fact, there are eleven eigenvalues of the quadratic problem in the interval (.4392, .4489). The third through the fifth eigenvalues were determined much quicker since the spectrum of $W(\sigma)$ around $\sigma = .44$ had been essentially resolved resulting in better initial vectors for the iteration.

TABLE 5

Actual eigenvalues	Iterations	Lanczos steps	Error in computed eigenvalues
−.30231628	4	16	1×10^{-7}
−.43926127	3	30	1×10^{-7}
−.43951256	1	2	2×10^{-8}
−.43992960	1	6	3×10^{-8}
−.44051182	1	2	2×10^{-8}

7. Conclusions. A basic method has been presented to compute some eigenvalues and their associated eigenvectors of the quadratic λ-matrix (1). Monotone quadratic convergence to the extreme eigenvalues in $\mathcal{P}$ and $\mathcal{S}$ using the vector algorithm and to any eigenvalue using the block algorithm has been proved. However, the cost of computing the eigenvectors of $W(\sigma_i)$ in the inner loop will limit the practical value of either algorithm to determining only a few of the eigenpairs of (1).

REFERENCES

[1] R. J. DUFFIN, *A minimax theory for overdamped networks*, J. Rational Mech. Anal., 4 (1955), pp. 221–233.

[2] B. S. GARBOW, J. M. BOYLE, J. J. DONGARRA AND C. B. MOLER, *Matrix Eigensystem Routines—* EISPACK *Guide Extension*, Lecture Notes in Computer Science 51, Springer-Verlag, New York, 1977.

[3] PETER LANCASTER, *Lambda-Matrices and Vibrating Systems*, Pergamon Press, New York, 1966.

[4] ———, *A review of numerical methods for eigenvalue problems nonlinear in the parameter*, ISNM, 38 (1977), pp. 43–67.

[5] BERESFORD N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[6] E. H. ROGERS, *A minimax theory for overdamped systems*, Arch. Rational Mech. Anal., 16 (1964), pp. 89–96.

[7] AXEL RUHE, *Algorithms for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal., 10 (1973), pp. 674–89.

[8] D. S. SCOTT, *Block Lanczos software for symmetric eigenvalue problems*, Tech. Rep. ORNL/CSD-48, Union Carbide Corporation, Nuclear Division, Oak Ridge, TN, 1979.

[9] ———, *Solving sparse quadratic λ-matrix problems*, Tech. Rep. ORNL/CSD-69, Union Carbide Corporation, Nuclear Division, Oak Ridge, TN, 1980.

[10] ———, *Solving sparse symmetric generalized eigenvalue problems without factorization*, SIAM J. Numer. Anal., 18 (1981), pp. 102–110.

[11] ROBERT C. WARD, *The combination shift QZ algorithm*, SIAM J. Numer. Anal., 12 (1975), pp. 835–53.

[12] J. WILDHEIM, *Vibrations of rotating circumferentially periodic structures*, Quart. J. Mech. Appl. Math., 34 (1981), pp. 213–29.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 1, March 1982

THE ADVANTAGES OF INVERTED OPERATORS IN RAYLEIGH-RITZ APPROXIMATIONS*

D. S. SCOTT†

Abstract. Generalized eigenvalue problems are often solved by a combination of inverse iteration and the Rayleigh-Ritz procedure. In this paper we show that significant advantages can be obtained in this context by applying the Rayleigh-Ritz procedure to an inverted operator, either explicitly while using subspace iteration or implicitly by applying the Lanczos algorithm to the inverted operator. Since the Lanczos algorithm is much more powerful than subspace iteration it should be used whenever possible.

Key words. eigenvalues, eigenvectors, symmetric matrices, Rayleigh-Ritz procedure, Lanczos algorithm

1. Introduction. In this paper we consider the problem of computing some eigenpairs of the generalized eigenvalue problem

$$(A-\lambda M)z=0, \tag{1.1}$$

where A and M are symmetric (Hermitian) matrices and M is positive definite. To be explicit we assume that all the eigenvalues lying in some given interval $[a, b]$ are to be computed along with the corresponding eigenvectors. Such problems occur in the dynamic analysis of structures and in many other applications.

Notation. $\lambda_1 \leqq \lambda_2, \leqq \cdots, \leqq \lambda_n$ will be the eigenvalues of the pencil (A, M).

Section 2 describes subspace iteration, the most commonly used solution technique for (1.1), § 3 describes two potential problems with its usual implementation and § 4 shows that both problems can be eliminated if the Rayleigh-Ritz procedure is applied to an inverted operator. Section 5 shows that the Lanczos algorithm can be applied directly to the inverted operator without further modification. Section 6 discusses the important special case of M semidefinite, and § 7 presents a numerical example. Section 8 gives our summary and conclusions.

2. Subspace iteration. The most commonly used solution technique for generalized eigenvalue problems is subspace iteration (see Bathe and Wilson [1976] for more details), which is a combination of inverse iteration to generate subspaces and the Rayleigh-Ritz procedure to resolve the subspace into approximate eigenvectors.

Before starting the algorithm, a shift σ is chosen (in the desired interval $[a, b]$) and the triangular factorization $(A-\sigma M)=LDL^*$ is computed. The number of negative elements of D equals the number of eigenvalues of the pencil (A, M) which are less than σ. This fact is particularly useful if a sequence of shifts is used since it insures that no eigenvalue can be unknowlingly missed. The operator $(A-\sigma M)^{-1}M$ has the same eigenvectors as the original pencil (A, M) but the eigenvalues are transformed to $\mu_i = 1/(\lambda_i - \sigma)$. This means that the eigenvalue nearest σ (provided it is unique) becomes the dominant one and the power method with $(A-\sigma M)^{-1}M$ will converge to the corresponding eigenvector.

* Received by the editors February 27, 1981 and in revised form September 15, 1981. This research was sponsored jointly by the Mathematics Department, University of Linkoping, Sweden and the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract W-7405-eng-26 with the Union Carbide Corporation.

† Department of Computer Sciences, University of Texas, Austin, Texas 78712.

If more than one eigenpair is desired, then several vectors must be operated on simultaneously. To prevent all the vectors from converging to the same eigenvector, it is necessary to modify them at each step. In subspace iteration this is done by applying the Rayleigh–Ritz procedure. Unfortunately the operator $(A-\sigma M)^{-1}M$ is not symmetric, so it is unsuitable for the Rayleigh–Ritz procedure. In standard subspace iteration the original pencil (A, M) is used instead. We give a listing of the algorithm.

SUBSPACE ITERATION

Initialize: Choose σ and an initial set of vectors $X_0=(x_1, x_2, \cdots, x_j)$ and factor $A-\sigma M=LDL^*$.

Iterate: For $k=1, 2, 3, \cdots$ until convergence

1. Solve $(A-\sigma M)Y_k=MX_{k-1}$ for Y_k.
2. Compute the reduced matrices $A_k=Y_k^*AY_k$ and $M_k=Y_k^*MY_k$.
3. Solve the reduced problem $A_kG_k=M_kG_k\Theta_k$ for eigenvalues $\Theta_k=\operatorname{diag}(\theta_1, \theta_2, \cdots, \theta_j)$ and normalized eigenvectors $G_k=(g_1, g_2, \cdots, g_j)$.
4. Set $X_k=Y_kG_k$.

The Ritz values (the θ's) converge to the j eigenvalues closest to σ and the columns of X converge to the corresponding eigenvectors. In practice it is found to be cost effective to iterate with more vectors than are needed.

3. Error bounds and ghost vectors.

3. Error bounds and ghost vectors. One problem with subspace iteration as given above is the lack of a guaranteed stopping criterion. Bathe and Wilson recommend stopping when the change in the θ's from one step to the next is less than the desired accuracy. While this has been found to be sufficient for most practical problems, it would be reassuring to have error bounds available in deciding when to stop. The following theorem (Parlett [1980, p. 318]) gives the simplest bound. For any vector x, $\|x\|_{M^{-1}}\equiv\sqrt{x^*M^{-1}x}$.

THEOREM 1. *For any vector $x\neq 0$ and any scalar θ, there exists an eigenvalue λ of the pencil (A, M) with $|\lambda-\theta|\leqq\|(A-\theta M)x\|_{M^{-1}}/\|Mx\|_{M^{-1}}$.*

The M^{-1} norm in the denominator of the bound is not of concern since the M^{-1} cancels, but evaluating the numerator requires solving a system of equations in M. If M is not the identity matrix, this may be expensive or impossible (if M is only positive semidefinite). All other available bounds also involve M^{-1} norms.

The other problem can occur when the pencil (A, M) has eigenvalues outside both ends of the desired interval $[a, b]$, that is, when the desired eigenvalues are interior. In this case it is possible to have more Ritz values (θ's) in the interval $[a, b]$ than there are eigenvalues. This happens when a linear combination of eigenvalues outside both ends of $[a, b]$ is found to lie inside $[a, b]$. The mere existence of these "ghost values" makes it difficult to monitor convergence of the desired eigenvalues (see Taylor and Rollins [1978], for example), but (even worse) it is possible for ghost vectors to prevent the discovery of a good approximate eigenvector which lies in the subspace. We give a small example illustrating this phenomenon.

Example 1. Let $\varepsilon\ll 1$, $M=I$ and

$$A=\begin{bmatrix}\varepsilon & \varepsilon & 0\\ \varepsilon & \varepsilon & 1\\ 0 & 1 & \varepsilon\end{bmatrix}.$$

The eigenvalues of A are $\lambda_1=\varepsilon-\sqrt{1-\varepsilon^2}\simeq\varepsilon-1$, $\lambda_2=\varepsilon$ and $\lambda_3=\varepsilon+\sqrt{1=\varepsilon^2}\simeq\varepsilon+1$. There is only one eigenvalue near 0, and one would expect to find it quickly using inverse iteration ($\sigma = 0$). Indeed, if we take $X_0 = (1\ \ 1\ \ 1)^*$ (a common choice), then

$$Y_1 = A^{-1}X_0 \simeq \begin{bmatrix} \frac{1}{\varepsilon} \\ 1 \\ 0 \end{bmatrix},$$

and $X_1 \simeq \begin{bmatrix} \varepsilon \\ 1 \\ 0 \end{bmatrix}$ with eigenvalue estimate ε and residual norm ε, which is quite satisfactory. However, if the vector $e_3 = (0\ \ 0\ \ 1)^*$ is added to the initial subspace so that

$$X_0 = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix},$$

then

$$Y_1 = A^{-1}X_0 \simeq \begin{bmatrix} \frac{1}{\varepsilon} & -1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix},$$

which leads to reduced matrices

$$A_1 = Y_1^*AY_1 \cong \begin{bmatrix} \frac{1}{\varepsilon} & 0 \\ 0 & 0 \end{bmatrix}$$

and

$$M_1 = Y_1^*Y_1 = \begin{bmatrix} \frac{1}{\varepsilon^2} & -\frac{1}{\varepsilon} \\ \frac{1}{\varepsilon} & 2 \end{bmatrix},$$

and finally to $\theta_1 = 0$ and $\theta_2 = 2\varepsilon$ with

$$X_1 = 2^{-1/2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 0 & 0 \end{bmatrix}.$$

Both θ's are near 0, and neither column of X_1 is near an eigenvector (each has residual norm $2^{-1/2}$). Thus, the addition of e_3 to the initial subspace has concealed the good approximate eigenvector e_1.

We show in the next section that both of these difficulties can be eliminated by applying the Rayleigh–Ritz procedure directly to an inverted operator.

4. Inverted operators. The eigenvalues of $(A-\sigma M)^{-1}$ are equal to $1/(\lambda_i - \sigma)$. Thus, the problem of finding all the eigenvalues of (A, M) inside the interval $[a, b]$ is transformed into finding all the eigenvalues of $(A-\sigma M)^{-1}M$ outside the interval $[1/(a-\sigma), 1/(b-\sigma)]$. Since these are the extreme eigenvalues, it is impossible for linear combinations of unwanted eigenvalues to lie in the range of interest. Therefore the problem of ghost values could be eliminated if the Rayleigh–Ritz procedure could be applied to $(A-\sigma M)^{-1}M$. Unfortunately, as mentioned previously, the operator $(A-\sigma M)^{-1}M$ is not symmetric and so is unsuitable for the Rayleigh–Ritz procedure.

There are two ways to symmetrize the operator $(A-\sigma M)^{-1}M$, which are equivalent if $M=I$.

1. Factor $M=LL^*$ and use $L^*(A-\sigma M)^{-1}L$. The operator $L^*(A-\sigma M)^{-1}L$ is symmetric and has the same eigenvalues as $(A-\sigma M)^{-1}M$. Hence the desired eigenvalues are the extreme ones and no ghosts can occur. Furthermore, since this method leads to a standard eigenvalue problem there is no need to compute M^{-1} norms of residuals and it is possible to use the Lanczos algorithm. See Ericsson and Ruhe [1980] for more details. The major drawback of the method is the required factorization of M. There is also the nuisance of back transforming the computed eigenvectors, since the eigenvectors of $L^*(A-\sigma M)^{-1}L$ are different from those of the pencil (A, M).

2. Use the pencil $(M(A-\sigma M)^{-1}M, M)$. This pencil has the same eigenvalues and *eigenvectors* as the operator $(A-\sigma M)^{-1}M$. Therefore, there are no ghost vectors and no need to back transform the computed eigenvectors. Furthermore, the computation of the bound in Theorem 1 does not require the factorization of M since the M^{-1} cancels in evaluating $\|M(A-\sigma M)^{-1}Mx-\theta Mx\|_{M^{-1}}$. The only apparent drawback to method 2 is the inability to use the Lanczos algorithm. However, in the next section we will show that it is possible to run the Lanczos algorithm on the pencil $(M(A-\sigma M)^{-1}M, M)$ without factoring M.

In either case it is necessary to back transform the computed eigenvalues using the formula

$$\tau_i=\sigma+\frac{1}{\theta_i},$$

where θ_i is the ith computed eigenvalue. We now return to Example 1 and examine the effect of applying the Rayleigh–Ritz procedure to the inverted operator A^{-1} $(\sigma=0)$.

$$A^{-1}=\begin{bmatrix}\frac{1}{\varepsilon}-\varepsilon & \varepsilon & -1\\ \varepsilon & -\varepsilon & 1\\ -1 & 1 & 0\end{bmatrix}\simeq\begin{bmatrix}\frac{1}{\varepsilon} & \varepsilon & -1\\ \varepsilon & -\varepsilon & 1\\ 1 & 1 & 0\end{bmatrix}.$$

The eigenvalues of A^{-1} are approximately $1/(\varepsilon-1)$, $1/(\varepsilon+1)$ and $1/\varepsilon$. The desired eigenvalue is now $1/\varepsilon$, which is the dominant eigenvalue of A^{-1}. The vector $(1\ \ \varepsilon\ \ 0)^*$ is still a good approximate eigenvector (as before), but the addition of e_3 to the initial subspace does not obscure this good vector when the inverted operator is used for the Rayleigh–Ritz procedure as shown below. Recalling from before that

$$Y_1=\begin{bmatrix}\frac{1}{\varepsilon} & -1\\ 1 & 1\\ 0 & 0\end{bmatrix}$$

leads to

$$A_1=Y_1^*A^{-1}Y_1=\begin{bmatrix}\frac{1}{\varepsilon^3}-\frac{1}{\varepsilon}+2-\varepsilon & -\frac{1}{\varepsilon^2}+2-2\varepsilon\\ -\frac{1}{\varepsilon^2}+2-2\varepsilon & \frac{1}{\varepsilon}-4\varepsilon\end{bmatrix}$$

and

$$M_1 = Y_1^* Y_1 = \begin{bmatrix} \frac{1}{\varepsilon^2} & -\frac{1}{\varepsilon} \\ -\frac{1}{\varepsilon} & 2 \end{bmatrix}.$$

Solving this reduced problem yields $\theta_1 \simeq 1/\varepsilon$ and $\theta_2 \simeq -\varepsilon$ with

$$Y_1 \simeq \begin{bmatrix} 1 & -\varepsilon^2 \\ \varepsilon^2 & 1 \\ 0 & 0 \end{bmatrix}.$$

Back transforming the θ's yields $\tau_1 = \varepsilon$ and $\tau_2 = -1/\varepsilon$, only one of which is near zero. The first column of Y_1 is still very close to the eigenvector of A. The addition of e_3 to the initial subspace has not concealed this vector.

5. The Lanczos algorithm. In Parlett [1980, p. 323] it is shown that the Lanczos algorithm can be run directly on the pencil (A, M) provided M is positive definite and that certain systems of equations with M as the coefficient matrix can be solved. The form of the equations is

$$Mu_{j+1} = Aq_j - Mq_j\alpha_j + Mq_{j-1}\beta_j. \tag{5.1}$$

If the pencil $(M(A-\sigma M)^{-1}M, M)$ is used, then every term in (5.1) has a leading factor M and the vector u_{j+1} can be found by canceling the M's. That is,

$$u_{j+1} = (A-\sigma M)^{-1}Mq_j - q_j\alpha_j + q_{j-1}\beta_j, \tag{5.2}$$

and the need to solve equations in M has disappeared. We give a listing of the algorithm.

```
THE LANCZOS ALGORITHM ON $(M(A-\sigma M)^{-1}M, M)$.
Pick $r_1 \neq 0$, compute $\beta_1 = \sqrt{r_1^* M r_1}$, and set $q_0 \equiv 0$.
For $j = 1, 2, \cdots$ do 1 to 5
  1. $q_j = r_j/\beta_j$
  2. $u_j = (A-\sigma M)^{-1}Mq_j - q_{j-1}\beta_j$
  3. $\alpha_j = q_j^* M u_j$
  4. $r_{j+1} = u_j - q_j\alpha_j$
  5. $\beta_{j+1} = \sqrt{r_{j+1}^* M r_{j+1}}$
     if $\beta_{j+1} = 0$ Stop.
```

The above algorithm is not new; it has appeared in at least two technical reports (Newman and Flanagan [1976] and van Kats and van der Vorst [1977]) and probably others, although it seems not to have appeared in the open literature. Newman and Flanagan derive the above algorithm and then go on to describe an implementation which in all other respects is seriously flawed. van Kats and van der Vorst give a slightly more general derivation which allows the A matrix to be skew-symmetric but make no mention of the possibility of using origin shifts.

The above algorithm can also be derived directly from the standard Lanczos algorithm simply by observing that the operator $(A-\sigma M)^{-1}M$ is selfadjoint with respect to the M-norm. Thus the only required change is that the inner products needed in the algorithm must be computed as M-inner products.

All the Lanczos theory goes through in this context. See Parlett [1980] for a general discussion and Ericsson and Ruhe [1980] for the modifications needed in the context of inverted operators. In particular, no factorization of M is needed in computing the residual norms and it is straightforward to incorporate Ericsson's perturbation of the Ritz vectors to improve the accuracy of the computed eigenvectors with respect to the original pencil (A, M).

6. Semidefinite *M*. In structural vibration problems the M matrix is often diagonal and only positive semidefinite. At first glance this would appear to invalidate the use of the Lanczos algorithm since it was derived by "cancelling" M's or by invoking an M-norm.

M now represents only a seminorm which could lead to a $\beta_{j+1} = 0$ in step 5 of the algorithm even when $r_{j+1} \neq 0$. Fortunately a zero β indicates that all the computed eigenvalues are exact, and so a zero β is something to be welcomed rather than feared. In essence, the algorithm behaves as if the starting vector has been purged of any component in the direction of the infinite eigenvalues, and thus it is restricted to the finite subspace and can compute only the finite eigenvalues.

Viewing the process with respect to the pencil $(M(A - \sigma M)^{-1}M, M)$ leads to a similar conclusion. Each infinite eigenvalue has become a continuous spectrum. This in turn could lead to a reduced problem with a continuous spectrum which is known to cause numerical problems. However, the Lanczos algorithm generates a reduced problem with the identity as the reduced mass matrix. The only effect of the continuous spectrum is the eventual appearance of $\beta_{j+1} = 0$ when $r_{j+1} \neq 0$. However, as noted above, this occurs precisely when every finite eigenvalue represented in the starting vector r_1 has been computed exactly.

Thus the above algorithm is only effective for computing the finite eigenpairs. If the infinite eigenvectors are desired, it is necessary to compute the null vectors of the pencil (M, I).

The final theoretical problem is the interpretation of the inertia of $A - \sigma M = LDL^*$, since infinite eigenvalues can be either "positive" or "negative" depending on the matrix A. However, the difference in the inertia counts at two different values of σ still gives the number of eigenvalues between them; which is the important quantity.

7. Numerical example. The following numerical example was chosen to illustrate the above version of the Lanczos algorithm. The particular implementation of the code uses block Lanczos with no reorthogonalization and will be documented in a future paper.

The A matrix is tridiagonal with

$$\begin{aligned} a_{ii} &= (-1)^{i+1}2, && i = 1, 2, \cdots, 100, \\ a_{i,i+1} &= 1, && i = 1, 2, \cdots, 99, \\ a_{i,i-1} &= 1 && i = 2, 3, \cdots, 100, \end{aligned}$$

while the M matrix is diagonal with

$$m_{ii} = \frac{1}{i}, \qquad i \not\equiv 1 \bmod 3, \qquad m_{ii} = 0, \qquad i \equiv 1 \bmod 3.$$

There are 34 infinite eigenvalues of (A, M), 17 of which are "positive" and 17 of which are "negative". The other 66 eigenvalues are given in Table 1. The code computed all the eigenvalues of (A, M) in the interval $[-100, 100]$ to four figures of accuracy. A block size of 4 was chosen by the code.

TABLE 1
Finite eigenvalues of (A, M).

−308.255	7.40311
−289.244	14.1328
−270.185	21.4065
−251.127	32.4894
−232.070	34.9585
−214.675	48.3409
−213.014	51.3299
−201.392	61.7082
−193.960	70.2620
−188.102	75.0432
−174.909	88.3626
−174.810	89.2522
−161.517	101.673
−155.861	108.268
−148.223	114.978
−136.818	127.299
−134.927	128.278
−121.628	141.575
−117.782	146.339
−108.326	154.870
−98.7579	165.386
−95.0189	168.164
−81.7046	181.456
−79.7525	184.434
−68.3782	194.747
−60.7847	203.487
−55.0308	208.097
−41.9884	222.541
−41.5448	241.598
−28.2057	260.656
−23.2134	266.611
−14.5097	279.715
−5.46899	298.834

The code is designed to use a sequence of shifts to compute the eigenvalues but the shifting strategy will not be discussed here. A total of six shifts was used, the first two of which were at 100 and −100, to determine the number of desired eigenvalues. The number of eigenvalues accepted for each shift are given in Table 2. Every computed eigenvalue was accurate to at least 4 figures.

TABLE 2
History of Lanczos run.

Shift	Number of negative pivots	Lanczos steps	Eigenvalues accepted
100	62		
−100	37	11	5
−48.9	44	10	8
22.07	53	4	5
58.35	57	4	5
89.34	62	2	2

8. Conclusions. It has been shown that significant advantages can be obtained if an inverted operator is used for the Rayleigh–Ritz step of subspace iteration. Furthermore, the Lanczos algorithm presented here can be used whenever subspace iteration is applicable and the same advantages are obtained. Since the Lanczos algorithm is much more effective than subspace iteration (see Nour-Omid, Parlett and Taylor [to appear]) it is always to be preferred. The algorithm given here is also preferable to that based on the operator $L^*(A-\sigma M)^{-1}L$, since no factorization of M is required and no back transformation of the computed eigenvectors is needed. This is especially true if M is semidefinite, since the factorization of M may involve numerical rank determination and the back transformation of the eigenvectors is more complicated.

Acknowledgment. The author thanks B. N. Parlett for pointing out the derivation of the Lanczos algorithm based on viewing $(A-\sigma M)^{-1}M$ as selfadjoint in the M-norm.

REFERENCES

1. K.-J. Bathe and E. Wilson, *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1976.
2. T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method in the numerical solution of large, sparse, generalized, symmetric eigenvalue problems*, Math. Comp., to appear.
3. M. Newman and P. F. Flanagan, *Eigenvalue extraction in NASTRAN by the tridiagonal reduction (FEER) method—real eigenvalue analysis*, NASA Contractor Report CR-2731, NASA, Washington DC, August 1976.
4. B. Nour-Omid, B. N. Parlett and R. L. Taylor, *Lanczos versus subspace iteration for solution of eigenvalue problems*, to appear.
5. B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
6. R. L. Taylor and J. M. Rollins, *Modified subspace computation for eigenvalues and eigenvectors*, Consulting Report No. 78-1, prepared for PMB Systems Engineering, Inc., San Francisco, 1978.
7. J. M. van Kats and H. A. van der Vorst, *Automatic monitoring of Lanczos schemes for symmetric or skew-symmetric generalized eigenvalue problems*, ACCU Report TR-7, Academic Computing Center Utrecht, Budapestlaan 6, De Vithof-Utrecht, the Netherlands, November 1977.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 1, March 1982

GLIMM'S METHOD FOR GAS DYNAMICS*

PHILLIP COLELLA†

Abstract. We investigate Glimm's method, a method for constructing approximate solutions to systems of hyperbolic conservation laws in one space variable by sampling explicit wave solutions. It is extended to several space variables by operator splitting. We consider two problems. 1) We propose a highly accurate form of the sampling procedure, in one space variable, based on the van der Corput sampling sequence. We test the improved sampling procedure numerically in the case of inviscid compressible flow in one space dimension and find that it gives high resolution results both in the smooth parts of the solution, as well as at discontinuities. 2) We investigate the operator splitting procedure by means of which the multidimensional method is constructed. An $O(1)$ error stemming from the use of this procedure near shocks oblique to the spatial grid is analyzed numerically in the case of the equations for inviscid compressible flow in two space dimensions. We present a hybrid method which eliminates this error, consisting of Glimm's method, used in continuous parts of the flow, and the nonlinear Godunov method, used in regions where large pressure jumps are generated. The resulting method is seen to be a substantial improvement over either of the component methods for multidimensional calculations.

Key words. random choice method, gas dynamics, Glimm's method

1. Introduction.

The problem which motivates this study is the numerical calculation of time-dependent, discontinuous solutions to compressible fluid flow problems in one or more space variables. There are three criteria which such approximate solutions must simultaneously satisfy.

1) The approximate solution must be reasonably accurate in regions where the flow is smooth. Continuous waves should move at the correct speed, have the correct shape, steepen or spread at the correct rate.

2) Discontinuities which are transported along characteristics should be modeled in the approximate solution by sharp jumps which are transported at the correct speed. Examples of such discontinuities are: contact discontinuities (across which the density and temperature have jump discontinuities while the pressure and velocity remain continuous); the interface between two different materials, or between two different thermodynamic phases of the same material; lines or surfaces across which the solution is continuous, but some derivative of the solution is not.

3) Nonlinear discontinuities should be computed stably and accurately. Such discontinuities occur, for example, when there is mass transported across the discontinuity, as in the case of shock fronts in an ideal gas.

The main method used for computing such solutions has been to solve a set of finite difference equations which approximate the differential equations of motion. However, it is difficult to construct difference methods which satisfy all three of the above criteria simultaneously. For example, it is well known that a high order difference method may generate oscillations behind a shock. A first-order method will generally treat the same shock correctly, but numerical diffusion will cause it to give low-resolution results in continuous parts of the flow.

We will be examining an alternative approach to computing discontinuous flows, known variously as Glimm's method, the random choice method, or the piecewise sampling method. This method was first used by Glimm [10] as part of a constructive

* Received by the editors September 3, 1980, and in final form April 15, 1981.

† Lawrence Berkeley Laboratory, University of California, Berkeley, California 94720. This work was performed under the auspices of the U.S. Department of Energy at the Lawrence Livermore National Laboratory and the Lawrence Berkeley Laboratory, and supported in part by the Director, Office of Basic Energy Sciences, Engineering, Mathematical, and Geosciences Division, under contract W-7405-ENG-48. Part of this work is contained in the author's Ph.D. dissertation, University of Calefornia, Berkeley.

existence proof of existence of solutions to systems of nonlinear hyperbolic conservation laws. It was developed by Chorin [3], [4] into an effective numerical method in the case of gas dynamics. In the first reference Chorin also introduced a multidimensional version of the scheme; in the second, he applied the method of reacting gas flow in one space variable. Since that time, the method had been used to compute compressible flow in cylindrical or spherical geometry (Sod [20], [22]), and in applications to some problems in petroleum engineering (Concus and Proskurowski [7], Albright, Concus and Proskurowski [1], Glimm, Marchesin, and McBryan [10], Glimm, Marchesin, Isaacson, and McBryan [11]).

Although one computes solutions on a grid with Glimm's method, it is not a difference method. Rather than computing a weighted sum to arrive at the value of the solution at a grid point, one samples values from an explicit wave solution. Thus, the method has built into it an approximate form of wave transport and interaction, without the smoothing of such information inherent in averaging. The introduction of such a sampling technique as a numerical method is quite recent, compared to the length of time difference methods have been in use, and has not been subject to the extensive scrutiny and application from which the latter has benefited. One of the purposes of this study is to indicate some of the features of Glimm's method which might make developing it worth the effort, as well as a few of the directions in which the development might go.

We consider in this study two fundamental problems.

1) We introduce a more accurate form of the sampling procedure for the one-dimensional method than that used in [3], based on the van der Corput sampling sequence. We compare the performance of the van der Corput sampling and the previously used random sampling schemes.

2) We investigate the operator splitting procedure by which Chorin constructs a multidimensional scheme from the one-dimensional method. A source of error stemming from this procedure, not noticed in [3], is analyzed here and a method for eliminating it is proposed and tested.

In one dimension Glimm's method, with the appropriate sampling, is seen to be superior to any difference method in meeting the three criteria given above. The final method obtained for multidimensional calculations, although it does not share the special properties of the one-dimensional method, has a number of interesting features, and is worthy of further investigation for its own sake.

This paper is divided into three sections. In § 2 we discuss Glimm's method as applied to gas dynamics in one space variable. We define the van der Corput sampling sequences, and compare the van der Corput and random sampling strategies. In § 3 we describe the operator splitting technique. In § 4 we compare Glimm's method to some difference methods and give some conclusions, and suggestions for future work.

2. Gas dynamics in one space variable. We want to construct approximate solutions to the initial value problem for Euler's equations for the motion of a one-dimensional, compressible, inviscid gas with a polytropic equation of state:

$$\frac{\partial U}{\partial t}+\frac{\partial F(U)}{\partial x}=0,$$

$$U(x,t)=U:R\times[0,T]\to R^3,\qquad U(x,0)=u(x)\quad\text{given,}\tag{2.1}$$

$$U=\begin{pmatrix}\rho\\ m\\ E\end{pmatrix},\qquad F(U)=\begin{pmatrix}\rho\\ m^2/\rho+p\\ (E+p)m/\rho\end{pmatrix},$$

where ρ is the density, m is the momentum, and E is the total energy per unit volume of the gas. We can express in terms of these variables the more familiar quantities v the velocity, and ε, the internal energy of the gas:

$$v=\frac{m}{\rho}, \qquad \varepsilon=\frac{E}{\rho}-\frac{u^2}{2}.$$

The pressure p which appears in the equation is a function of $\rho, \varepsilon: p=(\gamma-1)\rho\varepsilon=A\rho^\gamma$, where the constant $\gamma>1$ is the ratio of specific heats for the gas. Another quantity of interest is the entropy, S, defined (up to an additive and a multiplicative constant) $S=\log(p\rho^{-\gamma})=\log A$.

The system (2.1) is a first-order, hyperbolic system of conservation laws; i.e., the 3×3 matrix $A(U)$, the Jacobian of F, has three real eigenvalues,

$$\lambda_1(U)=u-c, \quad \lambda_2(U)=u, \quad \lambda_3(U)=u+c,$$

where $c=\sqrt{\gamma p/\rho}$ is the adiabatic sound speed. λ_i, $i=1,2,3$ are the characteristic velocities associated with the three modes of wave propagation for (2.1).

Since we are dealing with piecewise smooth solutions, we interpret (2.1) in the sense of distributions. That is, if $U(x, t)$ is discontinuous along a piecewise smooth curve $(l(t), t)$, then $dl/dt=s(t)$ must satisfy

$$(2.2) \qquad s(t)(U_L(l(t),t)-U_R(l(t),t))=F(U_L(l(t),t))-F(U_R(l(t),t)),$$

where

$$U_{L,R}(l(t),t)=\lim_{\varepsilon\uparrow 0,\varepsilon\downarrow 0} U(l(t)+\varepsilon,t).$$

The discontinuity must also satisfy the entropy conditions (Courant and Friedrichs [8]). Discontinuities calculated using any of the numerical methods discussed in this paper satisfy the entropy conditions if they satisfy (2.2).

The simplest initial value problem for which discontinuities appear is one for which the initial data is constant on either side of the origin, where it has a jump discontinuity:

$$\phi(x)=\begin{cases} U_L, & x<0, \\ U_R, & x>0, \end{cases} \qquad U_L, U_R\in R^3,$$

where we denote

$$U_{L,R}=\begin{pmatrix}\rho_{L,R}\\ m_{L,R}\\ E_{L,R}\end{pmatrix}=\begin{pmatrix}\rho_{L,R}\\ \rho_{L,R}u_{L,R}\\ p_{L,R}/(\gamma-1)+\frac{1}{2}\rho_{L,R}u^2_{L,R}\end{pmatrix}.$$

This problem is known as the Riemann problem; its solution is a fundamental component of Glimm's method. The special case of the Riemann problem in which $U_L=U_R=0$ is often referred to as the shock tube problem. The solution of the Riemann problem is discussed extensively in Chorin [3], Courant and Friedrichs [8], Godunov [12] and Sod [21] and van Leer. [23] In [3] and [21] detailed instructions for constructing solutions numerically are given; thus we will describe only qualitatively the structure of the solution.

Two general properties of the solution to a Riemann problem are that it is self-similar, i.e., $U(x,t)=h(x/t)$ for some piecewise continuous $h:R\to R^3$, and that it has the following additivity property: if $U(\xi, t\xi) = U_M \in R^3$ for some R, then the function

$$U_1(x,t) = \begin{cases} U(x,t), & \dfrac{x}{t} < \xi, \\ U_M, & \dfrac{x}{t} > \xi \end{cases}$$

is the solution to the Riemann problem with left and right states U_L, U_M. Similarly, the function

$$U_2(x,t) = \begin{cases} U_M, & \dfrac{x}{t} < \xi, \\ U(x,t), & \dfrac{x}{t} > \xi \end{cases}$$

is the solution to the Riemann problem with left and right states U_M, U_R. Geometrically, this says that the solutions U_1, U_2 fit together to form U.

One can divide the (x, t)-plane into four regions, I, II, III, IV, where $U(x, t)$ is constant (Fig. 1). These four regions are connected by three waves, each associated with one of the characteristic speeds. These are: a backward facing sonic wave (associated with $u - c = \lambda_1(U)$, between $l_{1,b}$ and $l_{2,b}$; a contact discontinuity (associated with $u = \lambda_2(U)$, occurring across the line l_s; and a forward facing sonic wave (associated with $u + c = \lambda_3(U)$ between $l_{1,f}$ and $l_{2,f}$. The pressure and velocity are continuous across the line l_s so they are equal to some fixed values p^*, u^* in II and III. Only the density changes across $l_s(t) = u^*t$, from ρ_L^* to ρ_R^*.

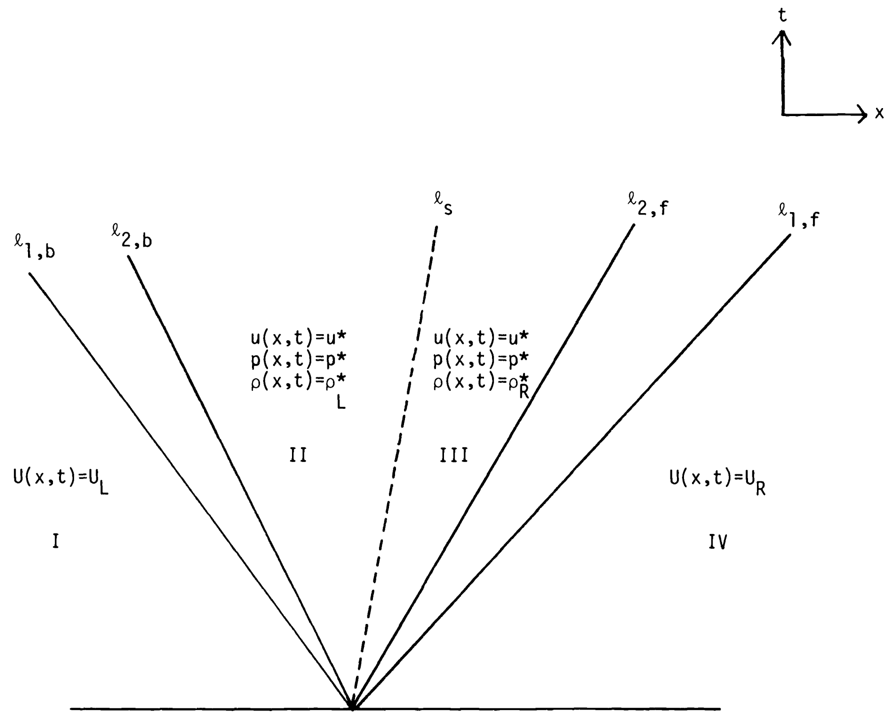


FIG. 1. *The Riemann problem for gas dynamics.*

As is discussed in [8], the hydrodynamic waves are uniquely determined by knowing the state of the gas on one side of the wave, and only the pressure on the other. For the backward facing wave, for example, there are two possibilities. If $p^* > p_L$ then $u^* < u_L$, $\rho_L^* > \rho_L$, $l_{1,b} = l_{2,b}$ and the wave is a shock associated with the characteristic velocity $u - c$. If $p^* < p_L$, then we have a backward facing centered rarefaction wave: $l_{1,b} < l_{2,b}$, $p(x, t)$ and $u(x, t)$ are continuous strictly monotone decreasing functions of x/t, and $u(x, t)$ a continuous strictly monotone increasing function of x/t, for (x, t) between $l_{1,b}$ and $l_{2,b}$. The description of the forward facing wave is the same, replacing U_L by U_R, u by $-u$, and $u + c$ by $u - c$.

In Fig. 2 we show the solution at a fixed time to the shock tube problem

$$\begin{aligned} p_L &= 1.0, & p_R &= 0.1, \\ \rho_L &= 1.0, & \rho_R &= 0.125, \\ u_L &= u_R = 0, & \gamma &= 1.4. \end{aligned} \tag{2.3}$$

The waves which occur are a backward facing rarefaction wave (A), a forward facing shock (B), and a contact discontinuity (C).

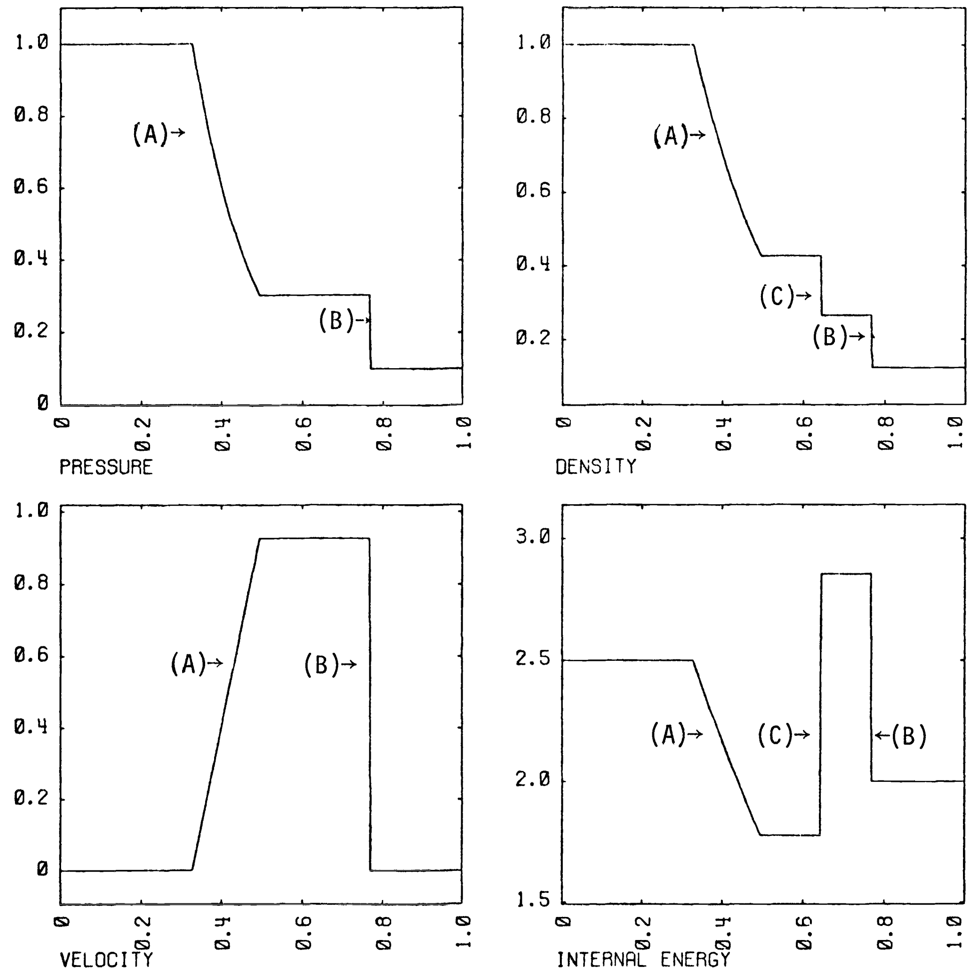


FIG. 2. *Solution at fixed time to shock tube problem* (2.3).

We can now describe Glimm's method for solving approximately the initial value problem (1.1). Let Δx be a spatial increment, Δt a time increment. We assume that, at time $n\Delta t$, the approximate solution is constant on intervals of length Δx:

$$(2.4)\quad U^{(\Delta x)}(x, n\Delta t) = U_j^n \in R^3 (j-\tfrac{1}{2})\Delta x < x < (j+\tfrac{1}{2})\Delta x, \qquad j = 0, \pm 1, \pm 2, \cdots.$$

We wish to compute an approximate solution which at time $n\Delta t$ has the same property:

$$U^{(\Delta x)}(x, (n+1)\Delta t) = U_j^{n+1}.$$

The procedure is given as follows:

(1) Define $U_n^e(x, t) n\Delta t < t < (n+1)\Delta t$ to be the exact solution to the initial value problem for (1.1) with initial data given by (2.4). The initial data consist of intervals where the solution is constant, separated by jump discontinuities; i.e., we have a succession of Riemann problems. If Δt is sufficiently small, then by finite propagation speed the waves from adjacent discontinuities do not intersect each other and the solutions to the adjacent Riemann problems fit together to given U_n^e (Fig. 3). A condition on Δt which guarantees that the waves do not intersect is

$$\frac{\Delta t}{\Delta x} = \lambda < \frac{1}{2} \sup_{\substack{x \in R \\ (n+1)\Delta t > t > n\Delta t}} |u_n^e(x, t)| + c_n^e(x, t).$$

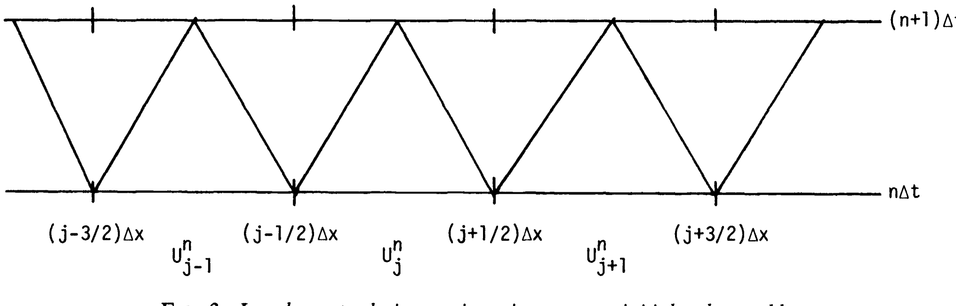


FIG. 3. *Local exact solution to piecewise constant initial-value problem.*

When doing calculations, one usually uses the more easily verified

$$(2.5)\qquad \frac{\Delta t}{\Delta x} = \lambda < \sigma \sup_{x \in R} |u^{(\Delta x)}(x, n\Delta t)| + c^{(\Delta x)}(x, n\Delta t) = \sigma \sup_j (|u_j^n| + c_j^n),$$

where σ is a constant, $0 < \sigma < \frac{1}{2}$.

(2) Choose $a^{n+1} \in [0, 1)$ and take

$$U_j^{n+1} = U_n^e((j - \tfrac{1}{2} + a^{n+1})\,\Delta x, (n+1)\,\Delta t).$$

See Fig. 4.

Thus we obtain a solution at time $(n+1)$ which depends on a sequence $\vec{a} = a^1$, $a^2, \cdots$; much of the remainder of this section will be devoted to determining the best choice for the sequence $\vec{a}$.

At first glance, this method might look complicated, but in fact it requires the evaluation of the solution to a Riemann problem once per zone per timestep. Let

$$h_{j-\frac{1}{2},n}\left(\frac{n-(j-\frac{1}{2})\Delta x}{t - n\Delta t}\right)$$

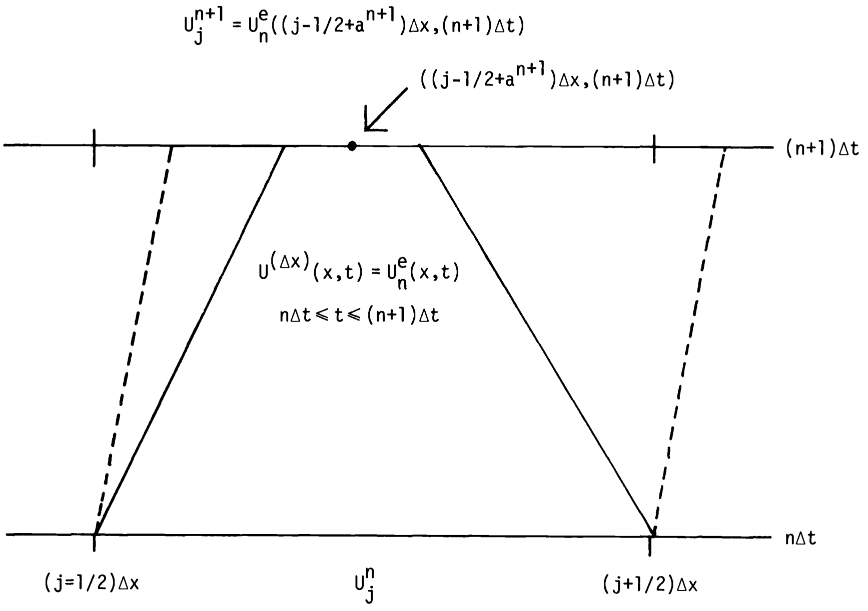

U_j^{n+1} = U_n^e((j-1/2+a^{n+1})Δx,(n+1)Δt)
((j-1/2+a^{n+1})Δx,(n+1)Δt)
(n+1)Δt
U^{(Δx)}(x,t) = U_n^e(x,t)
nΔt≤t≤(n+1)Δt
nΔt
(j=1/2)Δx
U_j^n
(j+1/2)Δx

FIG. 4. *Sampling the local exact solution.*

be the solution to the Riemann problem with left and right states U_{j-1}^n, U_j^n and let

$$\bar{U}_{j-\frac{1}{2}}^n = h_{j-\frac{1}{2},n}\left(\left(a^{n+1}-\frac{1}{2}\right)\frac{\Delta x}{\Delta t}\right).$$

Then

$$U_j^{n+1} = \begin{cases} \bar{U}_{j-\frac{1}{2}}^n, & a^{n+1} > \frac{1}{2}, \\ \bar{U}_{j+\frac{1}{2}}^n, & a^{n+1} < \frac{1}{2}. \end{cases}$$

The procedure given here is slightly different from that used previously, in that the mesh is fixed, rather than shifting by $\Delta x/2$ every timestep. Sampling back to a fixed grid shows that the relation to Godunov's method is immediate: in Godunov's method, U_j^{n+1} is taken to be

$$\frac{1}{\Delta x}\int_{(j-\frac{1}{2})\Delta x}^{(j+\frac{1}{2})\Delta x} U_n^e(x, \Delta t)\, dx;$$

in Glimm's method, one chooses a representative point value of the local exact solution.

The mechanism by which Glimm's method models wave propagation in a gas is most easily demonstrated by the following example. Let U_L, U_R be the left and right states of a Riemann problem whose solution consists of a single discontinuity propagating at speed $s > 0$. The exact solution for this problem is

$$U(x, t) = \begin{cases} U_L, & x < sT, \\ U_R, & x > sT. \end{cases}$$

We will solve this initial value problem using Glimm's method. First, it is obvious that, for any time step n there is an $l(n) = j_0 - \frac{1}{2}$, j_0 an integer, such that

$$U_j^n = \begin{cases} U_L, & j < l(n), \\ U_R, & j > l(n). \end{cases}$$

$l(n)$ is the location of the shock in the approximate solution, and satisfies

$$l(n+1)=\begin{cases} l(n) & \text{if } a^{n+1}>\lambda s, \\ l(n+1) & \text{if } a^{n+1}<\lambda s, \end{cases} \qquad l(0)=\tfrac{1}{2},$$

so that

$$l(n)=l(0)+N\{j=1,\cdots,n;\, a^j\in[0,\lambda s)\},$$

where $N\{j=n_1+1,\cdots,n_2,\, a^j\in I\}$ denotes the number of j, $n_1<j\leqq n_2$ such that a^j is contained in I. We say that a sequence $\vec{a}$ is equidistributed if the proportion of times that a^j is contained in I is asympotically equal to $|I|$, the length of I; i.e., if we define

$$\frac{1}{n_2-n_1}N\{j=n_1+1,\cdots,n_2;\, a^j\in I\}-|I|\equiv\delta(\vec{a},n_1,n_2,I),$$

then $\vec{a}$ is equidistributed if $\lim_{n_2\to\infty}$, $\delta(\vec{a},n_1,n_2,I)=0$ for each fixed n_1, I. Given this notation we write

$$\begin{aligned} l(n)\Delta x &= l(0)\Delta x+\Delta x\lambda sn+\delta(\vec{a},0,n,0,\lambda s))n\,\Delta x \\ &= l(0)\Delta x+n\Delta t\left(1+\frac{1}{\lambda}\delta(\vec{a},0,n[(0,\lambda s))\right). \end{aligned}$$

If $\vec{a}$ is equidistributed, then $l(n)\to sT$ in the limit $n\to\infty$, $|n\Delta t-T|<\Delta t$, $\Delta t/\Delta x=\lambda>0$.

Thus the shock in the approximate solution at each time step either moves by Δx or does not move at all. Over many time steps, the cumulative displacement is close to that of the exact solution, the leading term in the error being proportional to $\delta(\vec{a},0,n,[0,\lambda s))$. In general, a piecewise continuous flow will be represented by $O(1/\Delta x)$ waves of strength $O(\Delta x)$, all having differing speeds, as well as an $O(1)$ number of discontinuities of strength $O(1)$. Furthermore, the speeds and strengths of the waves will be changing in a piecewise continuous fashion as a function of time. In order to model such a flow correctly by the above mechanism, one needs to choose $\vec{a}$ such that δ is as small as possible, uniformly in I, n_1 for n_2-n_1 large relative to 1, but $(n_2-n_1)\Delta t$ small relative to the characteristic times in which the wave speeds change. The sampling procedure given below seems to be optimal from the point of view of these requirements.

The simplest form of this sampling sequence is due to van der Corput (see [14]). Let

$$n=\sum_{k=0}^{m} i_k 2^k, \qquad i_k=0,1$$

be the binary expansion of $n=1,2,\cdots$. Then

$$a^n=\sum_{k=0}^{m} i_k 2^{-(k+1)}.$$

The easiest way to see how the sequence is constructed is to write down the first few elements in it:

$1=1_2$,	$a^1=.5$	$=.1_2$,
$2=10_2$,	$a^2=.25$	$=.01_2$,
$3=11_2$,	$a^3=.75$	$=.11_2$,
$4=100_2$,	$a^4=.125$	$=.001_2$,
$5=101_2$,	$a^5=.625$	$=.101_2$,
$6=110_2$,	$a^6=.375$	$=.011_2$,
$7=111_2$,	$a^7=.875$	$=.111_2$,
$8=1000_2$,	$a^8=.0625$	$=.0001_2$.

So

$$a^i \leqq .5 \quad \text{if } i \text{ is} \begin{cases} \text{even,} \\ \text{odd,} \end{cases} \qquad \frac{k}{4} \leqq a^i < \frac{k+1}{4} \quad \text{if } i \equiv j(k) \bmod 4,$$

$k = 0, 1, 2, 3$, where $j(0) = 0$, $j(1) = 2$, $j(2) = 1$, $j(3) = 3$. In general, if one divides the unit interval into the subintervals $(r2^{-s}, (r+1)2^{-s}) r = 0, \cdots, 2^s - 1$, then for each r there is exactly one q for which $q_0 < q < q_0 + 2^s$ such that $a^q \in [r2^{-s}, (r+1)2^{-s})$.

We will have need of a variant of this procedure for use in multidimensional problems. Let k_1, $k_2 > 0$ be integers, $k_1 > k_2$ relatively prime. The (k_1, k_2) van der Corput sampling sequence $\vec{a}$ is given by

$$a^n = \sum_{l=0}^{m} q_l \cdot k_1^{-(l+1)},$$

where

$$q_l \equiv k_2 i_l \bmod k_1 \quad \text{and} \quad \sum_{l=0}^{m} i_l k_1^l = n$$

is the base k_1 expansion of n. Thus the binary van der Corput sampling sequence given above is the special case $k_1 = 2$, $k_2 = 1$.

All the van der Corput sampling sequences are equidistributed. In fact $\delta(\vec{a}, n_1, n_2, I) \leqq (C_1 \log k_1(n_2 - n_1) + C_2)/(n_2 - n_1)$, where C_1, C_2 are constants depending on k_1, k_2 but not on n_1, n_2 or I. For the binary van der Corput sequence $C_1 = 3$, $C_2 = 1$. In the example given above, this gives an error bound of $O(\Delta x |\log \Delta x|)$.

In previous computational work for gas dynamics using Glimm's method, random sampling was used; i.e., the values were drawn from a random number generator implemented on the computer, usually with some variance reduction technique, such as stratification for random sampling, for which $\delta = O(1/\sqrt{n})$, giving an error bound in our simple shock example of $O(\sqrt{\Delta x})$.

Lax [17] proposed the use of a nonrandom equidistributed sequence due to Richtmyer and Ostrowski, defined by $a^n \equiv \sqrt{r} \bmod 1$, where r is an integer which is not the square of another integer.

We shall not discuss the Richtmyer–Ostrowski sampling sequence in detail here, save to note that in numerical experiments, and in simple analytical examples, one obtains results using the Richtmyer–Ostrowski sequence similar to those obtained using van der Corput sampling. However, the bound on δ is stronger for the van der Corput sequence than that obtained for the Richtmyer–Ostrowski sequence, as well as being explicitly uniform in I and n_1; uniformity in I and n_1, does not hold explicitly for the Richtmyer–Ostrowski sequence. Also, van der Corput sampling has some special properties which guarantee that certain qualitative features of the continuous part of the solution preserved in the approximate solution, at least for simple waves (see [5], [6]). Finally, van der Corput sampling has several straightforward extensions to two or more dimensions which guarantee good distribution properties in the square, even for finite sample sizes. In contrast, it has been pointed out by Maltz and Hitzl [18] that such an extension of the Richtmyer–Ostrowski sequence can give rise to poor distribution in the square for finite sample sizes.

In an effort to understand the errors introduced by the interaction of the sampling and variations in time in the wave speeds, we consider the following class of test problems. The initial data consist of two discontinuities located at x and x_r, separated by constant states:

$$U(x,0)=\begin{cases} U_l, & x<x_l, \\ U_m, & x_l<x<x_r, \\ U_r, & x>x_r, \end{cases}$$

$$U_{l,m,r}=\begin{pmatrix} \rho_{l,m,r} \\ \rho_{l,m,r}u_{l,m,r} \\ p_{l,m,r}/(\gamma-1)+\frac{1}{2}\rho_{l,m,r}u^2_{l,m,r} \end{pmatrix}.$$

We choose U_l, U_m, U_r such that U_l and U_m can be connected by a forward facing shock and that U_m and U_r can be connected by a forward facing centered rarefaction wave (Fig. 5).

The shock overtakes the rarefaction, the cancellation between them weakening both (Fig. 6, (A)). The nonlinear coupling between the modes produces waves of the other two families in back of the shock and moving to the left, away from the shock.

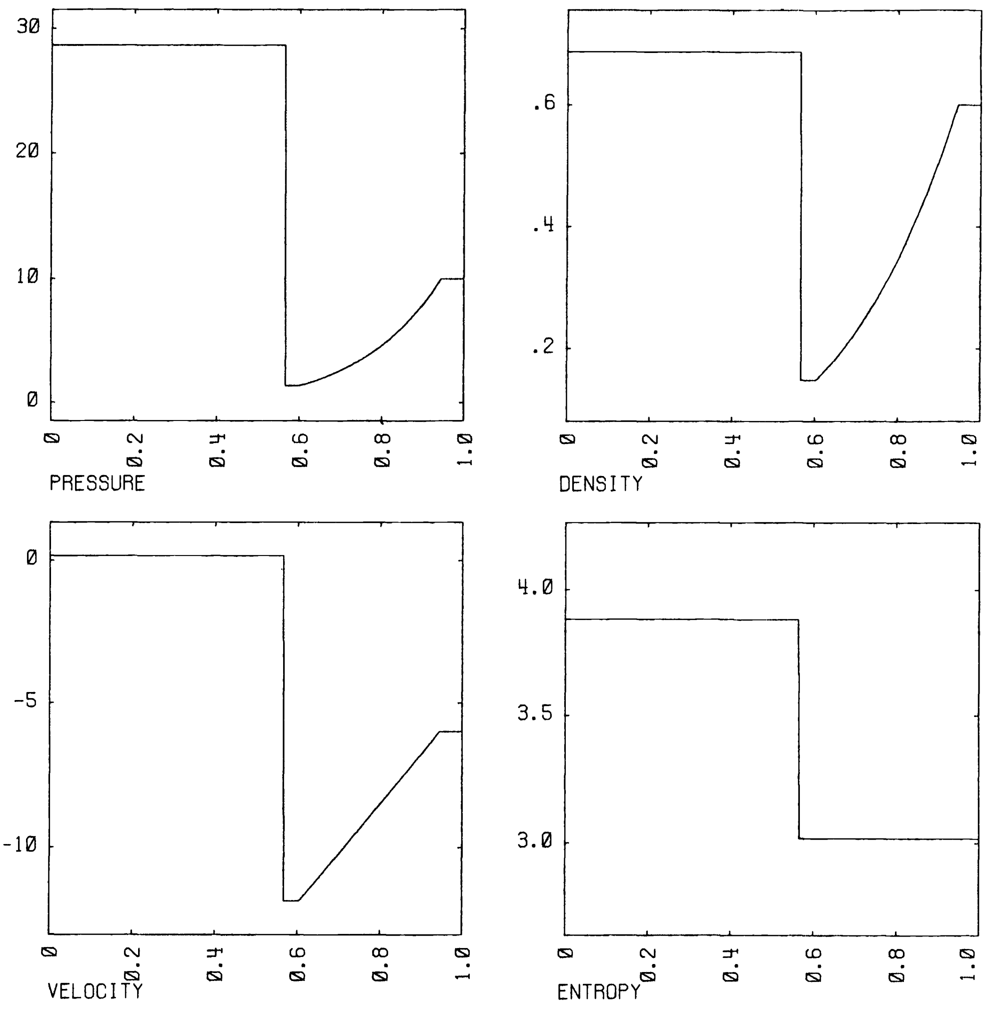


FIG. 5. *Waves initially generated in* 1D *test problem.*

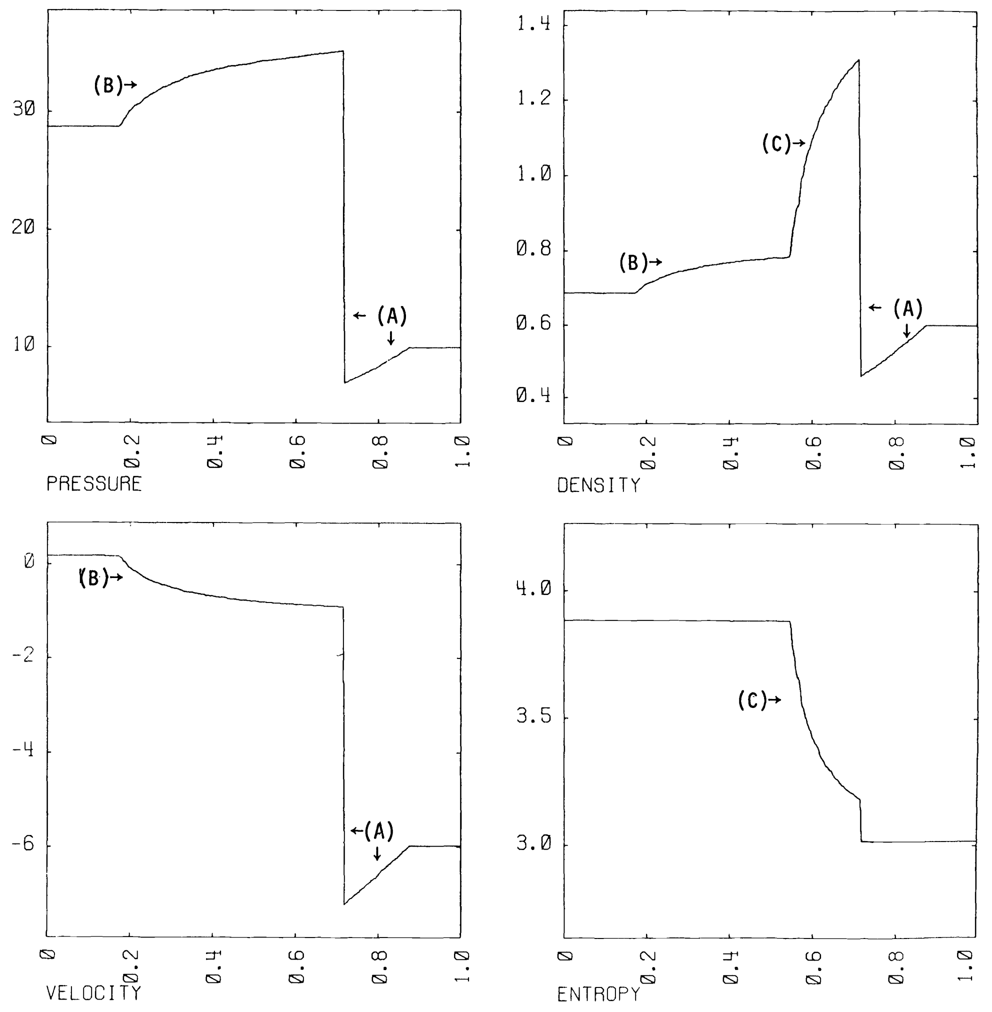


FIG. 6. *Computed solution to* 1D *test problem, van der Corput sampling,* $\Delta x = .0025$.

These are, a backward facing compression wave (Fig. 6, (B)), and a strong entropy/density wave (Fig. 6, (C)) advected passively by the velocity field $u(x, t)$.

In Figs. 7–9 we show the calculation of such a problem using Glimm's method with, respectively, a random sampling sequence, a stratified random sampling, and the binary van der Corput sampling sequence. The initial data are:

$$\begin{aligned} &p_l = 28.68, && p_m = 1.39, && p_r = 10.0, \\ &\rho_l = .6878, && \rho_m = .146, && \rho_r = .6, \\ &u_l = .0181, && u_m = -11.9, && u_r = -5.98, \\ &x_l = .4, && x_r = .9, && \gamma = 1.4. \end{aligned}$$

All calculations were done on the spatial interval [0, 1], with boundary conditions at 0 and 1 obtained by assuming the solutions satisfy $\partial U/\partial x|_{x=0,1} = 0$. The various solutions being compared were computed with $\Delta x = .01$, and are represented graphically by circles for the computed values at mesh points, interpolated by a dotted line.

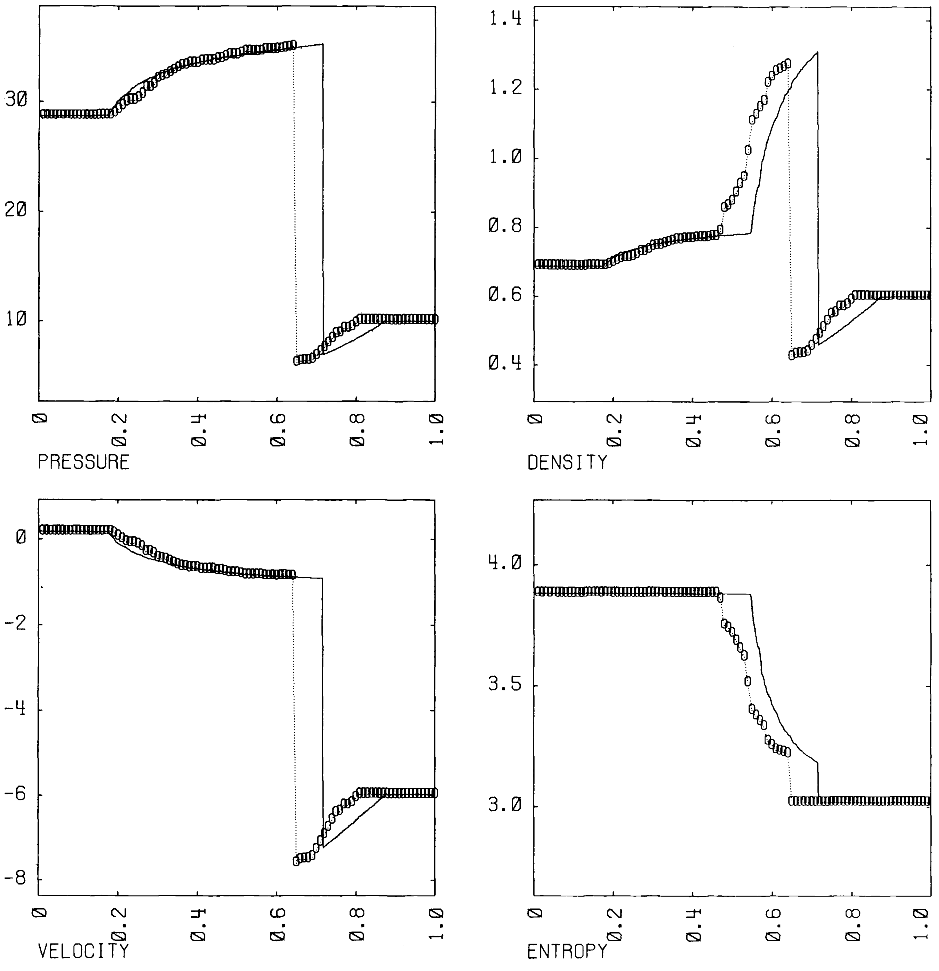


FIG. 7. *Computed solution to* 1D *test problem, random sampling,* $\Delta x = .01$.

Also plotted on each of the graphs, with a solid line, is a solution obtained using Glimm's method, with van der Corput and $\Delta x = .0025$. Having compared the latter solution with a similar one done for $\Delta x = .005$ we found that the two results differed by less than .5%, so that the method has converged for $\Delta x = .0025$. For the purposes of comparing the various $\Delta x = .01$ solutions, we treat the $\Delta x = .0025$ solutions as exact, against which the $\Delta x = .01$ solutions can be compared.

The sampling governs the rate at which the shock and rarefaction interact. If $s^n_{q+1/2}$ is the speed of the shock, located between zones q and $q+1$ at time step n, and $\lambda^n_+ = u^n_{q+1} + c^n_{q+1}$, then the shock will cancel with a piece of the rarefaction wave, and produce more wave of the other two families, at time step $n+1$, if and only if

$$a^{n+1} \in \left[\frac{\Delta t}{\Delta x} \max(\lambda^n_{+,0}), \frac{\Delta t}{\Delta x} s^n_{q+\frac{1}{2}}\right).$$

Thus the loss of gradient information observed in the randomly sampled solution (Fig. 7) is a result of random fluctuations in the rate of interaction between the shock and rarefaction which is producing the wave. The use of stratified random sampling

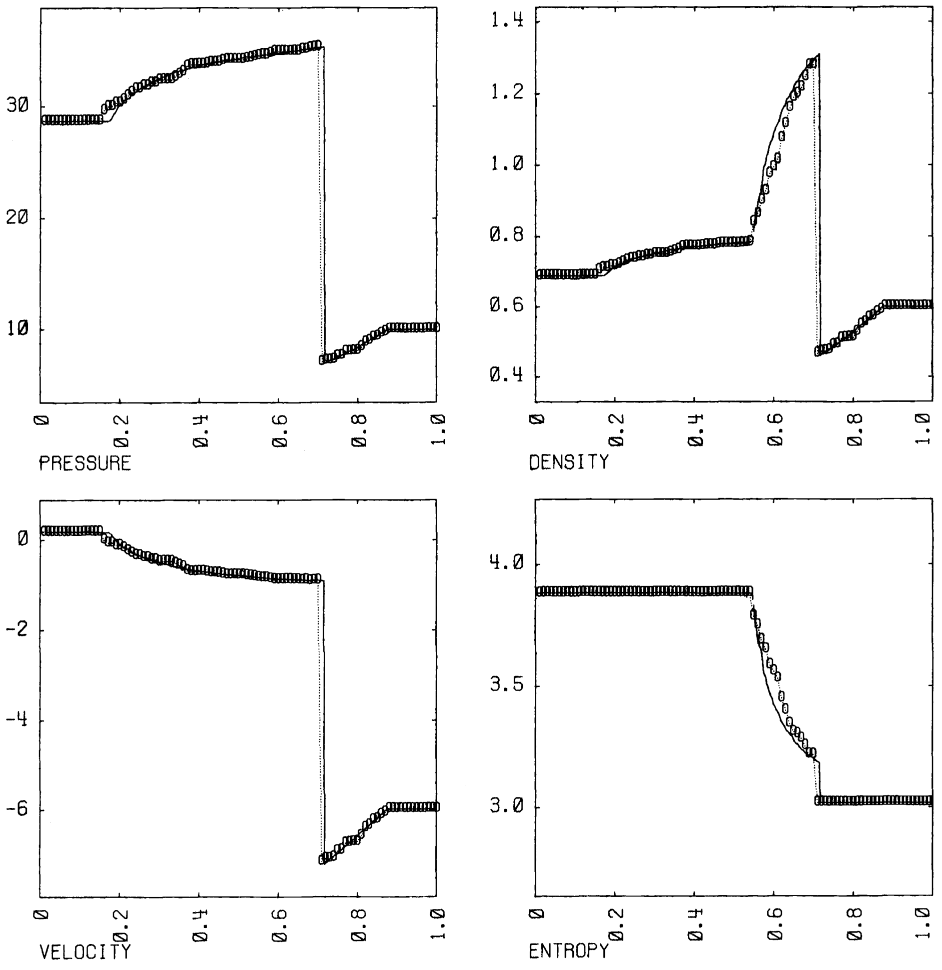


FIG. 8. *Computed solution to* 1D *test problem,* (7, 3) *stratified random sampling,* $\Delta x = .01$.

(Fig. 8) produces smoother profiles than those obtained with the unmodified random sequence, but the shape of the entropy wave is incorrect; in particular, there is a sizable deviation in the density profile, a failure to conserve mass. The profile obtained using van der Corput sampling (Fig. 9) is in much closer agreement with the $\Delta x = .0025$ result, the rate of wave production being modeled much better than in the other two cases. In fact, if one uses van der Corput sampling, one can use a much coarser mesh and still get good results for this problem. In Fig. 10 we present the results obtained on this problem with binary van der Corput sampling, and $\Delta x = \frac{1}{30}$. The absolute locations of the waves, and their locations relative to each other, are correct to within Δx; more important, the size and shape of the waves, which are more sensitive to the cumulative error introduced by the sampling, are in very close agreement with the $\Delta x = .0025$ result. In all the calculations, the shock discontinuity is sharp, as guaranteed by Glimm's method.

3. Operator splitting. In [3], Chorin proposed a method for computing multi-dimensional unsteady compressible flow using Glimm's method by means of operator splitting. We can write the equations of motion for an ideal gas in two space

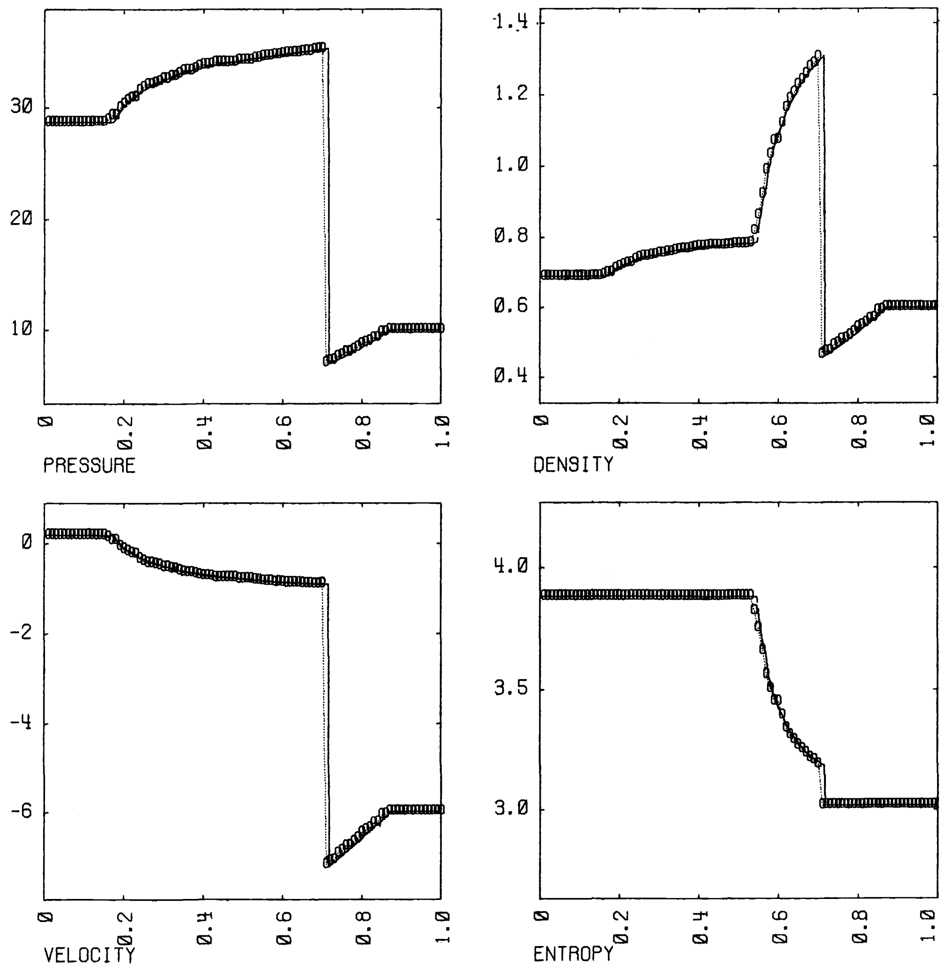


FIG. 9. *Computed solution to test problem, van der Corput sampling,* $\Delta x = .01$.

dimensions as

$$\frac{\partial U}{\partial t}+\frac{\partial}{\partial x}(F(U))+\frac{\partial}{\partial y}(G(U))=0,$$

$$U(x, y, t)=U\colon R^2\times[0, T]\to R^4,$$

$$U(x, y, 0)=\phi(x, y)\phi\colon R^2\to R^4,$$

$$U=\begin{pmatrix}\rho\\ m\\ n\\ E\end{pmatrix},\quad F(U)=\begin{pmatrix}m\\ m^2/\rho+p\\ mn/\rho\\ (m/\rho)(E+p)\end{pmatrix},\quad G(U)=\begin{pmatrix}n\\ mn/\rho\\ n^2/\rho+p\\ (n/\rho)(E+p)\end{pmatrix}.$$

Here ρ is the density, m is the x-component of momentum, n is the y-component of momentum, and E is the total energy. We can express the velocity $\vec{v}$ and the internal energy ε in terms of the above variables: $v_x = m/\rho$ is the x-component of the velocity, $v_y = n/\rho$ is the y-component of the velocity, and $\varepsilon = E/\rho - \frac{1}{2}(v_x^2+v_y^2)$. The pressure p is a function of ρ and ε: $p=(\gamma-1)\rho\varepsilon$, where γ, the ratio of specific

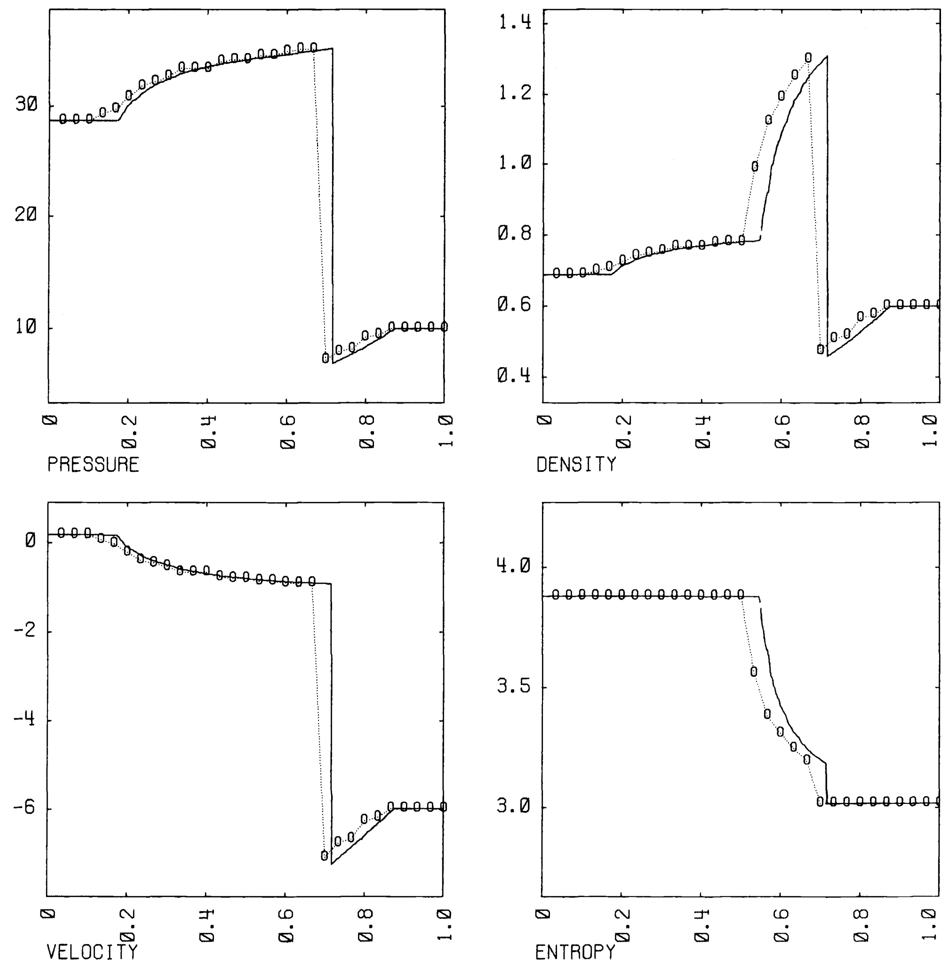


FIG. 10. *Computed solution to test problem, van der Corput sampling,* $\Delta x = \frac{1}{30}$.

heats, is a constant assumed to be greater than 1. Thus, as was the case for one space variable, the value U at a given point is uniquely determined by the values of ρ, p and $\vec{v}$ at that point.

We wish to construct approximate solutions

$$U^{\Delta x, \Delta y}(x, y, n\,\Delta t) = U^n_{i,j} \in R^4,$$
$$(i-\tfrac{1}{2})\,\Delta x < x < (i+\tfrac{1}{2})\,\Delta x, \qquad (j-\tfrac{1}{2})\,\Delta y < y < (j+\tfrac{1}{2})\,\Delta y,$$

where x, y are spatial increments, t is a time increment, and i, j, n are integers, $n \geqq 0$.

Assume we know $U^n_{i,j}$ and want to find $U^{n+1}_{i,j}$; the procedure is as follows.

1) For each j perform one time step of Glimm's method for the equation

$$\frac{\partial V}{\partial t} + \frac{\partial}{\partial x}(F(V)) = 0.$$

taking as initial data $V^0_i = U^n_{i,j}$. Set the results $V^1_i = U^{n+\frac{1}{2}}_{i,j}$ (we denote this procedure by $(L^x_{\Delta t}U^n)_{i,j} = U^{n+\frac{1}{2}}_{i,j}$).

2) For each fixed i perform one time step of Glimm's method for the equations

$$\frac{\partial V}{\partial t}+\frac{\partial}{\partial y}(G(V))=0,$$

taking as initial data $V_j^0 = U_{i,j}^{n+\frac{1}{2}}$, time step t. Set the result $V_j^1 = U_{i,j}^{n+1}$ (we denote this procedure by $(L_{\Delta t}^y U^{n+\frac{1}{2}})_{i,j} = U_{i,j}^{n+1}$).

The solution thus derived at time $(n+1)\,\Delta t$ is interpreted as being the piecewise constant function

$$U^{\Delta x,\Delta y}(x, y, (n+1)\,\Delta t) = U_{i,j}^{n+1},$$
$$(i-\tfrac{1}{2})\,\Delta x < x < (i+\tfrac{1}{2})\,\Delta x, \qquad (j-\tfrac{1}{2})\,\Delta y < y < (j+\tfrac{1}{2})\,\Delta y.$$

A necessary condition on the time step t is that it must satisfy (1.4) for each of the one-dimensional calculations

$$\frac{\Delta t}{\Delta x} < \sigma \max_{i,j}\,(|v_{x,i,j}^n| + c_{i,j}^n),$$
(3.1)
$$\frac{\Delta t}{\Delta y} < \sigma \max_{i,j}\,(|v_{y,i,j}^n| + c_{i,j}^n), \qquad 0<\sigma<\tfrac{1}{2}.$$

The above procedure is formally the same as is done to construct multidimensional difference methods from one-dimensional ones. However, the mechanism by which Glimm's method propagates the solution to the equation in one dimension is rather different from that of different methods, as it requires many time steps for the cumulative effect of the sampling to give the correct wave speeds; therefore the actual justification of the splitting procedure, currently unknown, is likely to be quite different than the usual truncation error analysis for difference methods.

The Riemann problems in question are easily solved, given the solution for one-dimensional gas dynamics. For example, to solve Riemann's problem for

$$\frac{\partial V}{\partial t}+\frac{\partial}{\partial x}(F(V))=0,$$

$$V=\left(\bar{\rho}, \bar{\rho}\bar{v}_x, \bar{\rho}\bar{v}_y, \frac{\bar{p}}{\gamma-1}+(\bar{v}_x^2+\bar{v}_y^2)\frac{\bar{\rho}}{2}\right),$$

take the solution $\rho(x, t)$, $p(x, t)$, $u(x, t)$ in § 2 with

$$\rho_{L,R}=\bar{\rho}_{L,R}, \quad p_{L,R}=\bar{p}_{L,R}, \quad u_L=\bar{v}_{x,L}, \quad u_R=\bar{v}_{y,R},$$
$$\bar{\rho}(x,t)=\rho(x,t), \quad \bar{p}(x,t)=p(x,t), \quad \bar{v}_x(x,t)=u(x,t),$$

and $\bar{v}_y(x, t) = \bar{v}_{y,L}$ if (x, t) is to the left of the contact discontinuity l_s, $\bar{v}_y(x, t) = \bar{v}_{y,R}$ if (x, t) is to the right of the contact discontinuity l_s. Thus in the x-sweep, we have ordinary one dimensional gas dynamics, with the discontinuity in v_y passively advected. To solve the Riemann problem for $\partial V/\partial t + \partial/\partial y\ (G(V)) = 0$, interchange the roles of v_x and v_y.

To test the validity of this procedure, we looked at the simplest two-dimensional test problem possible. We took our computational domain to be the unit square with the computational mesh aligned with the x- and y-axes, and took the initial conditions to be

$$U(x,y)=\begin{cases} U_R, & x<y,\\ U_L, & x>y,\end{cases}$$

$$U_L = \begin{pmatrix} \rho_L \\ \rho_L v_{x,L} \\ \rho_L v_{y,L} \\ p_L/(\gamma-1) + (\rho_L/2)(v_{x,L}^2 + v_{y,L}^2) \end{pmatrix}, \quad U_R = \begin{pmatrix} \rho_R \\ \rho_R v_{x,R} \\ \rho_R v_{y,R} \\ p_R/(\gamma-1) + (\rho_R/2)(v_{x,R}^2 + v_{y,R}^2) \end{pmatrix}.$$

This is the Riemann problem, for which we have an analytic solution. Computationally, it is a two-dimensional problem, since the initial discontinuity is at a 45° angle to the mesh directions.

We denote by v_n the component of the velocity normal to $x = y$, v_t the component parallel to $x = y$:

$$v_n = \frac{v_x - v_y}{\sqrt{2}}, \qquad v_t = \frac{v_x + v_y}{\sqrt{2}},$$

$$v_{n,R} = \frac{v_{x,R} - v_{y,R}}{\sqrt{2}}, \qquad v_{t,R} = \frac{v_{x,R} + v_{y,R}}{\sqrt{2}},$$

$$v_{n,L} = \frac{v_{x,L} - v_{y,L}}{\sqrt{2}}, \qquad v_{t,L} = \frac{v_{x,L} + v_{y,L}}{\sqrt{2}}.$$

Throughout these test calculations we will set $v_{t,L} = v_{t,R} = 0$; i.e., we will be looking at problems for which there is no slip line in the exact solution. Unless otherwise indicated, the calculations shown were done on a 50×50 grid: $\Delta x = \Delta y = .02$. The results of the calculations are displayed by plotting the profiles of various quantities along the line $y = 1 - x$, and comparing them with the exact solution. In these plots, the computed values at the mesh points are graphed as circles, interpolated by a dotted line: the exact solution is plotted as a solid line. When boundary conditions are required, we assume the solution is constant on lines parallel to the initial jump. This was quite effective in preserving the symmetry of the solution, and enabled us to run for long times without noise from the boundary affecting the results.

The one-dimensional calculations using Glimm's method in the x and y directions require sampling sequences $\vec{a}_x$, $\vec{a}_y$ which we took to be two independent van der Corput sampling sequences: $\vec{a}_x$ was the $(3, 2)$ van der Corput sequence, and $\vec{a}_y$ was the $(5, 3)$ van der Corput sequence. This insured optimal distribution in the square $[0, 1) \times [0, 1)$.

In Fig. 11, we show the results for the following problem:

$$\begin{aligned} &\rho_L = .353, \quad &&\rho_R = .1, \quad &&\gamma = 1.667. \\ &p_L = 14.0, &&p_R = .5, && \\ &v_{N,L} = -1.78, &&v_{N,R} = 11.6, && \end{aligned} \tag{3.2}$$

The exact solution is a strong, right facing shock. It is almost stagnant (after 175 time steps, the exact shock point has moved only two zones). By this time, the oscillations (80% of the exact post-shock value in the pressure) have begun to make themselves known by a three-zone error in the shock location, the shock moving a distance more than two times greater than it should have. We see substantial values (60% of $|v_{n,L} - v_{n,R}|$) for v_T (x, y, t), the tangential component of the velocity appearing. Finally, the density profile shows a substantial deviation from conservation of mass.

The fundamental reason why large errors occur in this problem is that, although each half-step $L^x_{\Delta t}$, $L^y_{\Delta t}$ models the resulting one-dimensional gas dynamics well, the problem it is modeling is $O(1)$ incorrect from the point of view of the two-dimensional

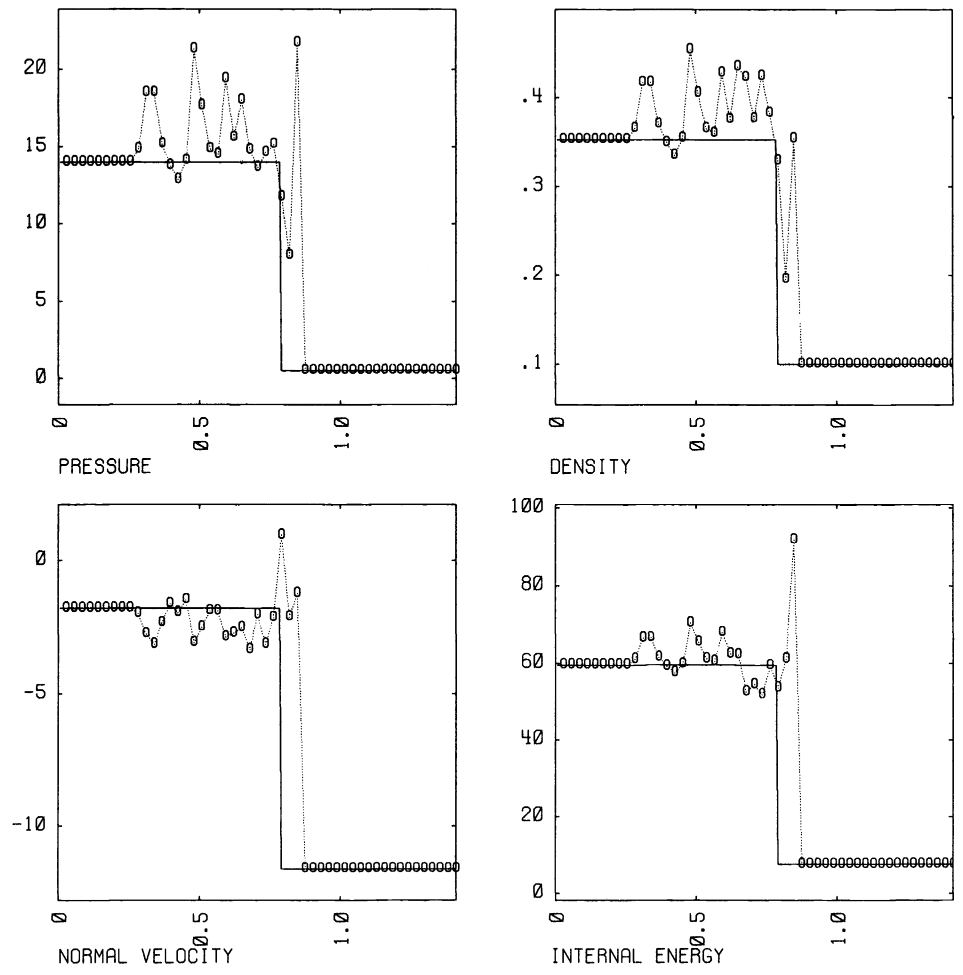


FIG. 11. *Diagonal Reimann problem* (3.2) *computed using Glimm's method*, $\sigma = .5$.

flow. For example, consider the problem one solves (one for each value of j) in the first x-pass in the test problem (3.1). They are each the same Riemann problem for a one-dimensional gas flow, with the jump taking place along the diagonal. The left and right states

$$V_{L,R} = \begin{pmatrix} \bar\rho_{L,R} \\ \bar\rho_{L,R}\bar u_{L,R} \\ \bar\rho_{L,R}(\bar v_y)_{L,R} \\ \bar p_{L,R}/(\gamma-1) + (\bar u^2_{L,R} + (\bar v^2_y)_{L,R})(\bar\rho_{L,R}/2) \end{pmatrix}$$

for the one-dimensional problem are

$$\bar p_{L,R} = p_{L,R}, \qquad \bar\rho_{L,R} = \rho_{L,R},$$

$$\bar u_L = \frac{v_{n,L}}{\sqrt 2}, \quad \bar u_R = \frac{v_{n,R}}{\sqrt 2}, \quad v_{y,L} = \frac{v_{n,L}}{\sqrt 2}, \quad v_{y,R} = \frac{v_{n,R}}{\sqrt 2}.$$

The jump in the velocity, $u_L - u_R$, is less than $v_{n,L} - v_{n,R}$ so a weaker forward facing shock than that of the original two-dimensional problem is produced, as well as a backward facing rarefaction wave. If we sample anywhere in the fan other than the left or right states, we get $(v_x)_{i,j}^{n+\frac{1}{2}} > u_L$, u_R. The new values

$$(v_x)_{i,j}^{n+\frac{1}{2}}, \quad \rho_{i,j}^{n+\frac{1}{2}}, \quad p_{i,j}^{n+\frac{1}{2}}$$

depend only on the sampling value a_x^1 and the ratio $\Delta t/\Delta x$ but not on Δt and Δx separately. So the difference between these and the exact answer is an $O(1)$ quantity relative to the mesh spacing. In particular, there is an $O(1)$ contribution to the tangential component of the velocity. Since there has been an $O(1)$ change in the thermodynamic variables p and ρ, there is no reason for the y-pass to produce a tangential velocity to cancel the one produced by the x-pass, and in fact it does not. Similar phenomena occur for a shock tube, (Fig. 12) or a Riemann problem whose solution consists of two centered rarefaction waves.

The above failures in the splitting procedure in situations when there are discontinuities in p, $\vec{v}$ can be viewed as a consequence of an invalid interchange of limiting procedures. Analytically, shock solutions are obtained as limits of viscous solutions as some set of diffusion coefficients go to zero. One might try to obtain the shocked

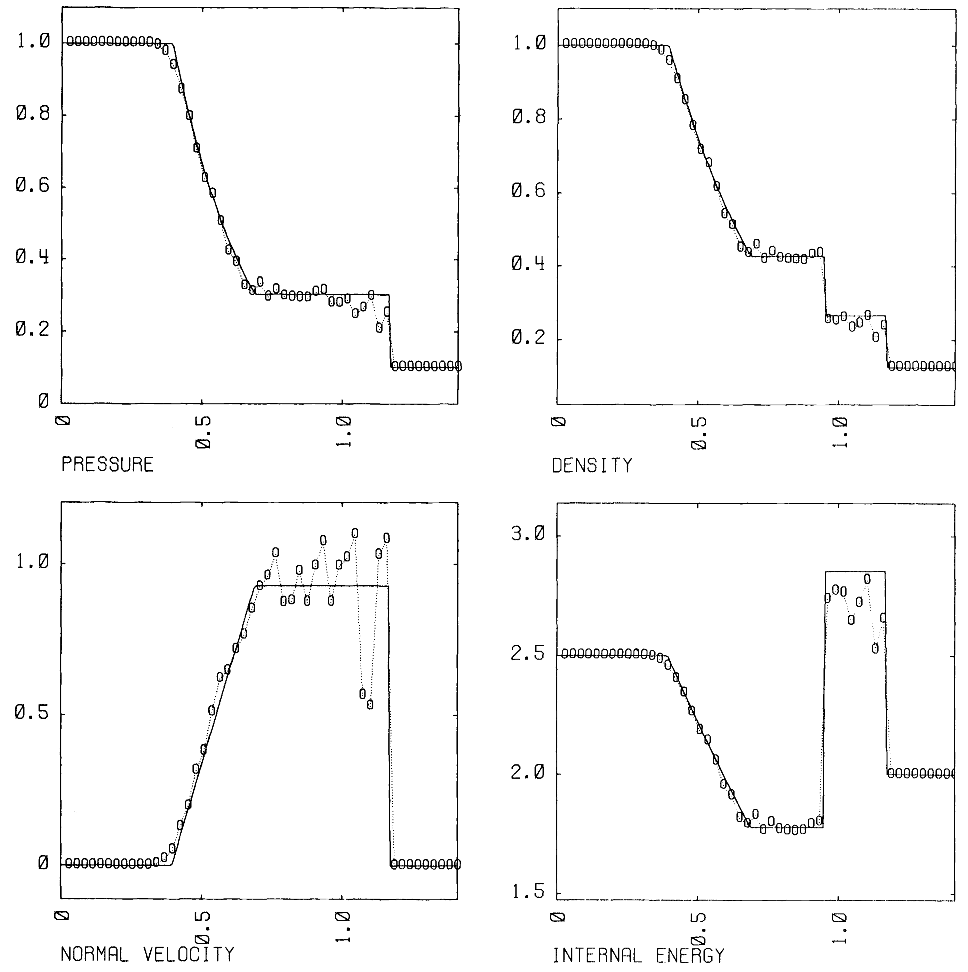


FIG. 12. *Shock tube problem* (2.3) *computed as diagonal Riemann problem using Glimm's method,* $\sigma = .5$.

solutions by using an operator splitting method to solve the viscous equations; the splitting procedure is then known to converge as $\Delta t \to 0$. Then, in the inviscid limit, the viscous solutions converge to the physically correct shocked solutions. In a difference method, the two limiting procedures take place simultaneously, with the coefficients multiplying the numerical diffusion approaching zero with Δt. The use of operator splitting with Glimm's method corresponds to letting the diffusion coefficients vanish for nonzero Δt. This interchange of limits is valid for continuous solutions, or near contact discontinuities, but near discontinuities in p or $\vec{v}$ the two limiting procedures are singular with respect to each other, and cannot be interchanged freely.

In order to correct this problem, we replace Glimm's method at discontinuities in p and $\vec{v}$ with a conservative finite difference method; the method we use in the nonlinear Godunov method (Godunov [2], Richtmyer and Morton [19]) adapted for Eulerian coordinates (Godunov et al. [13]).

We describe the procedure for advancing this hybrid Glimm–Godunov method by one timestep, in one space dimension; the extension to two space dimensions, is achieved by an operator splitting procedure like the one described above. First, one calculates the exact solution to the initial value problem to (1.1), as before. At those mesh points where one uses Glimm's method, one samples as before. At those mesh points $(j\Delta x, (n+1)\,\Delta t)$ where one wishes to use Godunov's method, one sets

$$(U_j^{n+1})_{\text{Godunov}} = \frac{1}{\Delta x}\int_{(j-\frac{1}{2})\Delta x}^{(j+\frac{1}{2})\Delta x} U^e(x, (n+1)\,\Delta t)\,dx.$$

By integrating the conservation law over the rectangle $[(j-\frac{1}{2})\,\Delta x, (j+\frac{1}{2})\,\Delta x]\times[n\,\Delta t, (n+1)\,\Delta t]$ we obtain, using the notation of § 2,

$$(U_j^{n+1})_{\text{Godunov}} = U_j^n + (F(h_{j-\frac{1}{2},n}(0)) - F(h_{j+\frac{1}{2},n}(0)))\frac{\Delta t}{\Delta x},$$

where F is the vector of fluxes for the conservation law being integrated. Finally, we need a prescription for deciding whether to use Glimm or Godunov. Let

$$p_j^{\max} = \max(p_j^n, p_{j-\frac{1}{2}}^{n^*}), \qquad -k_0 \leqq k-j \leqq k_0+1,$$
$$p_j^{\min} = \min(p_j^n, p_{j-\frac{1}{2}}^{n^*}), \qquad -k_0 \leqq k-j \leqq k_0+1,$$

where $p_{j-\frac{1}{2}}^{n^*}$ is the pressure in the region separating the two sonic waves which come from the Riemann problem centered at $((j-\frac{1}{2})\,\Delta x, n\,\Delta t)$ (see [3], [21], and the appendix to this paper). Then the prescription for choosing Glimm and Godunov is

$$(U_j^{n+1})_{\text{hybrid}} = \begin{cases} (U_j^{n+1})_{\text{Godunov}} & \text{if } \dfrac{p_j^{\max} - p_j^{\min}}{p_j^{\min}} > C_0, \\ (U_j^{n+1})_{\text{Glimm}} & \text{otherwise.} \end{cases}$$

Here c_0, k_0 are constants to be set at the beginning of the calculation. Roughly, c_0 is a measure of the strength of the weakest sonic wave in the problem that must be treated as a discontinuity, and k_0+1 is the effective width of a discontinuity. For weak problems (excess pressure ratios $\leqq 5$), it suffices to set $k_0 = 1$. For stronger shocks, it appears to be necessary to set $k_0 = 2$. In all the calculations presented here, $.05 \leqq C_0 \leqq .2$. The consequence of the nonoptimal choice of parameters is a loss of accuracy, not of stability: failing to detect a pressure jump results in noise; using Godunov's method unnecessarily results in the smoothing of relevant wave structures.

One can make several minor modifications of the method described above. The fact that we are using Godunov's method near strong pressure jumps makes it possible to introduce some approximations into the solution of the Riemann problem. Another consequence of sampling only if the sonic waves are weak is that, by accounting approximately for the interaction of those waves, one can use a larger time step, allowing $\sigma < 1$ in (3.1), which is the time step restruction for Godunov's method. We have implemented both of these changes in the method for the examples computed here; in an appendix to this paper, we describe the details of the algorithms used. Finally, we noticed that the first-order splitting algorithm described above can lead to errors near very strong shocks (excess pressure ratios greater than 100) computed using Godunov's method. In particular, large tangential components of velocity are generated behind the shock. We found that the use of the Strang splitting algorithm

$$U^{n+2} = L^x_{\Delta t} L^y_{\Delta t} L^y_{\Delta t} L^x_{\Delta t} U^n$$

reduced this error to the level found in regions where the flow is continuous.

In Fig. 13, we show the results for the problem (3.2) using the hybrid method. Since the solution is a shock discontinuity separating two constant states, this calculation is mostly a test of how well the nonlinear Godunov method computes a strong

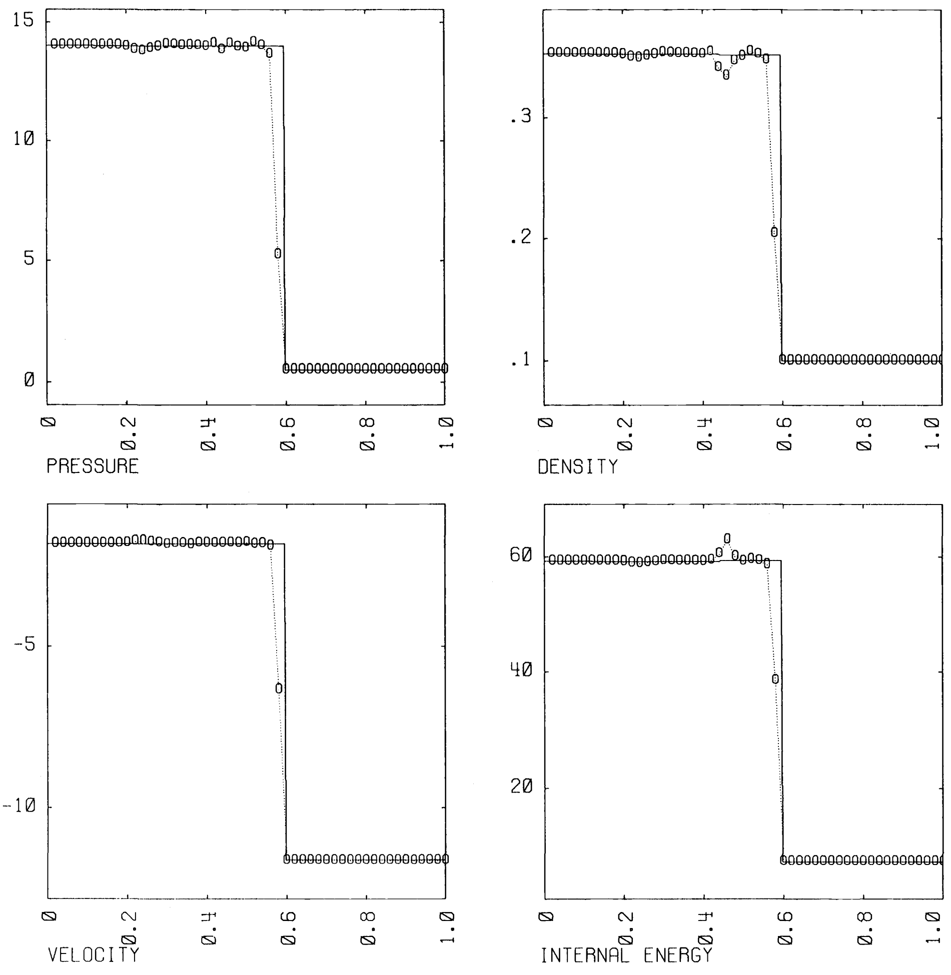


FIG. 13. *Diagonal Riemann problem* (3.2) *computed using hybrid Glimm–Godunov method,* $\sigma = .5$, $C_0 = .1$, $k_0 = 2$.

shock. The dip in the density behind the shock is a starting error, common to most conservative difference methods; it comes from starting a strong shock as a jump discontinuity. Since there is no numerical viscosity away from the shock the oscillation is not damped, but flows downstream unchanged.

Figure 14 shows the results obtained for the shock tube problem (1.3) calculated as a diagonal Riemann problem; Fig. 15 shows the result for the same problem using Godunov's method alone. The hybrid method treats the shock correctly, as opposed to the Glimm's method alone (Fig. 12). The hybrid method is also an improvement over Godunov's method alone: the three waves are clearly resolved; in particular, the contact discontinuity is spread over only three zones. We have found that, in general, the hybrid method spreads any discontinuity over a small (1–4) number of zones, independent of the zone size, regardless of whether the discontinuity is a shock, contact discontinuity, or slip surface.

In order to test this method on a more complex problem, we computed a two-dimensional Cartesian shock reflection problem used by van Leer [23] as a test

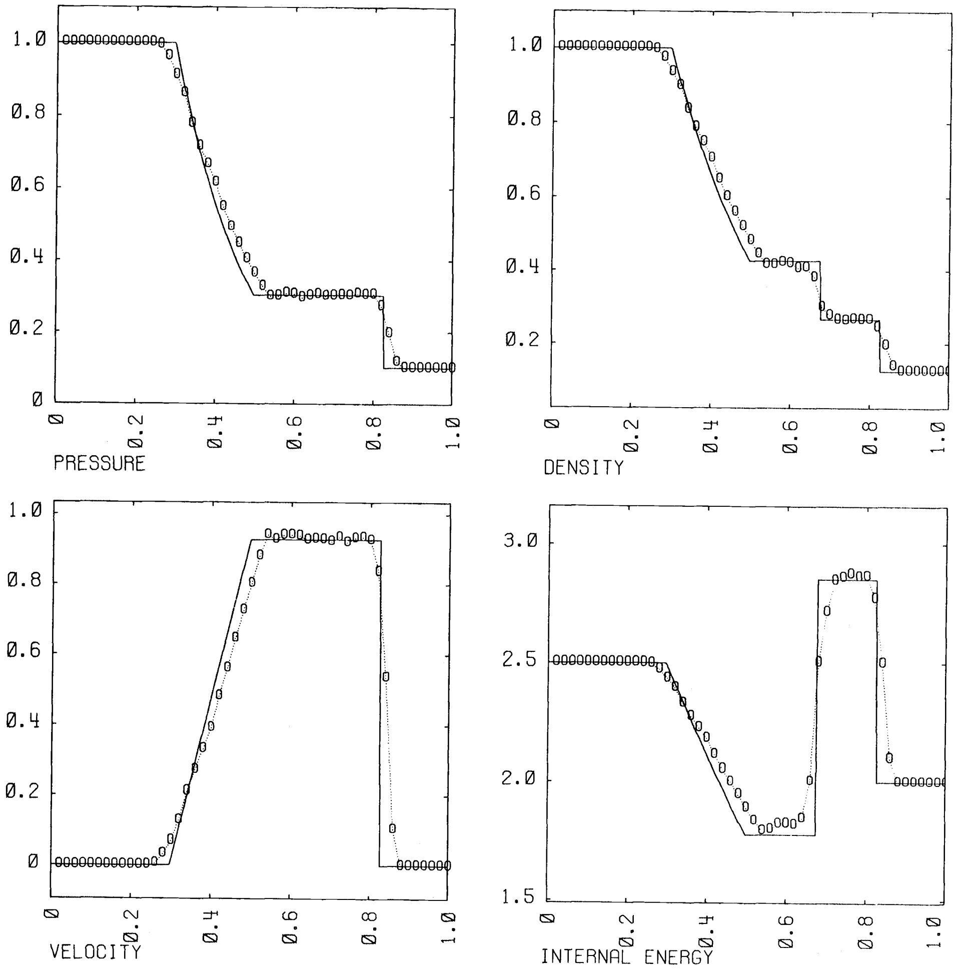


FIG. 14. *Shock tube problem computed as a diagonal Riemann problem using hybrid Glimm–Godunov, method* $\sigma = .5$, $C_0 = .1$, $k_0 = 1$.

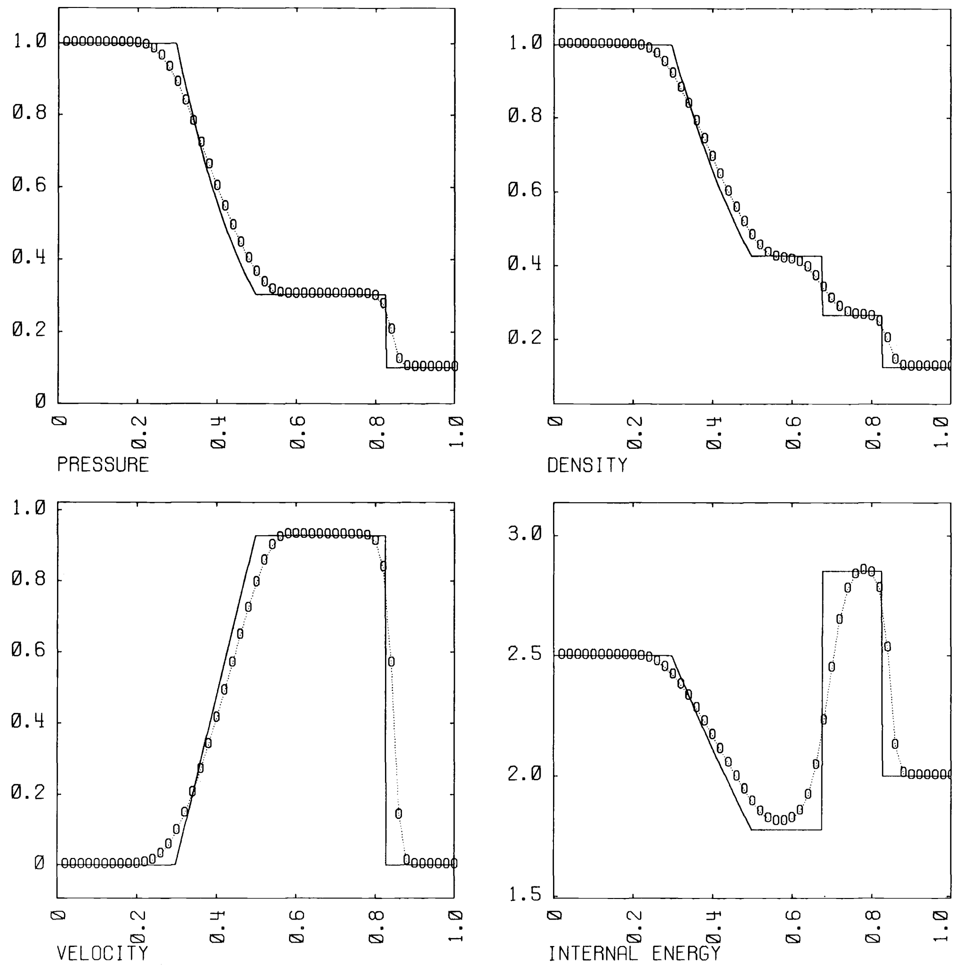


FIG. 15. *Shock tube problem* (2.3) *computed as a diagonal Riemann problem, using Godunov's method,* $\sigma = .5$.

problem; see also Woodward and Colella [24]. The computational domain is a channel of unit length, open at both ends. For $x < .2$, the channel has width $\frac{1}{3}$; at $x = .2$, the lower side of the channel is constricted, so that the width of the channel is $\frac{4}{15}$ for $x > .2$. Reflecting boundary conditions are imposed on the upper and lower sides of the channel, and on the segment $x = .2$, $0 \leqq y \leqq \frac{1}{15}$. The solution is assumed to be continuous at both ends. The initial conditions for this problem are those of uniform flow throughout the tube:

$$p(x, y, 0) = 1, \quad \rho(x, y, 0) = 1.4, \quad v_x(x, y, 0) = 3, \quad v_y(x, y, 0) = 0, \quad \gamma = 1.4;$$

with these initial conditions, a detached shock reflects off the constriction, and reflects off the upper side of the channel, having formed by time $t = \frac{4}{3}$ a three-shock Mach reflection configuration. According to Woodward [24], the correct location of the Mach stem along the side of the channel is right above the constriction at $x = .2$; the Mach stem should extend about one-fourth the distance across the left end of the channel.

We show results obtained using Godunov's method by itself (Fig. 16), and the hybrid method, with two different zone sites (Fig. 17, 18). The solution obtained using Godunov's method alone has the shock slightly in back of the step with the Mach

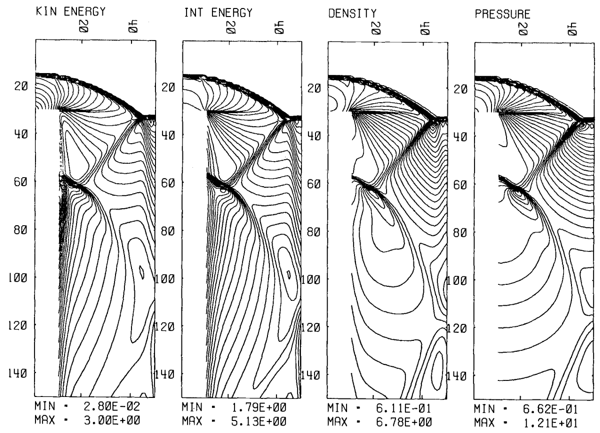


FIG. 16. *Wind tunnel problem at time $t = \frac{4}{3}$ computed using Godunov method, $\Delta x = \frac{1}{150}$, $\sigma = .9$.*

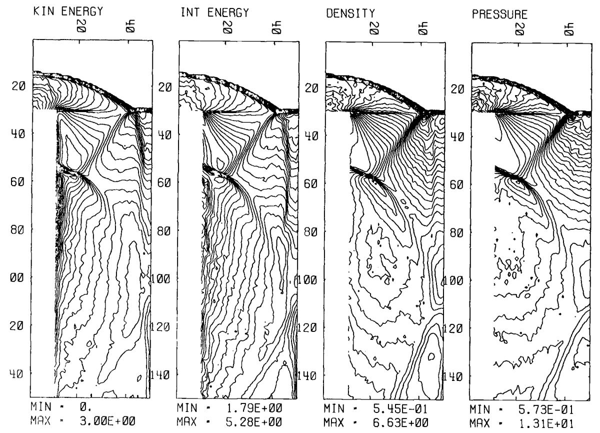


FIG. 17. *Wind tunnel problem at time $t = \frac{4}{3}$ computed using hybrid Glimm–Godunov method, $\Delta x = \frac{1}{150}$, $\sigma = .9$, $C_0 = 1.2$, $k_0 = 2$.*

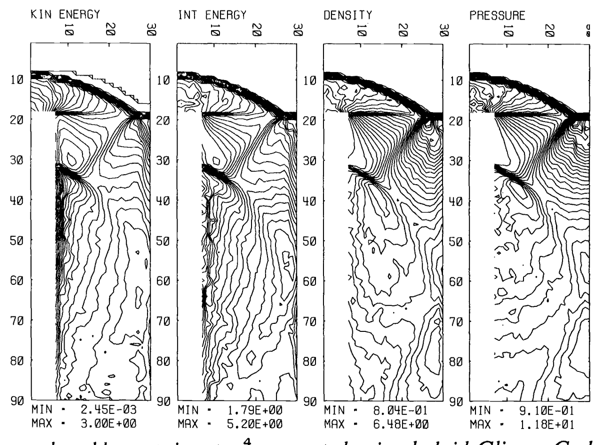


FIG. 18. *Wind tunnel problem at time $t = \frac{4}{3}$ computed using hybrid Glimm–Godunov method, $\Delta x = \frac{1}{90}$, $\sigma = .9$, $C_0 = 2.$, $k_0 = 2$.*

stem about half the correct length. The slip line emerging to the right of the triple point is spread over four or more zones, except right at the shock. The solutions obtained with the hybrid method both have the shock in the correct position to within one zone length, and the length of the stem differs from the cofrect length by two zone lengths. Both of the hybrid calculations have the slip line spread across two zones for its entire length.

4. Discussion and conclusions. In one space variable, Glimm's method has built into it an approximate form of linear and nonlinear wave propagation along characteristics, without the smoothing of such information, as occurs in difference methods, and without any complicated bookkeeping; the sampling procedure determining the weakest wave or wave interaction to be resolved. The motivation for using van der Corput sampling is that one obtains the best possible representation of the wave propagation in Glimm's method, independent of the speed of the waves. This is essential for the correct representation of continuous waves, particularly those produced by nonlinear wave interactions.

We would like to compare the performance of Glimm's method to that of difference methods. Sod [21] performed such a comparison, using a one-dimensional shock tube problem. The results obtained there were not the best possible, due to the use of stratified random sampling. On the other hand, comparing difference methods to Glimm's method on this problem is not entirely fair, either. As is pointed out in [4], it follows from the additivity property for solutions to the Riemann problem that the only values taken on by the computed solution are ones taken on by the exact solution, as well. In any case, we present in Fig. 19 the calculation done with Glimm's method, but using van der Corput sampling. The result obtained here is clearly superior to any of those in [21].

We compared the performance of Glimm's method to that of two difference methods on a shock and rarefaction interaction problem (Fig. 20) like the one described in § 3, but with the waves an order of magnitude stronger:

$$\begin{aligned} p_l &= 473.9, & p_m &= 1.077, & p_r &= 100, \\ \rho_l &= 23.27, & \rho_m &= 3.930, & \rho_r &= 100, \\ u_l &= 6.0, & u_m &= -4.0, & u_r &= -1.181, \\ x_l &= .3, & x_r &= .9, & \gamma &= 1.4. \end{aligned}$$

The solution has the same qualitative features as those of the weaker problem, except that the backward facing compression wave produced by the shock-rarefaction interaction has itself steepened into a shock at the time the solutions are compared. Otherwise, the waves are all much stronger; in particular, the passively advected density wave is a spike, two zones in width for the $\Delta x = .01$ cases. The two difference methods compared are the version of Godunov's method (Fig. 21) discussed in the previous section, and the MUSCL code written by Paul Woodward of Lawrence Livermore National Laboratory, based on the scheme of van Leer [23] (Fig. 22). These two methods represent, respectively, one of the most accurate of the first-order methods, and a state-of-the-art representative of the adaptive or hybrid difference methods. (For other examples, see Boris and Book [2], Harten [15], Harten and Zwas [16], Zalesak [25].) As before, we compare all three results with the answer obtained using Glimm's method, van der Corput sampling with $\Delta x = .0025$. All three methods obtain reasonably good answers for the pressure and velocity profiles, modulo the varying widths for the shock transition region. However, neither of the difference methods

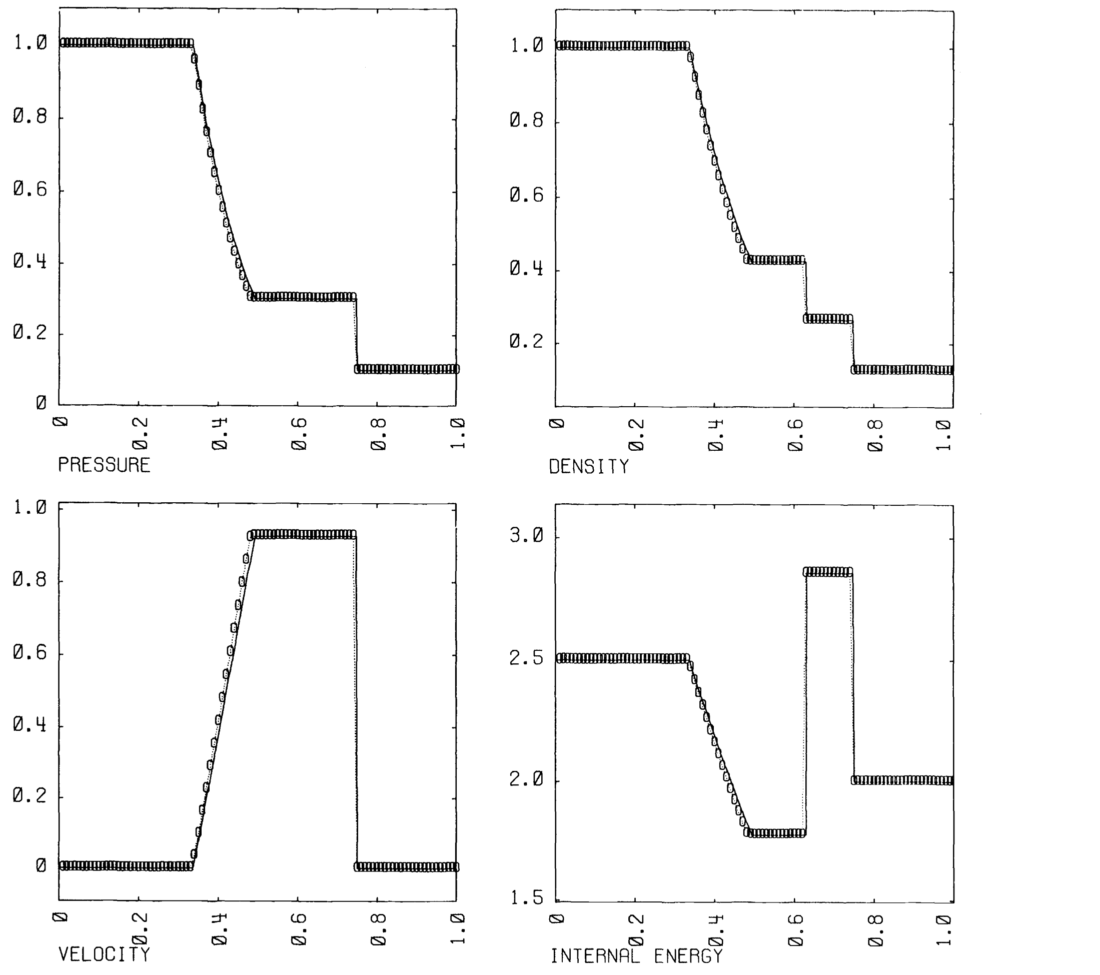


FIG. 19. *Solution to shock tube problem* (2.3) *in one dimension, computed using Glimm's method with van der Corput sampling,* $\Delta x = .01$.

are able to get the correct peak value of the density, nor the correct width for the density spike; Glimm's method, by virtue of its direct simulation of the wave interaction process without averaging, gets the correct answer.

The original proposal in [3] for using Glimm's method with operator splitting was seen to give incorrect results for flows in which there occur large jumps in the pressure and velocity along surfaces oblique to the mesh directions. By coupling Glimm's method with Godunov's method, we lose many of the special properties of the Glimm's method with respect to its treatment of shock interactions. However, the resulting method has a number of attractive properties. Of all the first-order difference methods, Godunov's method produces the narrowest shocks (2–3 zones wide). Both Glimm's method and Godunov's method are extremely stable, even in the strongly nonlinear region (the problem with Glimm's method at shocks is a loss of accuracy, not of stability). Finally, Glimm's method has no numerical diffusion, so that the hybrid method has no numerical diffusion away from regions where large pressure gradients are generated.

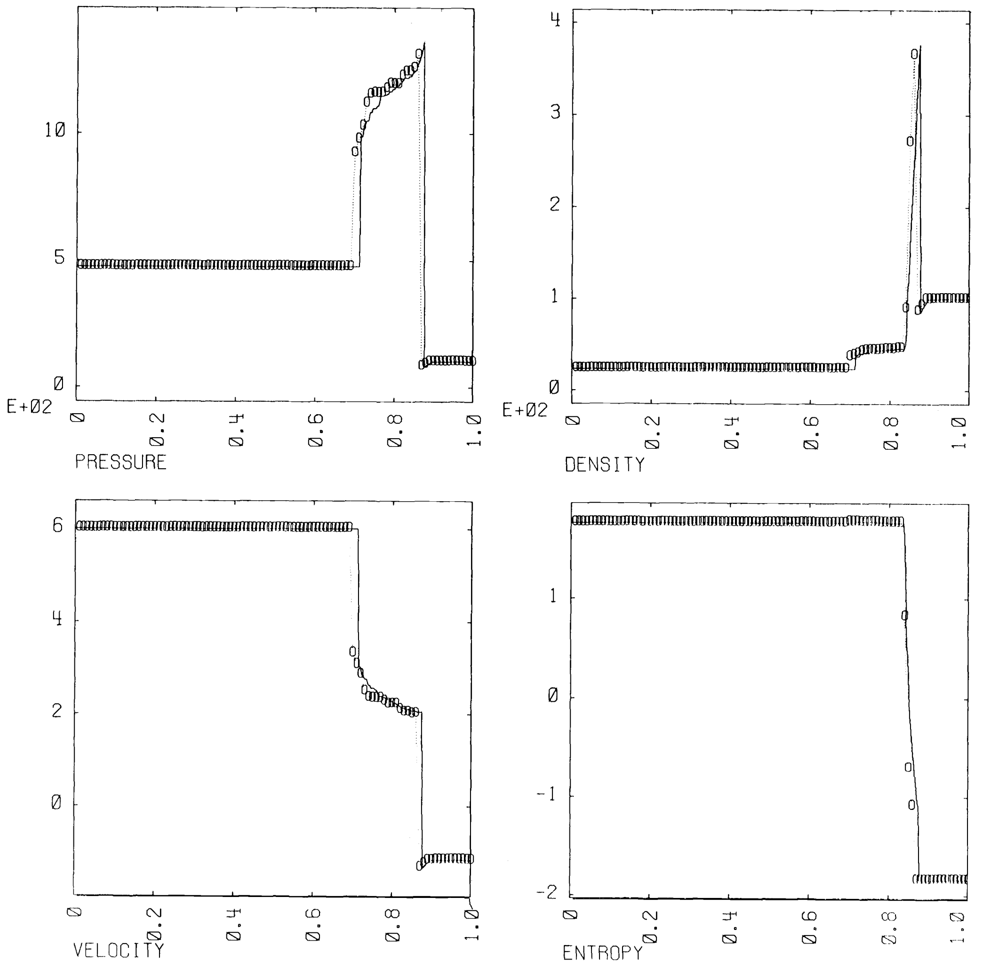


FIG. 20. *Solution to strong* 1D *interaction problem, computed using Glimm's method.*

For the purpose of comparison with the results of Sod [21], obtained by the various difference methods, we computed the shock tube problem (2.3) as a diagonal Riemann problem (Fig. 23), but on a 100×100 grid ($\Delta x = \Delta y = .01$); results for this problem compted using the MUSCL code are also given in [23]. In principle, the problem solved here is more difficult than the one solved in [21], since the latter is solved as a one-dimensional problem. But the answer is the same for both, and the results are worth comparing.

The calculation of the rarefaction, and the width of the shock transition in the results obtained with the hybrid Glimm–Godunov method, compare favorably with those obtained by any of the difference methods. The treatment of the contact discontinuity is clearly superior to that given by any of the difference methods in [21], and comparable to that obtained in [23]. The difference methods in [21] either spread the contact discontinuity over 6–10 zones, with the number of zones increasing as a function of time, or introduce substantial oscillations near the contact discontinuity.

The major weakness of the hybrid method is that it computes shocks which are 2–3 zones wide. This puts the method at a disadvantage compared to the methods

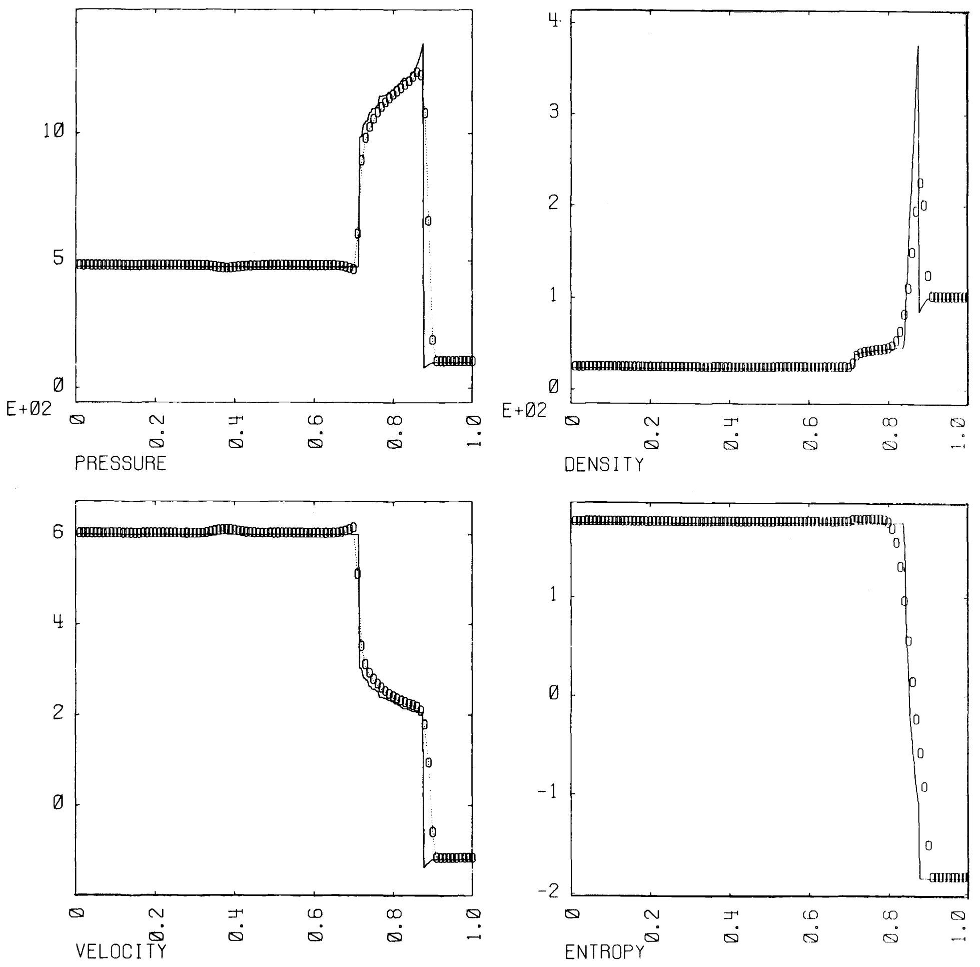


FIG. 21. *Solution to strong* 1D *interaction problem, computed using Godunov's method.*

which have narrower shocks, in computing problems such as the Mach reflection problem discussed in the previous section. The number of timesteps required after the time of reflection for the Mach stem to form increases as does the width of the shock; consequently, the MUSCL code having shocks which are 1–2 zones wide obtains, on coarser meshes, results comparable to those obtained here.

There are several directions in which further work is indicated. For one-dimensional flows, Glimm's method with van der Corput sampling is quite effective in modelling the interaction of discontinuities with smooth parts of the flow, without introducing unacceptable errors in the latter. The fact that the solutions to the Riemann problem we use in the numerical scheme satisfy exactly the conservation laws is probably not essential to the accuracy of the method, since much of that information is lost in the sampling. What is essential is that the solution which is sampled has built into it the physically correct waves and wave speeds to some reasonable order of accuracy. Thus it is feasible to try to model with Glimm's method the dynamics of other media than an ideal gas in Cartesian coordinates: for example, gas dynamics with source terms or unusual equations of state, or elastic-plastic flow.

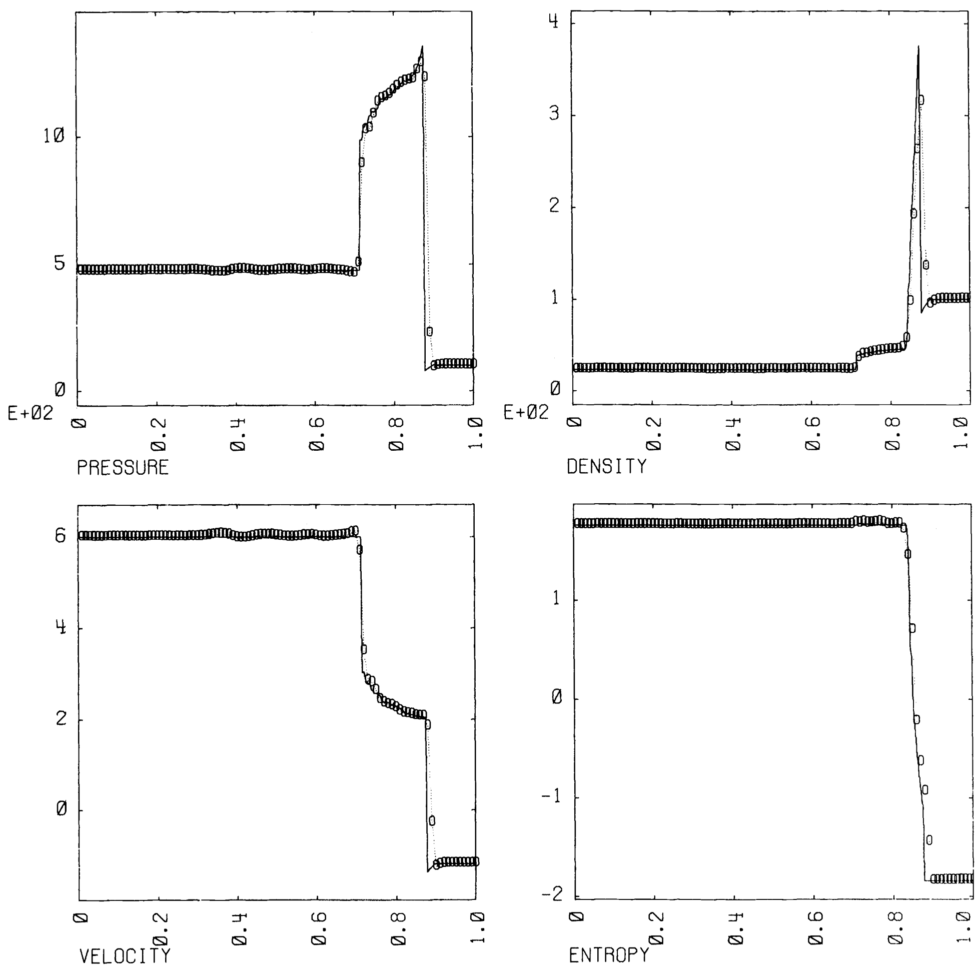


FIG. 22. *Solution to strong* 1D *interaction problem, computed using* MUSCL.

The central advantage of the hybrid Glimm–Godunov method is that it has the simplicity and stability of a first-order method, with substantially less numerical diffusion than is usually seen in first-order methods. As the method is currently engineered, it seems to be more accurate than the nonadaptive first- and second-order methods, but not as accurate as some of the adaptive methods. The main question to be answered is the determination of a set of optimum engineering decisions. One problem is that we have seen that the criterion for whether to use Glimm's method or Godunov's method at a point is different depending on the strength of the waves; the distinction between strong and weak waves should be made locally, by the algorithm. More generally, although the general principle for switching between the two methods is clear, the actual details of the procedure are still determined in a fairly ad hoc, problem-dependent fashion, and a more systematic algorithm is needed.

Appendix. Calculation of approximate solutions to the Riemann problem. In this appendix, we present a detailed description of the procedure used to calculate approximate values to the solution to the Riemann Problem in the two-dimensional hybrid Glimm–Godunov calculations described in § 2. The approximations introduced here

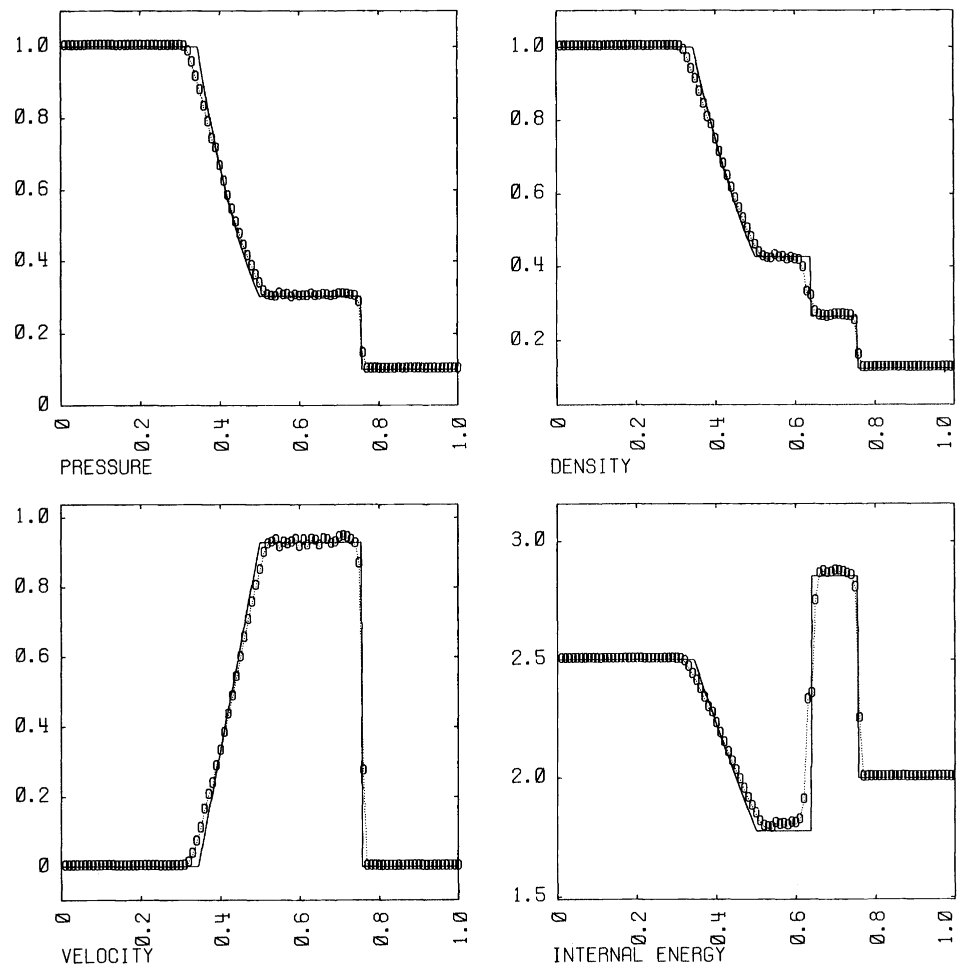


FIG. 23. *Shock tube problem* (2.3) *computed as a diagonal Riemann problem using hybrid Glimm–Godunov method,* $C_0 = .05$, $k_0 = 1$, $\sigma = .9$.

are designed to give sufficiently accurate answers for the minimum computational effort in the two situations which arise in those calculations: 1) the sonic waves are weak, or 2) the sonic waves are strong, but the values calculated are used only for computing fluxes in Godunov's method. The algorithm given here is also better suited for efficient implementation on a vector processor, such as the Cray-I, than those given previously.

The first step is an iteration to calculate p^*, the pressure of the gas between the two sonic waves. We use the Newton's method algorithm given in van Leer [23], with one important modification: we assume that the formulae for $W_{L,R}$, the mean Lagrangian wave speeds, are the same for both shocks and rarefactions:

$$W_{L,R}(p^*) = C_{L,R}\sqrt{1+\frac{\gamma+1}{2\gamma}\frac{(p^*-p_{L,R})}{p_{L,R}}},$$

$$C_{L,R} = \sqrt{\gamma p_{L,R}\rho_{L,R}} = \rho_{L,R}c_{L,R}.$$

Then the iteration proceeds as follows:

$$p^{*,0} = \frac{C_L p_R + C_R(p_L - C_L(u_R - u_L))}{C_L + C_R},$$

$$W^l_{L,R} = W_{L,R}(p^{*,l-1}),$$

$$Z_{L,R} = \frac{2(W^l_{L,R})^3}{C^2_{L,R} + (W^l_{L,R})^2},$$

$$u^{*,l}_L = u_L - \frac{p^{*,l-1} - p_L}{W_L}, \quad u^{*,l}_R = u_R + \frac{p^{*,l-1} - p_R}{W^l_R},$$

$$p^{*,l} = p^{*,l-1} - \frac{Z_L Z_R (u^{*,l}_R - u^{*,l}_L)}{Z_L + Z_R}.$$

One criterion for terminating the iteration is to terminate if

$$\frac{|p^{*,l} - p^{*,l-1}|}{p^{*,l}} < \varepsilon,$$

where $\varepsilon > 0$ is some predetermined tolerance, and set $p^* = p^{*,l}$. In programming this procedure for the Cray-I, we iterated a fixed number of times l_0, i.e., set $p^* = p^{*,l_0}$, independent of the left and right states. We obtained more than adequate accuracy using $l_0 = 4$, for even the strongest problems; it appears to be sufficient, for a wide class of problems, to set l_0 to be 1 or 2.

Having obtained p^*, we calculate the other quantities we will need:

$$W_{L,R} = W_{L,R}(p^*), \qquad u^* = \frac{p_L - p_R + u_L W_L + u_R W_R}{W_L + W_R}.$$

If, at any point in the iteration, $p^{*,l} < 0$, we reset p^* to be equal to some floor value $1 \gg p_{\min} > 0$. If $p^{*,l} < 0$ for two iterations in a row, we terminate the iteration (or ignore the results), setting $p^* = p_{\min}$.

The second part of the procedure is to calculate the value of the solution at some given point (x, t); we denote the values of the pressure, density, velocity, and passive component of the velocity at that point by p, ρ, u, v. We follow the procedure given in [3], but use explicitly the reflection symmetry of the equations to consolidate some of the formulae.

Let $\psi = x/t$ and $s = \operatorname{sgn}(\psi - u^*)$. Then we define

$$(u_0, p_0, \rho_0, W_0, c_0, v_0) = \begin{cases} (u_L, p_L, \rho_L, W_L, c_L, v_L) & \text{if } s = -1, \\ (u_R, p_R, \rho_R, W_R, c_R, v_R) & \text{if } s = 1; \end{cases}$$

$$\bar{u}_0 = s u_0, \quad \bar{\psi} = s\psi, \quad \bar{u}^* = s u^*.$$

We then compute

$$\rho^* = \left(\rho_0^{-1} - \frac{(p^* - p_0)}{W_0^2}\right)^{-1}, \qquad c^* = \sqrt{\frac{\gamma p^*}{\rho^*}},$$

$$\lambda_0, \lambda^* = \begin{cases} \bar{u}_0 + c_0, \bar{u}^* + c^* & \text{if } p^* < p_0, \\ \bar{u}_0 + \dfrac{W_0}{\rho_0} & \text{if } p^* > p_0. \end{cases}$$

We evaluate p, ρ, u as follows:

$$p, \rho, u = \begin{cases} p^*, \rho^*, u^* & \text{if } \bar{\psi} \leqq \lambda^*, \\ p_0, \rho_0, u_0 & \text{if } \bar{\psi} > \lambda_0. \end{cases}$$

If $\lambda_0 > \bar{\psi} > \lambda^*$, then the solution is being evaluated inside a centered rarefaction wave, and we have

$$c = \left(\bar{\psi} - \bar{u}^* + \frac{2c^*}{\gamma - 1}\right)\frac{\gamma - 1}{\gamma + 1}, \quad u = s(\bar{\psi} - c), \quad \rho = \left(\frac{c}{c_0}\right)^{2/(\gamma-1)} \rho_0, \quad p = \frac{\rho c^2}{\gamma}.$$

One can replace the expression for ρ inside a rarefaction fan by the formula

$$\rho = \rho_0 \frac{1 - \alpha}{1 + \alpha}, \qquad \alpha = \frac{1}{\gamma - 1} \frac{c_0^2 - c^2}{c_0^2 + c^2},$$

which does not require evaluation of a rational power.

If the iteration is carried out to convergence, all the approximations introduced here are correct to third order in the pressure jump across the strongest rarefaction wave present. If one of the waves is a strong rarefaction, we will be using the results to calculate fluxes for Godunov's method, and the error committed by using the rarefaction shock formula in the iteration is lost in the averaging. However, it is essential to evaluate the solution inside the rarefaction fans, rather than treating the waves as jump discontinuities, as is done in [13]. Otherwise, sampling would not spread weak rarefactions; nor would the averaging in Godunov's method spread strong rarefaction shocks into rarefaction waves, if the speed of the rarefaction shock is close to zero.

Using approximations similar to those used above in computing the solution to the Riemann problem, we extend the sampling procedure to the case where the time step satisfies (3.1) for $\frac{1}{2} < \sigma < 1$, assuming that the sonic waves in the solution being sampled are weak. The procedure described below accounts correctly for the possible interpenetration of waves from successive Riemann problems to first order in the strengths of the sonic waves, and reduces to the previous sampling procedure when the waves do not intersect.

To update the solution at zone j, we first evaluate at $((j - \frac{1}{2} + a^{n+1})\,\Delta x, (n+1)\,\Delta t)$ the solution to the Riemann problems on either side of the zone:

$$U_{+,j}^{n+1} = h_{j-\frac{1}{2},n}\left(a^{n+1}\frac{\Delta x}{\Delta t}\right),$$

$$U_{-,j}^{n+1} = h_{j+\frac{1}{2},n}\left((a^{n+1} - 1)\frac{\Delta x}{\Delta t}\right),$$

$$U_{+,j}^{n+1} = U_{j-1}^n \text{ or } U_j^n \quad \text{or} \quad U_{-,j}^{n+1} = U_j^n \text{ or } U_{j+1}^n.$$

Then we can assume, since the sonic waves are weak, that the waves from the Riemann problem at $((j - \frac{1}{2})\,\Delta x, n\,\Delta t)$ and $((j + \frac{1}{2})\,\Delta x, n\,\Delta t)$ do not intersect, and set

$$(U_j^{n+1})_{\text{Glimm}} = \begin{cases} U_{-,j}^{n+1} & \text{if } U_{+,j}^{n+1} = U_j^n, \\ U_{+,j}^{n+1} & \text{otherwise.} \end{cases}$$

If $U_{+,j}^{n+1} \neq U_j^n, U_{j-1}^n$ and $U_{-,j}^{n+1} \neq U_j^n, U_{j+1}^n$, then we assume that the waves contained in the jump $(U_{+,j}^{n+1}, U_j^n)$ have reached and interacted with the waves contained in the jump $(U_j^n, U_{-,j}^{n+1})$. Using again the assumption that the sonic waves are weak, we see that the waves which intersect are: a part of a backward facing sonic wave, from the Riemann problem at $((j+\frac{1}{2})\,\Delta x, n\,\Delta t)$; a part of the forward facing sonic wave from the Riemann problem at $((j-\frac{1}{2})\,\Delta x, n\,\Delta t)$ and at most one contact discontinuity, which might come from either one of the Riemann problems (Fig. A1). In that case, we calculate $(p_j^{n+1}, \rho_j^{n+1}, u_j^{n+1}, v_j^{n+1})_{\text{Glimm}}$ as follows:

$$W_R = W_{R,j-\frac{1}{2}}^{n+1} \sqrt{\gamma p_{-,j}^{n+1} \rho_{-,j}^{n+1}} / \sqrt{\gamma p_j^n \rho_j^n},$$

$$W_L = W_{L,j+\frac{1}{2}}^{n+1} \sqrt{\gamma p_{+,j}^{n+1} \rho_{+,j}^{n+1}} / \sqrt{\gamma p_j^n \rho_j^n},$$

$$p_j^{n+1} = (W_R p_{-,j}^{n+1} + W_L(p_{+,j}^{n+1} - W_L \times (u_{-,j}^{n+1} - u_{-,j}^{n+1})))/(W_L + W_R),$$

$$u_j^{n+1} = u_{+,j}^{n+1} + W_R(p_j^{n+1} - p_{+,j}^{n+1}),$$

$$\left.\begin{aligned} \rho_j^{n+1} &= \left((\rho_{+,j}^{n+1})^{-1} - \frac{p_j^{n+1} - p_{+,j}^{n+1}}{W_R^2}\right)^{-1} \\ v_j^{n+1} &= v_{+,j}^{n+1} \end{aligned}\right\} \quad \text{if } (a^{n+1} - 1)\frac{\Delta x}{\Delta t} < u_{j+\frac{1}{2}}^{*,n+1},$$

$$\left.\begin{aligned} \rho_j^{n+1} &= \left((\rho_{-,j}^{n+1})^{-1} - \frac{p_j^{n+1} - p_{-,j}^{n+1}}{W_L^2}\right)^{-1} \\ v_j^{n+1} &= v_{-,j}^{n+1} \end{aligned}\right\} \quad \text{otherwise.}$$

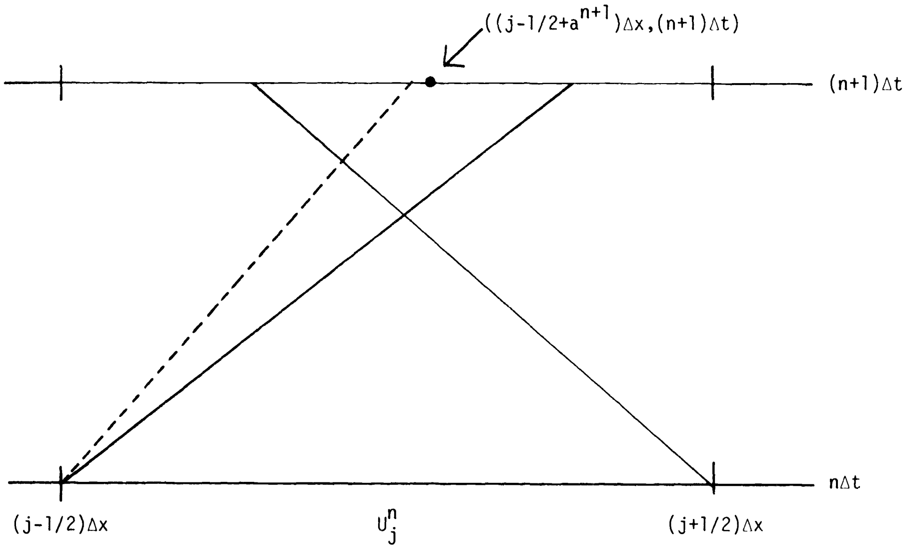


FIG. A1. *Interaction of waves from adjacent Riemann problems.*

Here $W_{L,j+\frac{1}{2}}^{n+1}$, $W_{R,j-\frac{1}{2}}^{n+1}$, $u_{j+\frac{1}{2}}^{*,n+1}$ are the mean Lagrangian wave speeds and the velocity of the gas between the two sonic waves for Riemann problems centered at $((j+\frac{1}{2})\,\Delta x, n\,\Delta t)$.

The above scheme can be implemented in such a way that almost all the calculations are vectorizable. For the test problems discussed in § 4, a program run on the Cray-I at the LLNLCC, compiled using the CFT compiler, took about 14 μs/zone/time step/space dimension, or about 35,000 zones/second for a two-dimensional problem. The iteration scheme for computing p^*, u^* is done once per zone, independent of whether one is sampling or averaging in that zone, and takes $(1.7 + 1.15 \times l_0)\ \mu s = 4\ \mu s$ for $l_0 = 2$, or less than one third of the time per timestep. A good deal of redundant work is performed because of the limited number of vectorized logical operations available at the Fortran level using the CFT compiler. As more of the Cray-I's capabilities become accessible, such as vectorized gather/scatter operations and bit-vector logic, the timing should improve substantially.

Acknowledgments. Part of this work is contained in a Ph.D. dissertation written at the University of California, Berkeley. The author wishes to thank his thesis advisor, Alexandre Chorin, for his encouragement and support. The author also thanks G. Sod and P. Woodward for many helpful discussions, and the Mathematics Group at Lawrence Berkeley Laboratory and the Theoretical Physics Division at Lawrence Livermore National Laboratory for their generous financial support.

REFERENCES

[1] N. ALBRIGHT, P. CONCUS, AND W. PROSKUROWSKI, *Numerical solution of the multidimensional Buckley–Leverett equation by a sampling method*, Rep. LBL-8452, Lawrence Berkeley Lab., Univ. of California, Berkeley, 1978.

[2] J. P. BORIS AND D. L. BOOK, *Flux-corrected transport.* I. SHASTA, *A fluid transport algorithm that works*, J. Comput. Phys., 23 (1973), pp. 38–69.

[3] A. J. CHORIN, *Random choice solution of hyperbolic systems*, J. Comput. Phys., 22 (1976), pp. 517–536.

[4] ———, *Random choice methods with application to reacting gas flow*, J. Comput. Phys., 25 (1977), pp. 252–272.

[5] P. COLELLA, *An analysis of the effect of operator splitting and of the sampling procedure on the accuracy of Glimm's method*, Ph.D. Dissertation, Univ. of California, Berkeley, June 1979.

[6] ———, *Error analysis of Glimm's method*, to appear.

[7] P. CONCUS AND W. PROSKUROWSKI, *Numerical solution of a nonlinear hyperbolic equation by a random choice method*, J. Comput. Phys., 30 (1979), pp. 153–166.

[8] R. COURANT AND K. O. FRIEDRICHS, *Supersonic Flow and Shock Waves*, Interscience, New York, 1948.

[9] J. GLIMM, *Solution in the large for nonlinear hyperlinear systems of equations*, Comm. Pure Appl. Math., 18 (1955), pp. 697–715.

[10] J. GLIMM, D. MARCHESIN AND O. MCBRYAN, *A numerical method for two phase flow with an unstable interface*, preprint.

[11] J. GLIMM, D. MARCHESIN, E. ISAACSON AND O. MCBRYAN, *A shock tracking method for hyperbolic systems*, preprint.

[12] S. K. GODUNOV, *Difference methods for the numerical calculations of discontinuous solutions of the equations of fluid dynamics*, Mat. Sb., 47 (1959), pp. 271–306 (in Russian).

[13] S. K. GODUNOV, A. V. ZABRODIN AND G. P. PROKOPOV, *A comutational scheme for two-dimensional nonstationary problems of gas dynamics and calculations of the flow from a shock wave approaching a stationary state*, USSR Computational Math. and Math. Phys., 1 (1961), pp. 1187–1218.

[14] J. M. HAMMERSLEY AND D. C. HANDSCOMB, *Monte Carlo Methods*, Methuen, London, 1965.

[15] A. HARTEN, *The method of artificial compression*: I. *Shocks and contact discontinuities*, AEC Res. and Development Rep. C00-3077-50, Courant Institute for the Mathematical Sciences, New York Univ., New York, 1974.

[16] A. HARTEN AND G. ZWAS, *Self-adjusting hybrid schemes for shock computations*, J. Comput. Phys., 9 (1972), pp. 568–583.

[17] P. D. LAX, *Hyperbolic systems of equations and computing*, SIAM Rev. 11 (1969), pp. 7–19.

[18] F. H. MALTZ AND D. L. HITZL, *Variance reduction in Monte Carlo computations using multi-dimensional Hermite polynomials*, J. Comput. Phys., 32 (1979), pp. 345–376.

[19] R. RICHTMYER AND K. MORTON, *Difference Methods for Initial-Value Problems*, 2nd ed., Interscience, New York, 1967.

[20] G. A. SOD, *A numerical study of a converging cylindrical shock*, J. Fluid Mech., 83 (1977), pt. 4, pp. 785–794.

[21] ———, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comput. Phys., 27 (1978), pp. 1–31.

[22] ———, *A numerical method for a slightly viscous axisymmetric flow with application to internal combustion engines*, Rep. LBL-9049, Lawrence Berkeley Lab. Univ. of California, Berkeley, 1979.

[23] B. VAN LEER, *Towards the ultimate conservative difference scheme*, V. *A second order sequel to Godunov's methods*, J. Comput. Phys., 32 (1979), pp. 101–136.

[24] P. WOODWARD AND P. COLELLA, *High resolution difference schemes for compressible gas dynamics*, in Proc. 7th Int. Conf. on Numerical Methods in Fluid Dynamics, June 1980.

[25] S. T. ZALESAK, *Fully multidimensional flux-corrected transport algorithms for fluids*, J. Comput. Phys., 31 (1979), pp. 335–362.

SIAM J. Sci. Stat. Comput.
Vol. 3, No. 1, March 1982

A COMPUTATIONAL APPROACH FOR CONSTRUCTING SINGULAR SOLUTIONS OF ONE-DIMENSIONAL PSEUDOPARABOLIC AND METAPARABOLIC EQUATIONS*

ROBERT P. GILBERT† AND JARL JENSEN‡

Abstract. In the present work a computational approach is developed for constructing singular solutions of the one-dimensional pseudoparabolic and metaparabolic equations. Recursive schemes are developed to determine expansion coefficients for the singularities. It is shown, furthermore, that the coefficients are themselves special solutions of ordinary differential equations. Closed form expressions are obtained for these coefficients using a symbolic and algebraic manipulation language. Several examples are provided to indicate the usefulness of this new approach to construct fundamental solutions.

Key words. Pseudoparabolic equations, metaparabolic equations, fundamental solutions, Riemann functions, FORMAC, symbol manipulation.

1. Fundamental solutions of metaparabolic equations.

As has been pointed out in [3], [8] pseudo- and metaparabolic equations occur in the modeling of a variety of physical problems such as the velocity of a nonsteady flow of a viscous fluid, the theory of seepage of homogeneous fluids through fissured rock, hydrostatic excess pressure during the consolidation of clay and the stability of liquid-filled shells. We consider metaparabolic equations in one space variable. These are equations of the form

$$M[u] := L[u] + M[u_t] = 0, \tag{1.1}$$

where

$$\begin{aligned} L[u] &:= \sum_{k=0}^{n} \alpha_k \frac{\partial^k u}{\partial x^k} \equiv \sum_{k=0}^{n} \frac{\partial^k}{\partial x^k}(a_{n-k}u), \qquad a_0 \equiv 1, \\ M[u] &:= \sum_{k=0}^{m} \beta_k \frac{\partial^k u}{\partial x^k} \equiv \sum_{k=0}^{m} \frac{\partial^k}{\partial x^k}(b_{m-k}u), \qquad m < n. \end{aligned} \tag{1.2}$$

The coefficients α_k, β_k may be taken to be arbitrary functions of x and t; however, since we will be interested in developing a function theoretic approach we usually require some amount of regularity.

Equation (2.1) may be rewritten in the form

$$\begin{aligned} \frac{\partial^n}{\partial x^n}\Bigg[u(x,t) + \sum_{k=0}^{n-1} \int_0^x \frac{(x-y)^k}{k!} a_{k+1} u(y,t)\,dy \\ + \sum_{k=0}^{m} \int_0^x \frac{(x-y)^{n-m+k-1}}{(n-m+k-1)!} b_k u_t(y,t)\,dy\Bigg] = 0, \end{aligned} \tag{1.3}$$

which by integrating n times becomes

$$\begin{aligned} u(x,t) + \sum_{k=0}^{n-1} \int_0^x \frac{(x-y)^k}{k!} a_{k+1}(y,t) u(y,t)\,dy \\ + \sum_{k=0}^{m} \int_0^x \frac{(x-y)^{n-m+k-1}}{(n-m+k-1)!} b_k(y,t) u_t(y,t)\,dy = \sum_{k=0}^{n-1} \phi_k(t) x^k := \Phi(x,t). \end{aligned} \tag{1.4}$$

Here the $\phi_k(t)$ may be determined by prescribing suitable boundary conditions.

* Received by the editors February 18, 1981.

† Applied Mathematics Institute, University of Delaware, Newark, Delaware 19711.
‡ Department of Solid Mechanics, The Technical University of Denmark, Lyngby, Denmark.

A procedure which is useful for constructing solutions for metaparabolic equations is based on determining a fundamental solution. See in this regard Colton [4], [5], [6], Gilbert–Hsiao [8], Brown–Gilbert–Hsiao [3], Gilbert–Roach [9], Gilbert–Schneider [10, [11] and Gilbert [7]. The methodology developed in these works suggests we seek a fundamental solution S of the form

$$S(x,t;\xi,\tau):=\sum_{i=0}^{\infty}\frac{s_l(x,\xi)l!(-1)^{l+1}}{(t-\tau)^{l+1}} \tag{1.5}$$

in the instance where the coefficients a_k, b_k are functions of the space variable alone. For what follows we shall make this assumption.

Upon substituting the series for S into the differential equation (1.1)–(1.2) we obtain a recursive scheme for the $s_l(x,\xi)$, namely

$$L[s_0]=0,\qquad L[s_{l+1}]=-M[s_l],\qquad l=0,1,\cdots. \tag{1.6}$$

Rewriting these in integral form, much in the same way as we did above for (1.1) and (1.2), we obtain

$$\begin{aligned} &s_{l+1}(x,\xi)+\sum_{k=0}^{n-1}\int_\xi^x\frac{(x-y)^k}{k!}a_{k+1}(y)s_{l+1}(y,\xi)\,dy\\ &\quad+\sum_{k=0}^{m}\int_\xi^x\frac{(x-y)^{n-m+k-1}}{(n-m+k-1)!}b_k(y)s_l(y,\xi)\,dy=\Phi_{l+1}(x,\xi),\qquad l=0,1,\cdots. \end{aligned} \tag{1.7}$$

For convenience, we choose $\Phi_l\equiv 0$, $l=1,2,\cdots$ and the first coefficient $s_0(x,\xi)$ as

$$s_0(x,\xi):=G(\xi,x), \tag{1.8}$$

where $G(\xi,x)$ is a Riemann function for $L[u]=0$. That is, $G(\xi,x)$ satisfies $L_x[G(\xi,x)]=0$, and also the initial conditions

$$\left.\frac{\partial^j G(\xi,x)}{\partial x^j}\right|_{x=\xi}=\delta_{n-1,j},\qquad j=0,1,\cdots,n-1. \tag{1.9}$$

In order to simplify some of our expressions it is useful for us to introduce the functions

$$\begin{aligned} p(x,y)&:=\sum_{k=0}^{n-1}a_{k+1}(y)\frac{(x-y)^k}{k!},\\ q(x,y)&:=-\sum_{k=0}^{m}b_k(y)\frac{(x-y)^{n-m+k-1}}{(n-m+k-1)!}. \end{aligned} \tag{1.10}$$

Let $P(x,y)$ be the resolvent kernel corresponding to the Volterra integral equation with kernel $p(x,y)$, i.e.,

$$\phi(x)+\int_\xi^x p(x,y)\phi(y)\,dy=\psi(x). \tag{1.11}$$

Then $s_{l+1}(x,\xi)$ may be represented in the form

$$s_{l+1}(x,\xi)=\int_\xi^x Q(x,y)s_l(y,\xi)\,dy,\qquad l=0,1,2,\cdots, \tag{1.12}$$

where

$$Q(x, y) := q(x, y) - \int_y^x q(z, y)P(x, z)\, dz.$$

For the purpose of analytical computations it would be useful to have a closed-form expression for $P(x, y)$, and hence $Q(x, y)$. With this in mind we note that the Riemann function may be written in the form

$$G(\xi, x) := \frac{(x-\xi)^{n-1}}{(n-1)!} - \int_\xi^x \frac{(z-\xi)^{n-1}}{(n-1)!} P(x, z)\, dz, \tag{1.13}$$

which may easily be verified by checking the initial conditions and observing that $G(\xi, x)$ is the solution of (1.11) when the nonhomogeneous term is given by

$$\psi := \frac{(x-y)^{n-1}}{(n-1)!}.$$

Along with the Riemann function we introduce a set of associated functions,

$$G_k(\xi, x) := \frac{(x-\xi)^k}{k!} - \int_\xi^x \frac{(z-\xi)^k}{k!} P(x, z)\, dz \tag{1.14}$$

$(k = 0, 1, 2, \cdots, n-1)$, $G_{n-1}(\xi, x) \equiv G(\xi, x)$. In terms of these functions $Q(x, y)$ may be represented as

$$Q(x, y) = -\sum_{k=0}^{m} b_k(y) G_{n-m+k-1}(y, x). \tag{1.15}$$

Formulae (1.12) and (1.15) indicate that iterated integrations involving products of the functions $G_k(x, y)$ must be performed. In practice, analytical calculations of this type would prove extremely tedious if not impossible to do by hand. It is with this obstacle in mind that we develop the next section, where it is shown how closed forms may be obtained for the coefficients $s_l(x, \xi)$ by making use of a symbolic and algebraic manipulation language. The language used here is an interactive and extended version [12] of FORMAC, based on the object-time library routines of FORMAC [13], [1].

2. Metaparabolic equations with constant coefficients. In this section we restrict our attention to the case where the coefficients a_k are constants. This leads to some simplification in the representation formulae, which for purposes of exposition is helpful. Indeed, we shall show that the associated Riemann function $G_k(\xi, x)$ may be uniquely determined from knowledge of the roots of the polynomial $x^n p(1/x, 0)$. Indeed from these roots we obtain a fundamental system of solutions for the differential equation satisfied by the resolvent kernel $P(x, y)$. The resolvent can then be determined from its initial data, which in turn, is related to the initial data satisfied by the given polynomial, $x^n p(1/x, 0)$, at $x = 0$.

We recall that $p(x, y)$ and $P(x, y)$ satisfy the identity

$$P(x, y) = p(x, y) - \int_y^x p(x, z)P(z, y)\, dz. \tag{2.1}$$

Since for the constant coefficient case $p(x, y)$, as defined by (1.10), must be a function of $(x-y)$ alone, (2.1) dictates that $P \equiv P(x-y)$ also. Hence (2.1) becomes

$$P(X) = p(X) - \int_0^X p(X-\sigma)P(\sigma)\, d\sigma. \tag{2.2}$$

By differentiating (2.2) with respect to X n times, it is seen that P satisfies an ordinary differential equation

$$P^{(n)}+a_1P^{(n-1)}+\cdots+a_{n-1}P'+a_nP=0. \tag{2.3}$$

Hence $P(X)$ must have the form

$$P(X)=\sum_{j=1}^{n} c_jX^{l_j-1}\,e^{R_jX}, \tag{2.4}$$

where R_j are the roots and l_j is the corresponding multiplicity of the algebraic equation

$$X^n+a_1X^{n-1}+\cdots+a_{n-1}X+a_n=0. \tag{2.5}$$

The coefficients c_j may be uniquely determined from the initial condition on $P^{(k)}(0)$, $(k=0,1,\cdots,n-1)$. Differentiating (2.4) successively we obtain

$$\sum_{j=1}^{n} c_jR_j^{k-l_j-1}\frac{k!}{\Gamma(k-l_j)}=P^{(k)}(0), \qquad k=0,1,\cdots,n-1. \tag{2.6}$$

The coefficient matrix of the equation system (2.6) may for any vectors R_j and l_j be generated by the FORMAC commands:

```
PROC COEFFICMAT:
/* Generate the coefficient matrix of (2.6)                  */
FNC(RPGAMMA) = STEP(0,$(1),10**25)/FAC(STEP(0,$(1),10**25)
*$(1));
DO K = 0 TO N - 1;
    DO J = 1 TO N;
    COEFMA(K,J) = R(J)**(K - L(J) - 1)*FAC(K)*RPGAMMA(K - L(J);
    END;
END;
PROC_END;
```

A relation between $P^{(k)}(0)$ and $p^{(k)}(0)$ is obtained by differentiating (2.2) with respect to X; namely, we obtain

$$\begin{gathered} P(0)=p(0),\\ P^{(k)}(0)=p^{(k)}(0)-\sum_{n=1}^{k} p^{(n-1)}(0)P^{(n-k)}(0), \qquad k=1,2,\cdots,n-1. \end{gathered} \tag{2.7}$$

From (1.10) we have $p^{(k)}(0)=a_{k+1}$, $(k=0,1,\cdots,n-1)$. Therefore by using (2.7) we have $P^{(k)}(0)=f(a_k,a_{k-1},\cdots,a_1)$. Since the a_k are easily obtained from the roots R_j, $P^{(k)}(0)$ may be expressed in terms of the roots. The FORMAC commands generating the expressions for $P^{(k)}(0)$ are for any n:

```
/* Generate the coefficients A(K) of (2.5)                    */
A(0) = 1; POL = 1;
DO J = 1 TO N; POL = POL*(X - R(J)); END;
POL = POL*X;
DO K = 1 TO N; A(K) = COEFF(POL,X**(N + 1 - K)); END;
PROC RIGHTHANDS:
/* Generate the right hand side of (3.6)                      */
PO(0) = A(1);
DO K = 1 TO N - 1;
```

```
PO(K)=A(K+1);
    DO I=1 TO K;
    PO(K)=PO(K)-A(I)*PO(K-I);
    END;
END;
PRO_END;
```

The results of the symbolic manipulation show that

$$(2.8)\quad P^{(k)}(0) = -\sum_{i_{k+1}=1}^{n} R_{i_{k+1}} \sum_{i_k=i_{k+1}}^{n} R_{i_k} \cdots \sum_{i_2=i_3}^{n} R_{i_2} \sum_{i_1=i_2}^{n} R_{i_1}, \qquad k=0,1,\cdots,n-1.$$

The determinant Δ of the coefficient matrix of (2.6) is given by

$$(2.9)\qquad \Delta = \prod_{j=1}^{M}\left\{\prod_{i=j+1}^{M}(r_i-r_j)^{\kappa_i\kappa_j}\right\}\left\{\prod_{i=1}^{\kappa_j-1}(\kappa_j-i)!\right\},$$

where r_j, $(j=1,2,\cdots,M)$ are the distinct roots of (2.5) and κ_j are their respective multiplicities.

The solutions c_k of the equation system (2.6) are given by

$$(2.10)\quad c_k = -r_i^{n+l_k-\kappa_i} f_k(n,M,r_\gamma,\kappa_\gamma) \Big/ \prod_{j=1, j\neq i}^{M} (r_i-r_j)^{\kappa_i+\kappa_j-l_k}, \qquad k=1,2,\cdots,n,$$

where the relations among k, i and l_k are given by

$$(2.11)\qquad k=\sum_{j=1}^{i-1}\kappa_j + l_k, \qquad i=1,2,\cdots,M, \quad l_k=1,2,\cdots,\kappa_i$$

and f_k are polynomials in r_γ.

If there are no multiple roots then $f_k=1$. In the case of one n-double root, f_k is given by

$$f_k=\binom{n}{k}\frac{1}{(k-1)!}.$$

In the case where r_i is the only double root,

$$f_1=\sum_{j=2}^{n} j r_1^{n-j} C(j),$$

where $C(\lambda)$ is defined as the coefficient of $x^{n-\lambda}$ in the product $\prod_{\gamma=2}^{n-1}(x-r_\gamma)$ and $f_k=1$ for $k\neq 1$. In the general case of several multiple roots f_k is more complicated.

From (1.14) we obtain the following expressions for the associated Riemann functions:

$$G_\nu(X)=\frac{X^\nu}{\nu!}\left\{1-X\int_0^1 \lambda^\nu P(X(1-\lambda))\,d\lambda\right\}, \qquad \nu=0,1,\cdots,n-1.$$

Using (2.4) we have

$$(2.12)\qquad G_\nu(X)=\frac{X^\nu}{\nu!}\left\{1-X\sum_{k=1}^{n} c_k X^{l_k-1} e^{R_kX}\int_0^1 \lambda^\nu(1-\lambda)^{l_k-1} e^{-R_kX\lambda}\,d\lambda\right\},$$

which after integrating by parts becomes

$$G_\nu(X)=\frac{X^\nu}{\nu!}\left\{1+\sum_{k=1}^{n} c_k R_k^{-l_k}(l_k-1)!(-1)^{l_k-1}\right\}$$

$$-\sum_{k=1}^{n} c_k e^{R_k X}\sum_{\gamma=0}^{l_k-1} R_k^{-(\gamma+\nu+1)}X^{l_k-1-\gamma}\frac{(l_k-1)!(\gamma+\nu)!}{\gamma!(l_k-1-\gamma)!\nu!}(-1)^\gamma$$

$$+\sum_{k=1}^{n} c_k \sum_{\gamma=1}^{\nu} R^{-(\gamma+l_k)}X^{\nu-\gamma}\frac{(\gamma+l_k-1)!}{\gamma!(\nu-\gamma)!}(-1)^{l_k-1},$$

or

$$G_\nu(X)=-\sum_{k=1}^{n} c_k e^{R_k X}\sum_{\gamma=0}^{l_k-1} R_k^{-(\gamma+\nu+1)}X^{l_k-1-\gamma}\frac{(l_k-1)!(\gamma+\nu)!}{\gamma!(l_k-1-\gamma)!\nu!}(-1)^\gamma. \tag{2.13}$$

By calculating the $G_\nu(X)$ using the FORMAC procedure SL, it may be seen that the $G_\nu(X)$ have the form

$$G_\nu(X)=\sum_{k=1}^{n} r_i^{n-\nu+l_k-\kappa_i-1}X^{l_k-1}\,e^{r_i X}\phi_k(n,M,r_\gamma,\kappa_\gamma,\nu)\prod_{j=1,j\neq i}^{M}(r_i-r_j)^{-\kappa_i-\kappa_j+l_k}. \tag{2.14}$$

Here the ϕ_k are polynomials in r_γ, and contain the factor $v_i^{-\lambda}$ when $\lambda=n-\nu+l_k-\kappa_i-1<0$. The dependence of k on i and l_k is given by (2.11).

If there are no multiple roots then $\phi_k=1$, and in the case of one n-tuple root ϕ_k is given by

$$\phi_k=\frac{(n-1-\nu)!}{(k-1)!(n-k)!\Gamma(k-\nu)};$$

when r_1 is the only double root we obtain the special form

$$\phi_1=\sum_{j=2}^{n}(j-1-\nu)r_1^{n-j}C(j),$$

with $\phi_k=1$ for all $k\neq 1$.

From (1.15) we have, in the case where the coefficients a_k are constants,

$$Q(x,y)=-\sum_{\nu=n-m-1}^{n-1} b_{\nu+m+1-n}(y)G_\nu(x-y). \tag{2.15}$$

Substituting $G_\nu(x-y)$ from (2.14) into (2.15) we obtain

$$Q(x,y)=-\sum_{k=1}^{n} r_i^{n+l_k-\kappa_i-1}(x-y)^{l_k-1}\,e^{r_i(x-y)}\prod_{\substack{j=1\\ j\neq i}}^{M}(r_i-r_j)^{-\kappa_i-\kappa_j+l_k}$$

$$\cdot\sum_{\nu=n-m-1}^{n-1} b_{\nu+m+1-n}(y)r_i^{-\nu}\phi_k(n,M,r_\gamma,\kappa_\gamma,\nu).$$

The $s_l(x,\xi)$ are calculated using the recursive scheme (1.12) where the initial coefficient is seen from (1.8) to be

$$s_0(x,\xi)=G(\xi,x)=G_{n-1}(\xi,x)=G_{n-1}(x-\xi). \tag{2.16}$$

If the coefficients b_k are constants, (1.12) yields with $X=x-\xi$

$$s_{l+1}(X)=\int_0^X Q(X-z)s_l(z)\,dz. \tag{2.17}$$

The calculation of the coefficients $s_l(x, \xi)$ given the coefficients a_k and b_k may be performed by the FORMAC procedure SL given below:

```
PROC SL;
/* Calculate the roots R(J) of (3.5) and                    */
/* the vector L(J)                                          */
CALL ROOTS;
/* Print_out the R(J) and L(J)                              */
DO II=1 TO N; P(R(II), L(II)); END;
/* Generate the coefficient matrix of (3.6)                 */
CALL COEFFICMAT;
/* Generate the right-hand side of (3.6)                    */
CALL RIGHTHANDS;
/* Change index by 1 in the coefficient matrix and the      */
/* right-hand side                                          */
DO K=N TO 1 BY -1;
    DO J=1 TO N;
    COEFMA(K,J)=COEFMA(K-1,J);
    END;
RIGHTS(K)=PO(K-1); END;
/* Solve the linear equation system (3.6). DET means        */
/* the determinant of the coefficient matrix                */
C=LIEQU(N,1,COEFMA,RIGHTS,DET):
DO K=1 TO N; C(K)=C(K)/DET; END;
/* Compute G(K) using (3.12)                                */
DO II=0 TO N-1;
/* F and H are variables for intermediate results           */
H=0;
    DO K=1 TO N;
    F=INTGR(T**II*(1-T)**(L(K)-1)*£E**(-R(K)*X*T),T);
    F=EVAL(F,T,1)-EVAL(F,T,0);
    H=H+C(K)*X**(L(K)-1)*£E**(R(K)*X)*F;
    END;
G(II)=X**II/FAC(II)*(1-X*H);
P(G(II)); /* Print_out G(II)                                */
END;
/* Compute Q using (3.15)                                   */
Q=0;
DO II=N-M-1 TO N-1;
Q=Q-B(II+M+1-N)*G(II);
END;
/* Define S(0) using (3.16)                                 */
S(0)=G(N-1);
/* Compute S(L) using (3.17)                                */
DO L=0 TO LMAX;
S(L+1)=INTGR(REPLACE(Q,X,X-Z)*REPLACE(S(L),X,Z),Z);
S(L+1)=EVAL(S(L+1),Z,X)-EVAL(S(L+1),Z,0); P(S(L+1));
END;
PROC_END;
```

As an example of use of the procedure SL we may calculate the first ten coefficients $s_l(x, \xi)$ in the case $n = 4$, $a_0 = 1$, $a_1 = -19$, $a_2 = 121$, $a_3 = -309$, $a_4 = 270$, $m = 2$, $b_0 = 1$, $b_1 = 2$, $b_2 = 5$.

So the following FORMAC commands:

```
N = 4; M = 2; LMAX = 8;
A(0) = 1; A(1) = −19; A(2) = 121;
A(3) = −309; A(4) = 270;
B(0) = 1; B(1) = 2; B(2) = 5;
CALL SL;
```

result in

```
R(1) = 2     R(2) = 3     R(3) = 5     R(4) = 9
L(1) = 1     L(2) = 1     L(3) = 1     L(4) = 1
```

$$G(0) = 243/56\ \text{£E}^{9\,X} - 125/24\ \text{£E}^{5\,X} + 9/4\ \text{£E}^{3\,X} - 8/21\ \text{£E}^{2\,X}$$

$$G(1) = 27/56\ \text{£E}^{9\,X} - 25/24\ \text{£E}^{5\,X} + 3/4\ \text{£E}^{3\,X} - 4/21\ \text{£E}^{2\,X}$$

$$G(2) = 3/56\ \text{£E}^{9\,X} - 5/24\ \text{£E}^{5\,X} + 1/4\ \text{£E}^{3\,X} - 2/21\ \text{£E}^{2\,X}$$

$$G(3) = 1/168\ \text{£E}^{9\,X} - 1/24\ \text{£E}^{5\,X} + 1/12\ \text{£E}^{3\,X} - 1/21\ \text{£E}^{2\,X}$$

$$\begin{aligned} S(1) = {} & -13/3528\ X\ \text{£E}^{9\,X} - 5/72\ X\ \text{£E}^{5\,X} - 5/36\ X\ \text{£E}^{3\,X} - 13/441\ X\ \text{£E}^{2\,X} \\ & + 253/74088\ \text{£E}^{9\,X} + 13/216\ \text{£E}^{5\,X} + 127\ \text{£E}^{3\,X} - 932/9261\ \text{£E}^{2\,X} \end{aligned}$$

$$\begin{aligned} S(5) = {} & -1/8233547616\ (371293/10\ X^{5} - 3113149/7\ X^{4} \\ & + 393867175/168\ X^{3} - 47538551645/7056\ X^{2} + 517466976247/49392\ X \\ & - 4846859564003/691488)\ \text{£E}^{9\,X} - 1/85766121\ (371293/120\ X^{5} \\ & + 3627247/21\ X^{4} + 255642920/63\ X^{3} + 66381192280/1323\ X^{2} \\ & + 112027632314/343\ X + 58041810204344/64827)\ \text{£E}^{2\,X} \\ & - 1/34992\ (625/2\ X^{5} + 138125/6\ X^{3} - 292625/9\ X^{2} + 11313175/36\ X \\ & - 2548219/9)\ \text{£E}^{3\,X} - 1/69984\ (625/2\ X^{5} - 3125\ X^{4} + 475625/24\ X^{3} \\ & - 10692875/144\ X^{2} + 7887775/48\ X - 47303981/288)\ \text{£E}^{5\,X} \end{aligned}$$

$$\begin{aligned} S(9) = {} & -(.22787383E-04\ X^{9} + .54689721E-04\ X^{8} + .01009754\ X^{7} \\ & - .01440624\ X^{6} + 1.43530921\ X^{5} - 4.82332329\ X^{4} + 76.978372\ X^{3} \end{aligned}$$

$$-246.409924\,X^{2}+1129.54984\,X-1478.40995)\,£E^{3\,X}$$

$$-(.11393691E-04\,X^{9}-.32130211E-03\,X^{8}+.00561686\,X^{7}$$

$$-.06703802\,X^{6}+.58885924\,X^{5}-3.83654649\,X^{4}+18.2915172\,X^{3}$$

$$-60.8249909\,X^{2}+126.974505\,X-125.934703)\,£E^{5\,X}$$

$$-(.17520003E-08\,X^{9}+.29826362E-06\,X^{8}+.23247604E-04\,X^{7}$$

$$+.00108914\,X^{6}+.03380729\,X^{5}+.72111122\,X^{4}+10.5703344\,X^{3}$$

$$+102.676988\,X^{2}+599.691297\,X+1604.34494)\,£E^{2\,X}-(.21900004E$$

$$-09\,X^{9}-.76168696E-08\,X^{8}+.12520753E-06\,X^{7}-.12716743E-05\,X^{6}$$

$$+.87641229E-05\,X^{5}-.42374496E-04\,X^{4}+.14335005E-03\,X^{3}$$

$$-.32640295E-03\,X^{2}+.45294103E-03\,X-.29128482E-03)\,£E^{9\,X}$$

Here for brevity s_2, s_3, s_4, s_6, s_7 and s_8 have been omitted and the coefficients of s_9 have been transformed from rational numbers into real numbers.

The results obtained for $s_l(x, \xi)$ may be verified by substituting the series for the fundamental solution S given by (1.5) into the differential equations (1.1)–(1.2). This is done in the procedure CONTROL:

```
PROC CONTROL;
/* Generate the fundamental solution S(x,t,z,tau) using          */
/* (2.5)                                                          */
S=0;
DO L=0 TO LMAX+1;
S=S+S(L)*FAC(L)*(-1)**(L+1)/(T-TAU)**(L+1); END;
/* Test the solution S using (2.1) and (2.2)                      */
NUL=0;
DO K=0 TO N; NUL=NUL+A(N-K)*DERIV(S,X,K); END;
ST=DERIV(S,T);
DO K=0 TO M; NUL=NUL+B(M-K)*DERIV(ST,X,K); END;
PROC_END;
```

3. Pseudoparabolic equations. We consider here the case of pseudoparabolic equations, namely, equations having the form

$$p[u]:=L[u_t]+M[u]=0, \tag{3.1}$$

where L and M are the operators defined by (1.2). As noted earlier, the fundamental singularity of a metaparabolic equation resembles that of a parabolic equation; indeed it is a perturbation of one. The pseudoparabolic equations on the other hand have singularities of quite a different nature; they resemble more closely those of an elliptic equation.

For the present development it is of interest to first investigate the singular solutions of the ordinary differential equation

$$L[u] := \sum_{k=0}^{n} \frac{\partial^k}{\partial x^k}(a_{n-k}u) = \delta(x-\xi), \tag{3.2}$$

where $x, \xi \in [0, l]$ and where $\delta(x-\xi)$ is a delta function having its source point at $x = \xi$. Given a fundamental family of solutions to $L[u]=0$ a fundamental singularity may be constructed in terms of the Wronskian $W(\phi_1, \phi_2, \cdots \phi_n)(x)$,as

$$K(x,\xi) := \frac{1}{a_n(\xi)W(\phi_1,\cdots,\phi_n)(\xi)} D(x,\xi), \qquad x \geqq \xi, \tag{3.3}$$

$$K(x,\xi) \equiv 0, \qquad x<\xi, \quad x, \xi \in [0, l],$$

where $D(x, \xi)$ is defined as

$$D(x,\xi) := \begin{vmatrix} \phi_1(\xi), \phi_2(\xi), \cdots, \phi_n(\xi) \\ \phi_1'(\xi), \phi_2'(\xi), \cdots, \phi_n'(\xi) \\ \vdots \\ \phi_1^{(n-1)}(\xi), \phi_2^{(n-1)}(\xi), \cdots, \phi_n^{(n-1)}(\xi) \\ \phi_1(x), \phi_2(x), \cdots, \phi_n(x) \end{vmatrix}.$$

If, in addition, we wish the fundamental singularity to satisfy the boundary conditions

$$\mathbf{U}_j[u] := \sum_{k=1}^{n} [M_{jk}u^{(k-1)}(0) + N_{jk}u^{(k-1)}(l)] = 0, \qquad j = 1, 2, \cdots, n, \tag{3.4}$$

this may be accomplished by adding to $K(x, \xi)$ a linear combination of the solutions $\{\phi_1, \cdots, \phi_n\}$ choosing the coefficient $c_j(\xi)$ such that

$$G(x,\xi) := K(x,\xi) + \sum_{j=1}^{n} c_j(\xi)\phi_j(x)$$

satisfies (3.4). When $\det |\mathbf{U}_k[\phi_j]| \neq 0$ then the system

$$\sum_{j=1}^{n} c_j(\xi)\mathbf{U}_k[\phi_j] = -\mathbf{U}_k[K](\xi), \qquad k = 1, 2, \cdots, n \tag{3.5}$$

may always be solved for the $c_j(\xi)$, $(j = 1, 2, \cdots, n)$.

For our purposes another way of constructing a fundamental singularity is actually more convenient. As before we consider the case of constant coefficients, taking $a_0 \equiv 1$.

A fundamental singularity is sought in the form

$$S(x,\xi) := \frac{1}{n}\theta(x-\xi)A(x,\xi) + B(x,\xi), \tag{3.6}$$

where

$$\theta(x-\xi) := \begin{cases} 1, & x>\xi, \\ 0, & x \leqq \xi \end{cases}$$

is the Heaviside function. Substituting this expression into (3.2), using the Leibnitz formula, and reordering the resulting summations yields

$$\frac{1}{n}\theta(x-\xi)L[A](x,\xi) + \frac{1}{n}\sum_{j=1}^{n} \delta^{(j-1)}(x-\xi) \sum_{k=j}^{n} \binom{k}{j} a_{n-k} \frac{\partial^{k-j}A}{\partial x^{k-j}} + L[B](x,\xi) = \delta(x-\xi). \tag{3.7}$$

If the a_{n-k} are constants, or analytic functions of x on $(-\infty, \infty)$, then $A(x, \xi)$ and $B(x, \xi)$ are locally analytic about each $x \in (-\infty, \infty)$; $A(z, \xi)$ and $B(z, \xi)$ could be continued, for example, by formal powers or other standard methods. As the Heaviside function, and the derivatives of δ-functions have the continuations [2, pp. 68–69]

$$(3.8)\qquad \begin{gathered} (x-\xi)^k\theta(x-\xi) \to -\frac{1}{2\pi i}(z-\xi)^k \log(\xi - z), \\ \delta^{(j-1)}(x-\xi) \to \frac{(-1)^j(j-1)!}{2\pi i(z-\xi)^j}, \end{gathered}$$

equation (3.7) has the continuation

$$-\frac{1}{n}L[A](z,\xi)\log(\xi-z) + \sum_{j=1}^{n}\frac{(-1)^j(j-1)!}{n(z-\xi)^j}\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A(z,\xi)}{\partial x^{k-j}}$$
$$+L[B](z,\xi) + \frac{1}{z-\xi} \equiv 0.$$

The singularity due to the logarithm gives rise to a multivalued function, and this singularity cannot be cancelled out by the polar singularities due to the terms $(z-\xi)^{-j}$. Hence, in order for the right-hand side to be identically equal to zero we must have

$$(3.9)\qquad L_x[A] = 0,$$

i.e., $A(z, \xi)$ is a solution of the homogeneous equations. In order that the polar singularities cancel we must impose initial conditions on the $A(x, \xi)$; these are given by

$$\frac{1}{n}\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A(z,\xi)}{\partial x^{k-j}} = \delta_{1,j} \quad \text{at } x=\xi, \quad j=1,2,\cdots,n.$$

The conditions may be satisfied by requiring that

$$(3.10)\qquad \left.\frac{\partial^j A(x,\xi)}{\partial x^j}\right|_{x=\xi} = \delta_{n-1,j}, \qquad j=0,1,\cdots,n-1.$$

Comparing (3.9), (3.10) with the equations for $G(\xi, x)$ as given by (1.8), (1.6), and (1.9), we notice that $A(x,\xi) \equiv G(\xi, x)$; that is, A is the Riemann function associated with the operator L. The function $B(x, \xi)$ is seen to satisfy

$$L[B](x,\xi) = \delta(x-\xi) - \frac{1}{n}\sum_{j=1}^{n}\delta^{(j-1)}(x-\xi)\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A}{\partial x^{k-j}}.$$

However, since the δ-functions are cancelled by zeros of the coefficients etc. we might continue this into the whole complex plane and solve it analytically.

Having obtained this representation for a fundamental singularity of (3.2) we seek a singular solution to (3.1) in the form

$$(3.11)\qquad S(x,t;\xi,\tau) := \frac{1}{n}\theta(x-\xi)A(x,t;\xi,\tau) + B(x,t;\xi,\tau).$$

Substituting this into (3.1), we obtain

$$(3.12)\qquad \begin{gathered} \frac{1}{n}\theta(x-\xi)L[A_t] + \frac{1}{n}\sum_{j=1}^{n}\delta^{(j-1)}(x-\xi)\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A_t}{\partial x^{k-j}} + L[B_t] + \frac{1}{n}\theta(x-\xi)M[A] \\ + \frac{1}{n}\sum_{j=1}^{n-1}\delta^{(j-1)}(x-\xi)\sum_{k=j}^{n-1}\binom{k}{j}b_{n-k-1}\frac{\partial^{k-j}A}{\partial x^{k-j}} + M[B] = \delta(x-\xi). \end{gathered}$$

Assuming that the coefficients A and B have the form

$$A(x,t;\xi,\tau):=\sum_{j=0}^{\infty} A_j(x,\xi)\frac{(t-\tau)^{j+1}}{(j+1)!},$$
$$B(x,t;\xi,\tau):=\sum_{j=0}^{\infty} B_j(x,\xi)\frac{(t-\tau)^{j+1}}{(j+1)!}, \tag{3.13}$$

we obtain equations for the A_j, B_j, by substituting these expressions into (3.12) and collecting like powers of $(t-\tau)$. The zeroth order term is

$$\frac{1}{n}\theta(x-\xi)L[A_0]+\frac{1}{n}\sum_{j=1}^{n}\delta^{(j-1)}(x-\xi)\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A_0}{\partial x^{k-j}}+L[B_0]=\delta(x-\xi). \tag{3.14}$$

As before, this implies $A_0(x,\xi)=G(\xi,x)$, $B_0(x,\xi)$ may be found as any solution of the resulting equation having set $L[A_0]\equiv 0$. The terms of order l are seen to be

$$\frac{1}{n}\theta(x-\xi)L[A_l]+\frac{1}{n}\sum_{j=1}^{n}\delta^{(j-1)}(x-\xi)\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A_l}{\partial x^{k-j}}+L[B_l]+\frac{1}{n}\theta(x-\xi)M[A_{l-1}]$$
$$+\frac{1}{n}\sum_{j=1}^{n-1}\delta^{(j-1)}(x-\xi)\sum_{k=j}^{n-1}\binom{k}{j}b_{n-k-1}\frac{\partial^{k-j}A_{l-1}}{\partial x^{k-j}}+M[B_{l-1}]=0. \tag{3.15}$$

Equating the coefficients of $\theta(x-\xi)$ identically to zero yields the recursive relation for the A_l $(l=1,2,\cdots)$,

$$L[A_l]=-M[A_{l-1}], \tag{3.16}$$

whereas the initial conditions are arrived at by requiring that the coefficients of the individual $\delta^{(j)}(x-\xi)$, $(j=0,1,\cdots,n-1)$ vanish at $x=\xi$. They are given by

$$\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A_l}{\partial x^{k-j}}+\sum_{k=j}^{n-1}\binom{k}{j}b_{n-k-1}\frac{\partial^{k-j}A_{l-1}}{\partial x^{k-j}}=0 \tag{3.17}$$
$$\text{at } x=\xi,\quad j=0,1,\cdots,n-1.$$

We notice that, when $l=1$, the

$$A_1^{(j)}:=\frac{\partial^j A_1}{\partial x^j},\qquad j=0,1,\cdots,n-1$$

are related to $A_0, A_0', \cdots, A_0^{(n-2)}$ at $x=\xi$. Since these derivatives all vanish at $x=\xi$, (3.17) implies for $l=1$ that

$$\sum_{k=j}^{n}\binom{k}{j}a_{n-k}\frac{\partial^{k-j}A_1}{\partial x^{k-j}}=0,\qquad j=0,1,\cdots,n-1. \tag{3.18}$$

As $a_0=1$, the system (3.18) may be uniquely solved by setting

$$\frac{\partial^j A_1}{\partial x^j}=0\quad \text{at } x=\xi,\qquad j=0,1,\cdots,n-1.$$

In a similar manner one verifies inductively that for $l=1,2,\cdots$

$$\frac{\partial^j A_l}{\partial x^j}=0\quad \text{at } x=\xi,\qquad j=0,1,\cdots,n-1. \tag{3.19}$$

The B_l may be chosen as any solution to the equation

$$L[B_l]=-F_l(x,\xi),$$

with

$$(3.20)\quad \begin{aligned} F_l(x,\xi) &:= \frac{1}{n}\sum_{j=1}^{n} \delta^{(j-1)}(x-\xi)\sum_{k=j}^{n}\binom{k}{j} a_{n-k}\frac{\partial^{k-j}A_l}{\partial x^{k-j}} \\ &+\frac{1}{n}\sum_{j=1}^{n-1} \delta^{(j-1)}(x-\xi)\sum_{k=j}^{n-1}\binom{k}{j} b_{n-k-1}\frac{\partial^{k-j}A_{l-1}}{\partial x^{k-j}}+M[B_{l-1}]. \end{aligned}$$

It is interesting to note at this point that the coefficients A_l computed here for the pseudoparabolic system are exactly the same ones we computed for the metaparabolic system; hence, the program given earlier has a further application to our present case.

That we obtain the same recursive system is not at all surprising, as the Laplace transform of the fundamental solution of the pseudoparabolic equation is the fundamental solution for the transformed (elliptic) partial differential

$$pL[u]+M[u]=\delta(x),$$

where p is the transformation parameter. The fundamental singularity of this equation may then be sought in the form

$$\hat{S}(x,\xi):=\frac{1}{n}\,\theta(x-\xi)A(x,\xi,p)+B(x,\xi,p).$$

$A(x,\xi,p)$ is the coefficient of the normalized singular term, hence it must be the equation's Riemann function. Now the coefficients $A(x,\xi,p)$ and $B(x,\xi,p)$ also depend upon the parameter p, and if we expand $A(x,\xi,p)$ in negative powers of p this gives us the above recursive system.

Before concluding this section we remark that it would have been possible to construct a fundamental singularity in the form

$$(3.21)\quad S(x,t;\xi,\tau):=\sum_{l=0}^{\infty} K_l(x,\xi)\frac{(t-\tau)^{l+1}}{(l+1)!},$$

where $K_0(x,\xi):=K(x,\xi)$ as given in (3.3). We then obtain the recursion formulae

$$(3.22)\quad L_x[K_{l+1}(x,\xi)]=-M[K_l(x,\xi)]$$

which integrate to

$$\begin{aligned} K_{l+1}(x,\xi) &= -\int_0^x M[K_l(\eta,\xi)]K_0(x,\eta)\,d\eta \\ &= \int_0^x \sum_{k=0}^{m}(-1)^k(b_{m-k}K_0(x,\eta))K_l(\eta,\xi)\,d\eta \\ &+\left[\sum_{p=1}^{m}\sum_{\substack{j+i=p-1\\ j,i\geqq 0}}(-1)^j K_l^{(i)}(\eta,\xi)b_{m-p}(K_0(x,\eta))^{(j)}\right]_0^x. \end{aligned}$$

We can normalize the terms $K_l(x,\xi)$, $l\geqq 1$ such that

$$\left.\frac{\partial^k K_l(x,\xi)}{\partial x^k}\right|_{x=0}=0,\qquad k=0,1,2,\cdots,n-1.$$

However, we must compute the derivatives

$$\frac{\partial^j}{\partial\eta^j}K_0(x,\eta)\quad \text{at } \eta=x,\quad j=0,1,\cdots,n-1.$$

CLAIM. $K_0^{(j)}(x, x^-) = 0$ *for* $j = 0, 1, \cdots, n-2$, *and*

$$K_0^{(n-1)}(x, x^-) = \frac{(-1)^{n-1}}{b_0}.$$

This may be seen by differentiating the determinant $D(x, \xi)$ by rows and noting that whenever two rows coincide as $x \to x^-$ this term does not contribute to the evaluation.

We obtain finally that

$$K_{l+1}(x, \xi) = -K_l(x, \xi) - \int_0^x M_\eta^*[K_0(x, \eta)]K_l(\eta, \xi)\,d\eta.$$

It is easy at this point to construct a fundamental singularity satisfying the boundary conditions (3.4). Defining

$$G_l(x, \xi) := K_l(x, \xi) + \sum_{j=1}^n C_{lj}(\xi)\phi_j(x)$$

and substituting this into (3.4) yields the condition (3.5) for each $l = 0, 1, 2, \cdots$, namely

$$\sum_{j=1}^n C_{lj}(\xi)\mathbf{U}_k[\phi_j] = -\mathbf{U}_k[K_l](\xi), \qquad k = 1, \cdots, n.$$

The generalized Green's function is then given by

$$G(x, t; \xi, \tau) := \sum_{k=0}^\infty G_k(x, \xi)\frac{(t-\tau)^{k+1}}{(k+1)!}.$$

We notice that $G_0(x, \xi) \equiv G(\xi, x)$ is the Green's function associated with L. Alternatively, the successively defined $G_k(x, \xi)$ might have been defined by the recursive scheme

$$G_{k+1}(x, \xi) = -\int_0^x M_\eta[G_k(\eta, \xi)]G(x, \eta)\,d\eta, \qquad k = 0, 1, \cdots.$$

REFERENCES

[1] K. A. BAHR, FORMAC73 *User's Manual*, GMD/IFV, Darmstadt, 1978.

[2] H. BREMERMANN, *Distributions, Complex Variables, and Fourier Transforms*, Addison-Wesley, Reading, MA, 1965.

[3] P. M. BROWN, R. P. GILBERT AND G. C. HSIAO, *Constructive function theoretic methods for fourth order pseudoparabolic equations in two space variables*, Rendiconti di Matematica, 8 (1975), pp. 935–951.

[4] D. COLTON, *Pseudoparabolic equations in one space variable*, J. Differential Equations, 12 (1972), pp. 559–565.

[5] ———, *On the analytic theory of pseudoparabolic equations*, Quart. J. Math., 23 (1972), pp. 179–192.

[6] ———, *Integral operators and the first initial boundary value problem for pseudoparabolic equations with analytic coefficients*, J. Differential Equations, 13 (1973), pp. 506–522.

[7] R. P. GILBERT, *A Lewy-type reflection principle for pseudoparabolic equations*, J. Differential Equations, 37 (1981), pp. 261–284.

[8] R. P. GILBERT AND G. C. HSIAO, *Constructive function theoretic methods for higher order pseudoparabolic equations*, Lecture Notes in Mathematics, 561, Springer, Berlin, 1977.

[9] R. P. GILBERT AND G. ROACH, *Constructive methods for meta and pseudoparabolic systems*, Bull. Math. Soc. Sci. Math. R. S. Roumanie, 68 (1976), pp. 97–108.

[10] R. P. GILBERT AND M. SCHNEIDER, *Generalized meta and pseudoparabolic equations in the plane*, in the Moscow Academy of Sciences Volume Honoring I. N. Vekua, Nauka, Moscow, 1978.

[11] ———, *On a class of boundary value problems for a composite system of first order differential equations*, J. Approx. Theory, 25 (1979), pp. 105–119.

[12] J. Jensen, TENSOR FORMAC—INTERAKTIVT FORMAC, Technical University of Denmark, 1978.

[13] R. G. Tobey et al., PL/1-FORMAC *symbolic mathematics interpreter*, SHARE Contributed Program Library, 360D-03.3.004, 1969.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

STEADY SHOCK TRACKING, NEWTON'S METHOD, AND THE SUPERSONIC BLUNT BODY PROBLEM*

G. R. SHUBIN†‡, A. B. STEPHENS†, H. M. GLAZ†, A. B. WARDLAW†
AND L. B. HACKERMAN†

Abstract. The steady shock tracking method, which combines shock tracking and Newton's method, is applied to the axisymmetric supersonic blunt body problem. The formulation is in conservation form and uses the constant total enthalpy condition to reduce the number of unknowns at each finite difference mesh point. On a transformed computational grid where the bow shock is a coordinate line, the discrete physical shock locations appear explicitly as unknowns in a set of finite difference equations which couples them to the other unknowns. The space-time characteristic compatibility conditions for the associated time-dependent problem are used in formulating the boundary conditions for the steady problem. The resulting system is solved using various modifications of Newton's method. Experiments are repeated with three linear system solvers whose efficiency is compared. The computed results for flow over a sphere are economically obtained and agree well with experiment. Continuation of the solutions with respect to some physical parameters is explored, and multiple solutions of one variation of our finite difference system are displayed.

Key words. blunt body flow, Newton's method, shock tracking, multiple solutions, continuation

1. Introduction. In [1] we presented an accurate and efficient method called "steady shock tracking" for solving certain inviscid steady flow problems with shocks, and applied this method to one-dimensional duct flow. This approach combines the desirable aspects of shock tracking (also known as shock fitting) with the rapid convergence of Newton's method in a way that is easy to implement. Recently, several papers have appeared concerning the application of Newton's method to fluid dynamics problems [2], [3], [4], with that of Gustafsson and Wahlund [4] also combining Newton's method with shock tracking for the supersonic blunt body problem. We have independently extended our steady shock tracking method to handle the blunt body problem. Our approach differs considerably from that of [4].

The physical problem is described in § 2A; for details see [5]. The most common technique for obtaining steady solutions of such problems is the "time-asymptotic" method wherein the time-dependent equations of fluid dynamics are used to proceed from some initial guess of the flow field to a steady state. This method tends to reduce errors in the initial guess rapidly at first, but then converges rather slowly, often with waves propagating through the computational domain. Explicit finite difference schemes require little work per time step but have stability restrictions which limit the time step size. Implicit schemes are often stable regardless of time step (in practice this step size is limited [6]), but require more work per step. In either case, the time asymptotic process can be inefficient if the criterion for achieving a steady state is somewhat exacting.

For approximating the shock wave one must choose either "shock capturing" or "shock tracking" methods. Shock capturing methods which do nothing special at shocks are very easy to implement, but they smear the computed shock over several finite difference mesh points, or they introduce computational oscillations. Alternatively, shock tracking methods which explicitly treat the shock as a moving free surface

* Received by the editors October 9, 1980, and in final revised form May 27, 1981. This work was funded by NSWC Independent Research funds and by the Reentry Technology Program Office.

† Applied Mathematics Branch (R44), Naval Surface Weapons Center, Silver Spring, Maryland 20910.
‡ Current address: Exxon Production Research Co., P.O. Box 2189, Houston, Texas 77001.

give excellent results at the shock but are relatively complicated to implement. Moretti [7] applied his explicit time-asymptotic shock tracking procedure to the blunt body problem and obtained results which agree well with experiment. Since then, the time asymptotic shock tracking method with either explicit or implicit interior finite differencing (but always an explicit shock treatment) has become a standard technique. Various modifications have also been devised to handle a wider variety of body configurations (e.g. [8], [9]) and to accurately treat low free stream Mach numbers [10]. Finally, a very different approach to the blunt body problem is to iteratively solve the "inverse problem" (specify the shock shape and compute the body shape until the desired body is obtained). This procedure can yield useful results but is ill-conditioned [11].

In steady shock tracking we first consider an unsteady shock tracking formulation in which a time-dependent transformation maps the unknown shock location into a constant computational coordinate and the physical region of interest into a rectangle. We then drop all time derivative terms to obtain a time-independent system of differential equations governing the steady flow problem. Discretizing these differential equations in computational coordinates yields a system of nonlinear difference equations (with the discrete shock locations appearing explicitly as unknowns) which is solved using a modified Newton's method.

In § 2 we present the finite difference computational method, discussing the continuous problem, the discrete problem, boundary conditions and the solution by Newton's method. We also consider continuation of the solution with respect to several parameters, and the possibility of multiple solutions or no solution. In § 3, we give computational results for a sphere showing good agreement with experimental data. We also present timing comparisions for some overall computational strategies employing three different linear system solvers, and exhibit multiple solutions to our nonlinear system. We use continuation to proceed from the solution for the sphere to a solution for an ellipse at a different Mach number.

2. Problem formulation.

2A. Continuous problem. The physical problem is depicted in Fig. 1a. An incoming supersonic stream with Mach number $M_\infty > 1$ and known density ρ_∞ and pressure p_∞ impinges on an axisymmetric blunt body and a bow shock forms separating the undisturbed flow to the left of the shock from the disturbed "shock layer" between the body and the shock. We want to determine the steady flow in the shock layer and the position of the bow shock. This steady flow is of mixed type, the governing equations being elliptic in the subsonic region (Mach number $M < 1$) and hyperbolic in the supersonic region ($M > 1$). For indented body shapes, embedded shocks may appear in the shock layer; here we consider only sufficiently smooth bodies for which this does not occur.

We first introduce the unsteady problem described in spherical coordinates (r, θ, ϕ) with axisymmetry assumed $(\partial/\partial\phi \to 0)$. See Fig. 1a. This time-dependent problem is hyperbolic with physical time being the time-like direction. The flow is described by

$$U_t + F_r + G_\theta + H = 0, \tag{1}$$

where

$$U = r^2 \sin\theta \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{bmatrix}, \qquad F = r^2 \sin\theta \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ u(\rho E + p) \end{bmatrix},$$

$$G = r\sin\theta \begin{bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(\rho E + p) \end{bmatrix}, \qquad H = \begin{bmatrix} 0 \\ -(2p+\rho v^2) r \sin\theta \\ -r(p\cos\theta - \rho uv \sin\theta) \\ 0 \end{bmatrix};$$

ρ is density, u is r-velocity, v is θ-velocity, $E = e + (u^2 + v^2)/2$ where e is specific internal energy and p is pressure (subscripts t, r, θ indicate partial differentiation). For simplicity, the equation of state is taken here to be that of a perfect gas

(2) $$p = (\gamma - 1)\rho e,$$

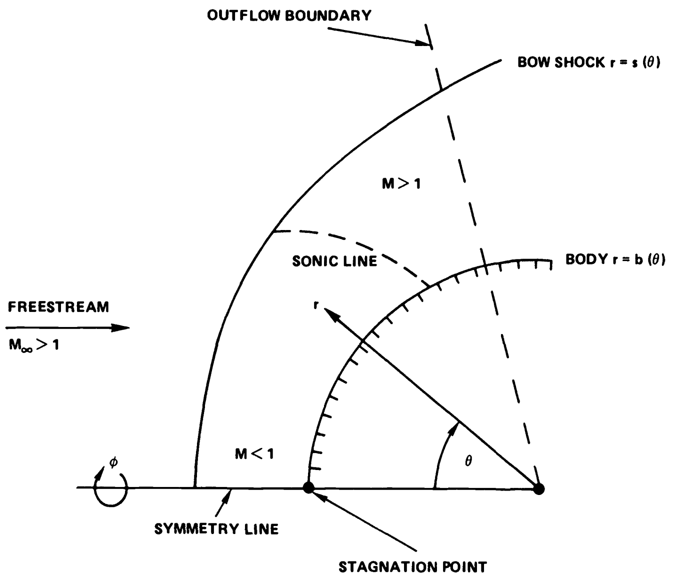


FIG. 1a. *Physical problem in axisymmetric spherical coordinates.*

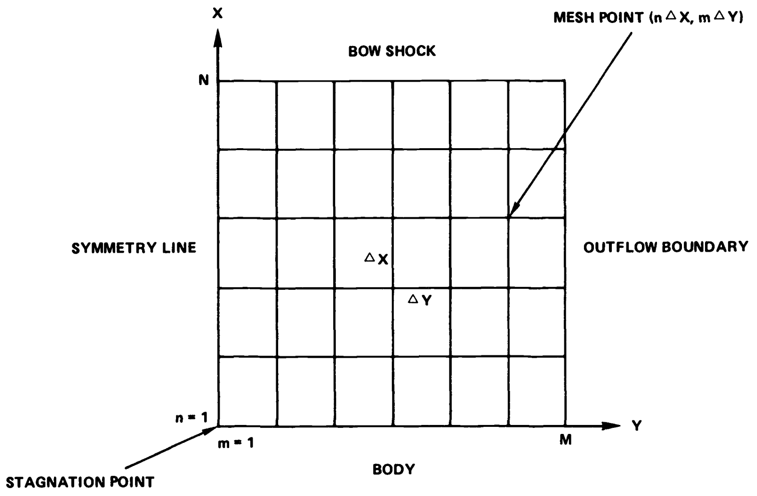


FIG. 1b. *Computational plane.*

where $\gamma = 1.4$ for air. We shall continue to use the symbol p with the understanding that it is to be replaced by (2).

In the steady state ($U_t = 0$) the energy conservation equation (component 4 of (1)) can be replaced by the condition of constant total enthalpy

$$\Omega = e + \frac{p}{\rho} + \frac{u^2+v^2}{2} = \gamma e + \frac{u^2+v^2}{2} = \Omega_\infty, \tag{3}$$

where Ω_∞ is evaluated at the known freestream conditions (i.e., Ω equals the same constant both inside and outside the shock layer). This simplification reduces the number of unknowns from four (ρ, u, v, e) to three (ρ, u, v). We note that (1) is identically satisfied on $\theta = 0$ by virtue of the factor $\sin\theta$. We have found this to be unsatisfactory in computation as it seems to "decouple" the flow on $\theta = 0$ from that at $\theta > 0$. For this reason we rewrite the steady state version of (1) as

$$\bar{F}_r + \bar{G}_\theta + \bar{H} = 0, \tag{4}$$

where

$$\bar{F} = r^2 \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \end{pmatrix}, \quad \bar{G} = r \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \end{pmatrix}, \quad \bar{H} = r \begin{pmatrix} \rho v \cot\theta \\ \rho uv \cot\theta - 2p - \rho v^2 \\ \rho v^2 \cot\theta + \rho uv \end{pmatrix}.$$

Equations (2), (3) and (4), together with boundary conditions to be specified in § 2B, comprise the continuous problem which is to be approximated. Our choice of the weak conservation form (4) was made in preference to the "non-conservation form" (e.g., [4], [7]) in anticipation of extending this work to flow fields with captured internal discontinuities such as embedded shocks.

Defining the known body as $r = b(\theta)$, the unknown shock as $r = s(\theta)$, and the outflow boundary as $\theta = \theta_{\max}$, the transformation [7]

$$X = \frac{r - b(\theta)}{s(\theta) - b(\theta)}, \qquad Y = \frac{\theta}{\theta_{\max}} \tag{5}$$

maps the physical shock layer of Fig. 1a into the computational rectangle of Fig. 1b. Equation (4) transforms to

$$[J(X_r\bar{F} + X_\theta\bar{G})]_X + [JY_\theta\bar{G}]_Y + J\bar{H} = 0, \tag{6}$$

where $J = \partial(r, \theta)/\partial(X, Y) = \theta_{\max}(s - b)$ is the Jacobian of the inverse transformation [12], and subscripts X, Y indicate partial differentiation.

2B. Boundary conditions. Computational boundary conditions must be specified at each point along the boundary of the computational rectangle, namely along the symmetry line $Y = 0$, the outflow boundary $Y = 1$, the body $X = 0$, and the shock $X = 1$. At the symmetry line, $v = 0$ and the unknowns ρ, u are even functions of θ. The third component of (4) or (6) is thus identically satisfied. Using L'Hospital's rule to evaluate $\lim_{\theta\to 0} \bar{H}$ and combining with G_θ gives the first two components of

$$\bar{F}_r + \tilde{H} = 0, \qquad \tilde{H} = \begin{pmatrix} 2r\rho v_\theta \\ 2r\rho u v_\theta - 2rp \end{pmatrix} \tag{7}$$

as two equations holding on $\theta = 0$, to be solved for the two unknowns ρ and u. Transforming using (5) gives

$$[JX_r\bar{F}]_X + J\tilde{H} = 0 \tag{8}$$

as the equations which govern the flow on $Y = 0$ in computational space.

For the other three boundaries, our approach to deriving computational boundary conditions makes use of the time-dependent formulation, especially the theory of characteristics for hyperbolic systems. From our point of view, the steady state solution is just a special kind of unsteady solution, and the concept of information being propagated along space-time characteristics is retained. From this perspective, the characteristic compatibility equations which hold along characteristics in the time-dependent problem still hold in the steady state, only they transmit the particular information that the solution is not changing in time. This would be the situation, for example, when a steady state is reached in a time-dependent method characteristics solution.

The characteristic ray directions and characteristic compatibility conditions for the time dependent problem (which has four unknowns since Ω is not constant) are derived in the Appendix. Considering the body and the shock, the qualitative nature of these characteristics, as projected on a $Y=\text{constant}$ plane, is depicted in Fig. 2. At the body, one characteristic (shown dashed) comes from outside the computational interval $0 \leqq X \leqq 1$, and is replaced by the physical boundary condition that the normal velocity at the wall is zero:

$$u - \frac{vb_\theta}{b} = 0. \tag{9}$$

This leaves three admissible compatibility conditions (i.e., those associated with characteristic rays reaching the boundary from inside the computational domain). As outlined in the Appendix, it is possible to identify one of these admissible conditions as being identically satisfied in the steady state due to the simplification of constant total enthalpy; this condition is then discarded. The two remaining relations holding at the wall are (dropping time derivative terms)

$$-p\rho_Y + \rho^2 e_Y = 0, \tag{10}$$

$$\begin{aligned} &-X_r D_b p_X + \rho c X_r u_X + \frac{\rho c X_\theta}{b} v_X + \frac{Y_\theta}{b}\left(\frac{v}{c} + \frac{b_\theta}{bD_b}\right) p_Y \\ &\quad - \frac{\rho v Y_\theta}{bD_b} u_Y + \frac{\rho Y_\theta}{b}\left(c + \frac{b_\theta v}{bD_b}\right) v_Y + \frac{\rho}{b}(c(2u + v\cot\theta) + vq_{\tan}) = 0, \end{aligned} \tag{11}$$

where $D_b^2 = 1 + (b_\theta/b)^2$, $q_{\tan} = (ub_\theta/b + v)/D_b$, and the speed of sound $c = (\gamma p/\rho)^{1/2}$.

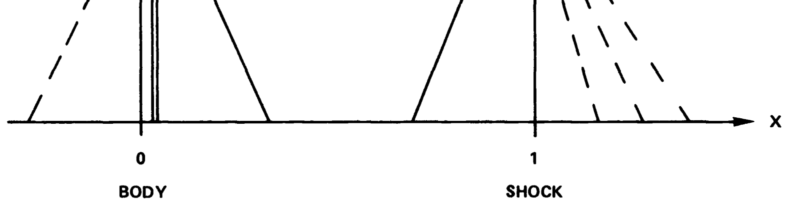


FIG. 2. *Behavior of characteristics at the body and the shock, projected on $Y=\text{constant}$ plane. Solid characteristics are admissible, dashed characteristics are inadmissible.*

Defining the entropy as $S = \ln(p/p_\infty) - \gamma \ln(\rho/\rho_\infty)$ where p_∞ and ρ_∞ are the freestream pressure and density, (10) can be rewritten as $S_Y = 0$ or

$$S = \text{constant}. \tag{12}$$

Summarizing, three independent conditions holding on the body $r = b(\theta)$ are (9), (11) and (12).

At the shock $r = s(\theta)$, one characteristic compatibility condition is admissible and three are not (Fig. 2). The admissible condition is (dropping time derivative terms)

$$\begin{aligned}\left(\frac{\bar{A}}{c} + X_r D_s\right) p_X + \rho\left(cX_r + \frac{\bar{A}}{D_s}\right) u_X + \frac{\rho}{s}\left(cX_\theta - s_\theta \frac{\bar{A}}{D_s}\right) v_X + \frac{Y_\theta}{s}\left(\frac{v}{c} - \frac{s_\theta}{sD_s}\right) p_Y \\ + \frac{\rho v Y_\theta}{sD_s} u_Y + \frac{\rho Y_\theta}{s}\left(c - \frac{s_\theta v}{sD_s}\right) v_Y - \frac{\rho}{s}[c(2u + v \cot \theta) + vQ_{\tan}] = 0,\end{aligned} \tag{13}$$

where $D_s^2 = 1 + (s_\theta/s)^2$, $Q_{\tan} = (us_\theta/s + v)/D_s$, and $\bar{A} = uX_r + vX_\theta/s$. The boundary conditions holding at the shock are the Rankine–Hugoniot jump conditions

$$\rho Q_n = \text{constant across shock}, \tag{14a}$$

$$p + \rho Q_n^2 = \text{constant}, \tag{14b}$$

$$e + \frac{p}{\rho} + \frac{Q_n^2}{2} = \text{constant}, \tag{14c}$$

$$Q_{\tan} = \text{constant}. \tag{14d}$$

Here $Q_n = (u - vs_\theta/s)/D_s$ is the velocity normal to the shock, and the constants are evaluated at some particular values of s, s_θ by using the known freestream conditions. Equation (14c) is equivalent to the statement of constant total enthalpy, (3), and is therefore not an additional condition. Summarizing, the independent conditions holding at the shock are (13), (14a), (14b), and (14d). Note that there are four conditions at the shock, whereas only three were needed at the body. The "extra" shock equation serves to determine the unknown shock location $r = s(\theta)$. This matter is discussed more fully in [1]. Gustafsson and Wahlund [4] closed their system in a different manner.

The outflow plane $Y = 1$ is assumed to be in the supersonic region (Fig. 1a). In this case, all four compatibility conditions are admissible, and no physical boundary conditions are required. The four conditions are, of course, equivalent to the original differential equations and we therefore use (6) in the computational space for determining the flow on $Y = 1$.

Finally, the point $X = 0$, $Y = 0$ (i.e., $r = b$, $\theta = 0$) is a stagnation point where $u = v = 0$. Additionally, the density and specific energy attain the stagnation values corresponding to a steady normal shock with upstream Mach number M_∞. Specifically, these stagnation values are obtained by solving (14) with $s_\theta = 0$ for the flow values behind the shock, and using (3) and the fact that the entropy S has the same value at the shock and at the stagnation point.

2C. Finite difference equations. A uniform mesh is used in the computational rectangle. The mesh points (X_n, Y_m) are given by $X_n = (n-1)\Delta X$, $n = 1, 2, \cdots, N$ with $\Delta X = 1/(N-1)$ and $Y_m = (m-1)\Delta Y$, $m = 1, 2, \cdots, M$ with $\Delta Y = 1/(M-1)$. The unknowns at each mesh point are $\rho_{n,m}$, $u_{n,m}$ and $v_{n,m}$ where $\rho_{n,m}$ represents $\rho(X_n, Y_m)$ and similarly for u and v. The specific internal energy e is defined in terms of these unknowns by the total enthalpy relation (3), and the pressure is then given by the equation of state (2).

Second order accurate differencing is used at "interior" points; both first and second order accurate approximations are considered at the body, shock, and outflow boundaries. At interior mesh points $n=2,3,\cdots,N-1$ and $m=2,3,\cdots,M-1$, (6) are approximated by the centered difference equations

$$\frac{\mathcal{F}_{n+1,m}-\mathcal{F}_{n-1,m}}{2\Delta X}+\frac{\mathcal{G}_{n,m+1}-\mathcal{G}_{n,m-1}}{2\Delta Y}+\mathcal{H}_{n,m}=0, \tag{15}$$

where $\mathcal{F}=J(X_r\bar{F}+X_\theta\bar{G})$, $\mathcal{G}=JY_\theta\bar{G}$, and $\mathcal{H}=J\bar{H}$.

On the symmetry line, (8) is approximated by

$$\frac{\mathcal{F}_{n+1,m}-\mathcal{F}_{n-1,m}}{2\Delta X}+\tilde{\mathcal{H}}_{n,m}=0, \tag{16}$$

where v_θ in (7) is given by $(Y_\theta)_{n,1}v_{n,2}/\Delta Y$ (this is second order by virtue of the oddness of v about $\theta=0$).

At the body $n=1$, $m=2,3,\cdots,M$, (9), (11) and (12) hold. Equation (9) is an algebraic equation where b and b_θ are given functions defining the body shape which are evaluated at the discrete body points. In (11), X-derivatives are approximated by a one-sided two-point (first order) formula, e.g., $p_X \sim (p_{2,m}-p_{1,m})/\Delta X$, or a three-point (second order) approximation, e.g., $p_X \sim (-3p_{1,m}+4p_{2,m}-p_{3,m})/(2\Delta X)$. Y-derivatives are approximated by centered differences along $X=0$ except at the outflow body point $n=1$, $m=M$ where two or three point backward Y-differences are used. All other terms are evaluated at the body point itself. At the shock, the Rankine–Hugoniot equations (14a, b, d) are algebraic equations holding at the shock points $n=N$, $m=1,2,\cdots,M$, where s_θ remains to be defined as finite differences of s. The terms in compatibility equation (13) are evaluated in the same manner as those in body equation (11), with the simplification on $m=1$ that $v=u_Y=p_Y=s_\theta=X_\theta=0$. The differencing of v_Y is the same as for the symmetry line equations. At the outflow boundary, (6) is differenced with centered X-differences and two or three point backward Y-differences.

It remains to discuss how the shock slope $s_\theta=s_YY_\theta$ is related to the discrete (unknown) shock locations $r=s_m=s((m-1)\Delta Y)$ at $m=1,2,\cdots,M$ in computational space. We investigate the effect of two different approximations, namely centered differencing

$$(s_\theta)_m=(Y_\theta)_m\frac{(s_{m+1}-s_{m-1})}{2\Delta Y} \tag{17}$$

(with 2- or 3-point backward differencing at $m=M$), and a four-point formula

$$(s_\theta)_m=(Y_\theta)_m\frac{(s_{m-2}-6s_{m-1}+3s_m+2s_{m+1})}{6\Delta Y} \tag{18}$$

(with appropriate simplification at $m=2$ using evenness of s, and a different four-point backward formula at $m=M$). This latter approach appears to yield a "smoother" shock than does (17).

Counting three unknowns (ρ, u, v) per mesh point (although the stagnation values are explicitly known) and M unknown shock locations, these finite difference equations constitute $K=M(3N+1)$ equations in K unknowns. This system of equations is solved by a modified Newton's method.

2D. Solution by Newton's method and continuation. For a nonlinear system of K equations in K unknowns written in the form $T(x)=0$, a damped Newton's method defines an iteration for the approximate solution x^ν by the linearization

$$T'(x^\nu)(x^{\nu+1}-x^\nu)=-\alpha T(x^\nu), \qquad 0<\alpha \leqq 1, \tag{19}$$

where T' is the Jacobian matrix associated with the mapping T, i.e., $T'_{ij}=\partial T_i/\partial x_j$, $(i,j=1,2,\cdots,K)$. If x^* is a solution to $T(x)=0$, then for $\alpha=1$ (usual Newton's method) and an initial guess x^0 sufficiently close to x^* the iterates x^ν converge quadratically to x^*. A less accurate initial guess can often be improved by taking $\alpha<1$ (giving linear convergence) until the damped Newton iterates are close enough to x^*. For complicated problems such as those addressed here, the Jacobian matrix T' may be fairly complicated to compute analytically. We therefore form T' using forward difference quotients

$$T'_{ij}=\frac{T_i(x+\varepsilon e_j)-T_i(x)}{\varepsilon}, \tag{20}$$

which when used in (19) still yields quadratic convergence under certain restrictions on T, T', x^0, and ε, and linear or superlinear convergence otherwise [13, p. 360]. It is interesting to note that some implicit schemes for solving $\partial x/\partial t=T(x)$ formally reduce to the damped Newton's method (19) in the limit $\Delta t\to\infty$ [14], [15].

The two primary difficulties which may be anticipated in applying Newton's method to a very large system of nonlinear equations are (i) to efficiently solve the large linear system (19) and (ii) to generate a sufficiently good initial guess. We concentrate first on the question of efficiency.

The primary reason why Newton's method can be efficiently implemented on the present and similar problems is that the Jacobian matrix T' is sparse and has a good deal of structure, leading to efficient solution of the linear system (19). This structure depends on the ordering of the unknowns and equations, and also on the particular difference approximations used. We consider the mesh points in the order $((n=1,N), m=1,M)$ where the "implied loop" notation borrowed from FORTRAN means that n goes from 1 to N for each increment of m. We order the unknowns as $((\rho_{n,m}, u_{n,m}, v_{n,m}, n=1,N), s_m, m=1,M)$ and the equations as (3 body equations, (3 interior equations, $n=2,N-1$), 4 shock equations, $m=1,M$). With this ordering and with first order accurate Y-differences at $m=M$ and centered s_θ, the structure of T' (illustrated for a coarse mesh in Fig. 3) is $M\times M$ block tridiagonal with square blocks of size $3N+1$. For the four point approximation (18) of s_θ, some stray columns appear outside the block tridiagonal structure. For second order (3 point backward) Y-differences at $m=M$, a stray block appears on the last row of T'. The use of three point X-differences at the wall and the shock does not alter the block structure but only complicates the blocks themselves.

Seeking the best method, we investigate array storage and computer time requirements in solving the linear system (19) using three different methods: (i) LU decomposition with partial pivoting and storage of the full matrix, (ii) the Harwell sparse matrix package [16], and (iii) a block tridiagonal solver [17]. An LU decomposition which takes advantage of the banded structure of T' would save storage but require essentially the same running time as (i). In the Harwell package, only nonzero entries of the Jacobian are stored, and pivoting is done using the strategy of Markowitz [18] with the constraint that the pivots are not less than some specified quantity. For method (iii) the subsystems are solved using (i). We remark that in-core solution of the linear system may become prohibitive for excessively fine computational meshes, necessitating an out-of-core solution.

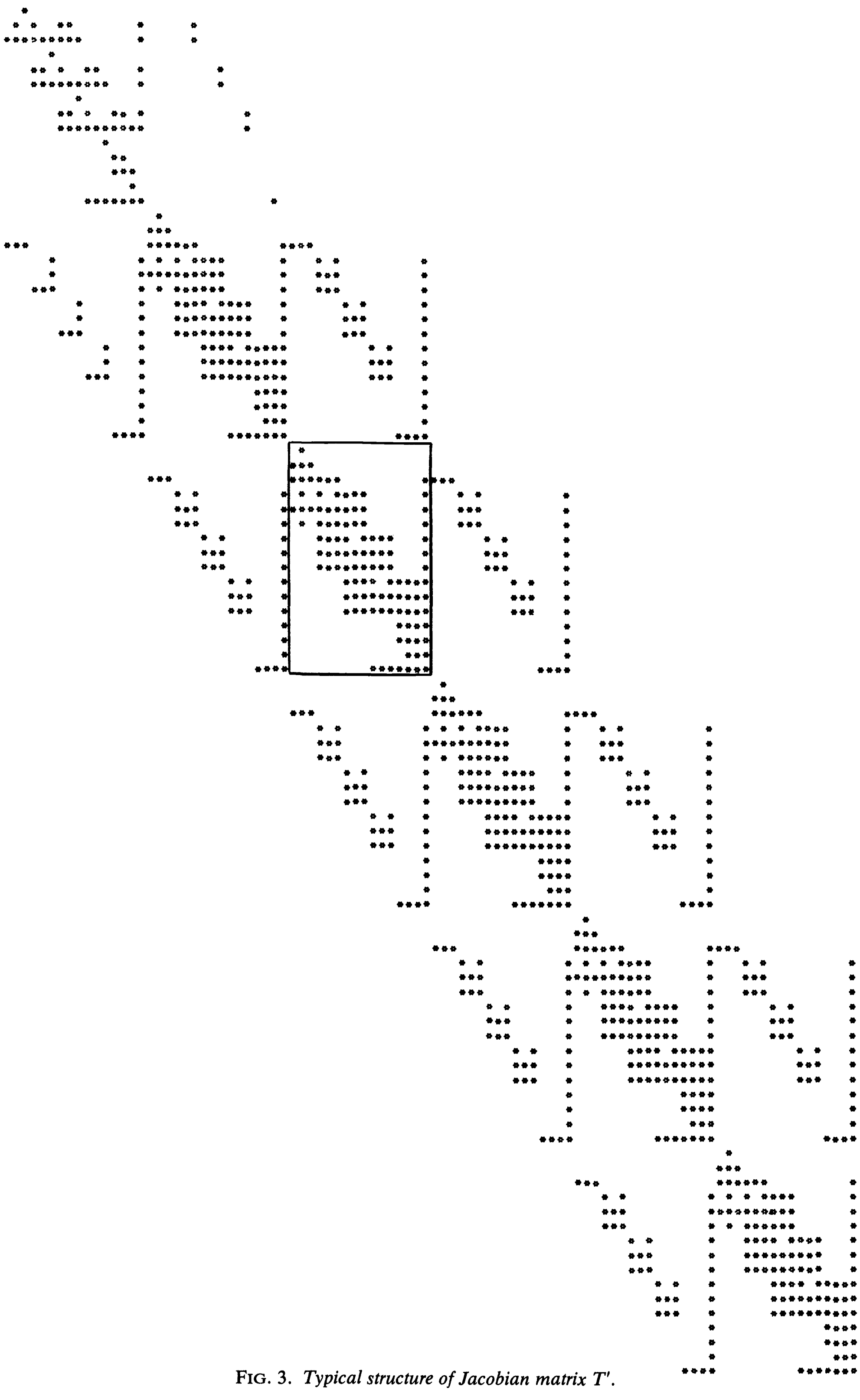


FIG. 3. *Typical structure of Jacobian matrix T'.*

In addition to solving the linear system efficiently, there are several modifications of Newton's method itself which may be employed as computational strategies aimed at solving the nonlinear system $T(x)=0$ more economically. First, by not updating the Jacobian at each iteration (keeping T' fixed), the same factorization of T' may be used to define several successive iterates (we call this "repeated backsolves"). This strategy circumvents the most costly part of the linear system solution, and in the case of one extra backsolve per iteration (two solves per new Jacobian), can yield cubic or better convergence [13, p. 315]. Second, some part of T' may be set to zero in order to obtain a matrix with a specific structure which is easy to factor. An example of this is the present problem with s_θ approximated by (18), leading to the nearly block tridiagonal T' described above. By "zeroing out" that part of T' which is not block tridiagonal, we obtain a linear system which can be solved by the block tridiagonal solver, but the iterative method that (19) represents is no longer Newton's method and hence quadratic convergence is lost. Another example is the use of Newton's method on successive sweeps of columns or rows in the finite difference mesh, rather than for the full problem of all unknowns and all equations considered at once. Although not investigated here, this is related to splitting for implicit methods and is equivalent to zeroing out part of the Jacobian for the full problem. Third, the nonlinear iteration might be made more efficient by using "quasi-Newton" methods [19] in which the Jacobian matrix (or its inverse or factorization) is advanced by rank-one updates. We have not explored this last approach and, to our knowledge, its application to fluid problems remains uninvestigated.

For the problem we have considered here, our initial guess was adequate to insure convergence. In general, the difficulty of constructing an adequate initial guess for the blunt body problem is to some extent dependent on the values of the physical parameters γ and M_∞ and the body shape $b(\theta)$. For example, the convergence of iterative methods becomes more difficult when M_∞ is near 1 [20], for larger values of γ (e.g., modelling nearly incompressible flow), and for more irregular body shapes. For these more difficult problems our computations indicate that continuation with respect to parameters and/or body shape is an effective procedure. Briefly, we start with a value of the parameters or given body shape for which we can solve the problem, and then solve a succession of problems, each time incrementing the parameter until the final problem is solved. If the increment is small enough, the solution of each problem is an adequate guess for the next problem. This type of continuation is related to Davidenko's method (see, e.g., [21]) where in general a homotopy parameter is used. More sophisticated continuation methods may also be employed [22], [23].

2E. Existence of solutions and convergence behavior. In general, a system of nonlinear algebraic equations $T(x)=0$ may have a unique (real) solution, multiple solutions, or no solution. In the case where $T(x)=0$ represents a physical problem and multiple solutions exist, sometimes only one solution is a good approximation of the physics, and other times the solutions may be very "close" to each other and it is difficult to choose the best approximation. The question of multiple solutions to model fluid dynamics problems has been investigated in [24], [25]; in the latter the bifurcation of such solutions is considered. The particular solution that is obtained in computation can depend both on the iterative method and on the initial guess. In our experience, a time-asymptotic method which is a consistent approximation of the time-dependent physics is less likely than Newton's method to find multiple solutions to fluid dynamics problems (since some of these solutions may be unstable). Indeed, as presented in § 3D, our computations with Newton's method reveal multiple solutions to the difference equations formulated above; we have seen no reference elsewhere to such multiple solutions for blunt body flow.

A significant advantage of Newton's method over the time-asymptotic method is that its iterates clearly indicate whether convergence is or is not occurring. This is true whether or not a solution to $T(x)=0$ exists. When a solution does exist, the quadratic convergence of Newton's method near the solution clearly identifies it. The time-asymptotic method may have very small iteration error while still quite far from a solution [26, p. 174]. When no solution exists or when near a local minimum, the Newton iterates exhibit oscillatory behavior clearly showing no convergence. Alternatively, the time-asymptotic method may have steadily decreasing iteration error over many time steps before eventually diverging (no solution), or have oscillatory iteration error before converging to a solution. Depending on the convergence criterion, one could erroneously infer the existence of a solution near a local minimum or even when one doesn't exist. We have seen most of these types of behavior when the present Newton method and time-asymptotic shock tracking methods are applied to some formulations of the blunt body problem.

3. Results. In this section we investigate the accuracy, convergence, and efficiency of the above outlined technique, discuss continuation with respect to some physical parameters, and exhibit multiple solutions of our finite difference system. In all cases we take $\varepsilon = 10^{-5}$ in (20) and $\gamma = 1.4$. Our convergence criterion for Newton's method is $\max_j |x_j^\nu - x_j^{\nu-1}| < 10^{-6}$.

3A. Accuracy. We first discuss flow over a sphere at incident Mach number $M_\infty = 5.017$ and compare our computed results with experimental data [27]. The initial guess for this computation is obtained as follows. A shock shape $r = s$ is assumed, and the shock jump conditions (14) are solved for the flow values behind the shock. At the body, the known stagnation point conditions, the assumption of isentropic flow along the body surface, and a modified Newtonian pressure distribution [28] are used to get the flow variables. The interior flow variables are obtained by linear interpolation along $Y =$ constant lines between the body and the shock, after which an adjustment is made to satisfy (3). Convergence of the Newton iteration starting from this guess is described in § 3B.

Using an $N = 6$, $M = 10$ mesh with $\theta_{\max} \sim 60°$, second order boundary and outflow conditions, and the four point s_θ approximation (18), results comparing computed shock shape and density contours (isopycnics) with experimental data are given in Fig. 4. The computational plots use linear interpolation. We remark that experimental isopycnics (though not frequently available) appear to provide a more stringent test of computational accuracy than does either shock shape or surface pressure. This point is illustrated in [29]. However, for the sake of completeness, we have included in Fig. 5 a comparison of the computed surface pressures on 6×10 and 11×14 meshes with the results obtained using the code described in [6].

For the above problem, second order accurate boundary conditions (shock, body, and outflow) yield results which are virtually indistinguishable from those obtained with first order accurate conditions. However, if $\theta_{\max}$ is increased to $\sim 100°$, the computation with first order accurate outflow conditions behaves as though there is no solution to the nonlinear difference equations (see § 2E), whereas with second order outflow conditions the computation converges normally.

The particular approximation used for s_θ influences the smoothness of the computed shock. When the four-point approximation (18) is replaced by centered differencing (17), the computed shock is oscillatory, with the magnitude of the oscillations

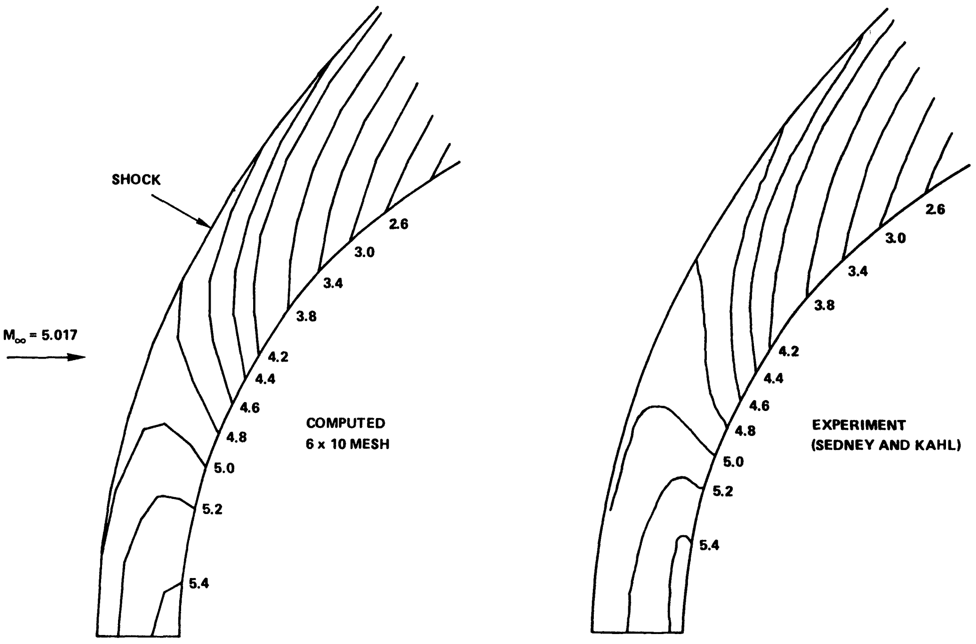


FIG. 4. *Computed and experimental isopycnics for $M_\infty = 5.017$ flow over a sphere.*

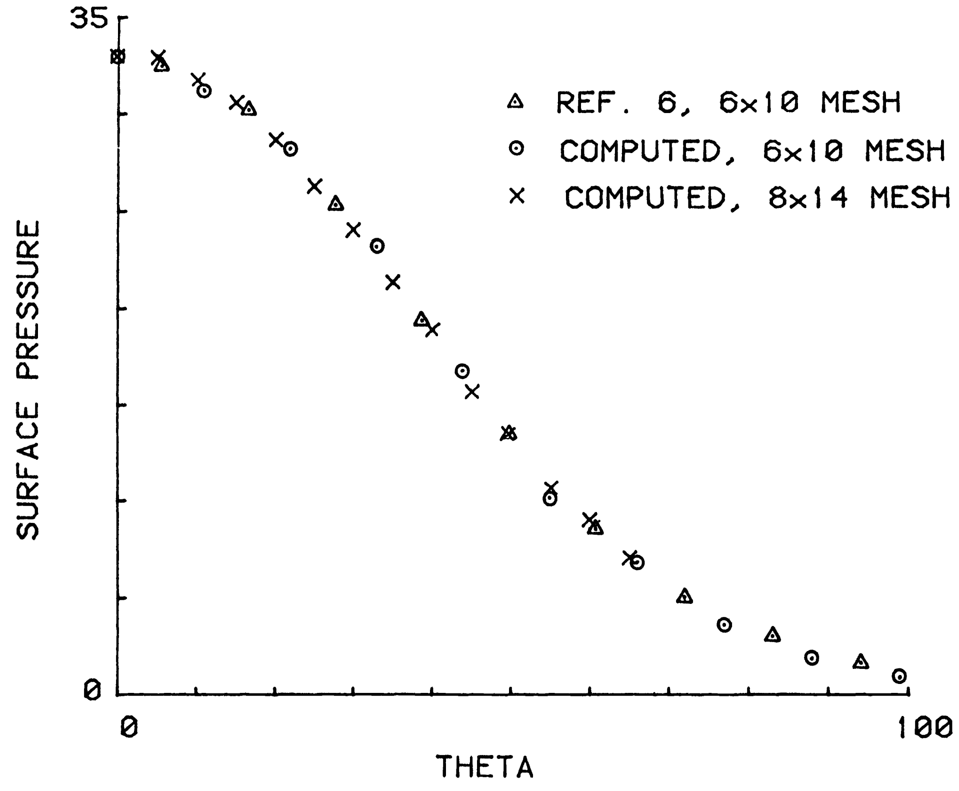


FIG. 5 *Comparison of computed surface pressures.*

becoming smaller at higher incident Mach numbers. Such oscillatory shock solutions will be shown in § 3D.

3B. Efficiency and convergence. We now consider the questions of efficiency and convergence. We are concerned with both economical solution of the large linear system and overall computational strategies such as choosing damping factors α and using repeated backsolves. Several computations have been made for the $M_\infty = 5.017$ sphere with varying mesh sizes, s_θ approximations, accuracy of outflow conditions and repeated backsolves. In each case the three linear system solvers (see § 2D) were tried. Results showing computer times for the various runs are given in Table 1. These times refer to computations made on the Lawrence Berkeley Laboratory CDC 7600 using the FTN4 compiler with normal optimization; the times include all overhead (e.g., printing) except the generation of the initial guess which is done separately. For $\alpha = 1$ (except where marked by * in Table 1) and using the Jacobian formed with forward difference quotients, quadratic convergence was obtained starting at the above-mentioned initial guess. At the first iteration this guess gives $\max_j T_j \sim 300$ and an iteration error of ~ 0.3.

TABLE 1

Mesh ($N \times M$)	Case	α	s_θ	Outflow accuracy order	Number solves per Jacobian	7600 seconds: Block tri-diagonal	Harwell sparse matrix package	Full LU decom-position
5×6	a	1	Centered	1	1	.71	.65	.87
6×10	b	1	Centered	1	1	2.22	2.02	3.65
	c	1	Centered	1	2	1.20	1.30	1.97
	d	1	Four-point	1	1	3.29*	2.47	3.92
	e	1	Centered	2	1	3.75*	2.20	3.92
8×14	f	(i)	Centered	1	1	9.12	10.08	36.09
	g	(ii)	Centered	1	1	7.28	8.21	27.82

* Corresponds to modified Newton method—see text. In cases f and g, see text for choice of α

In general the block tridiagonal and Harwell solvers require about the same amount of time (except *), whereas *LU* decomposition of the full matrix without provision for sparsity requires considerably more. For the Harwell package there is a trade off between the amount of storage required and its efficiency. For the most efficient case it requires about twice as much storage as the block tridiagonal solver, but it is more versatile in its ability to handle more general matrix structures. For example, in cases *d* and *e* of Table 1, the Jacobian T' is not block tridiagonal, so that those runs marked * represent a modified Newton's method in which part of the Jacobian is ignored (and convergence is not quadratic). When two solves per Jacobian was tried in case *c*, the convergence was better than cubic. In cases *f* and *g*, Newton's method with $\alpha = 1$ failed to converge from the initial guess. We therefore employed a strategy (i) of starting at $\alpha \sim .2$ and increasing α linearly toward 1 as $\max_j T_j$ decreased from its initial value to some preset lower value. It is also possible to keep α fixed until $\max_j T_j$ is small enough, but this is generally slower. If, in addition to strategy (i), we provisionally try $\alpha = 1$ at each iteration and override strategy (i) with an $\alpha = 1$

step if $\max_j T_j$ is reduced to at least 90% of its previous value, the savings in case g (strategy ii) result.

It is very difficult to compare the timings in Table 1 with those of time-dependent methods reported elsewhere. This is due to differences between computers, optimization, initial guesses, convergence criteria, Mach numbers, mesh sizes, etc. The interested reader is referred to [6] where times (on a different CDC 7600) are given for some explicit and implicit finite difference methods.

For a very rough comparison we have run the implicit code of [6] on a $M_\infty = 5$ sphere using a 6×10 mesh and an initial guess similar to ours. This computation takes 16 seconds (including generating the initial guess and all overhead) to run 600 iterations and achieves a maximum shock speed of 1.5×10^{-7} (2.4×10^{-4} after 300 iterations) and a maximum total enthalpy error of 0.9%.

3C. Continuation. Beginning with a solution on a 6×10 mesh for the $M_\infty = 5.017$ sphere, the method of continuation was used to first alter the body shape to an ellipsoid, and then to increase the Mach number to 6.8. We arbitrarily used five continuation steps in body shape and four in Mach number. The results, shown in Fig. 6, agree well with experimental data [30] for shock shape (no isopycnic data

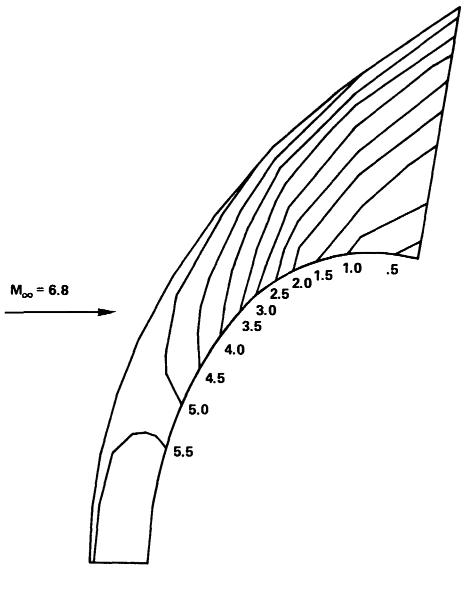


FIG. 6. *Computed isopycnics for $M_\infty = 6.8$ flow over an ellipsoid.*

could be located). In other experiments, we have used continuation with respect to γ, and foresee no difficulty in continuing to a different equation of state.

Some attempts were also made to obtain low M_∞ solutions via continuation. Beginning with a 6×10, $M_\infty = 5.017$ solution with $\theta_{\max} = 75°$ and second order boundary conditions, the Mach number M_∞ was decreased in steps of 0.5 to 2.517, then in steps of 0.1 to 1.817. This smaller step size was taken because the 0.5 step down from 2.517 led to divergence. Further attempts to get to smaller M_∞ failed, with the computation behaving as though there was no solution to the difference equations (see § 2E). Attempts to interpolate this solution onto a finer mesh and proceed also failed. At present, the authors are investigating changes in methodology which would enable the computation to reach very small values of M_∞.

3D. Multiple solutions. We have found two solutions to our finite difference system for flow over a sphere using first order accurate boundary conditions and the centered approximation for s_θ. These two solutions are displayed in Fig. 7 for $M_\infty = 3$.

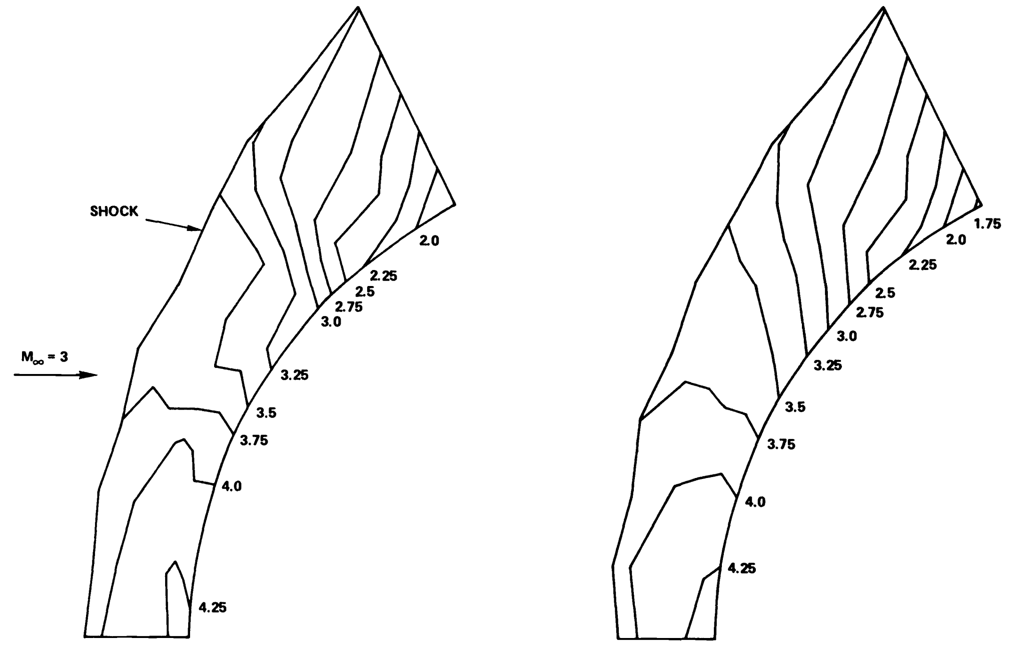


FIG. 7. *Two different solutions to the same system of finite difference equations for* $M_\infty = 3$ *flow over a sphere.*

Notice that the shock is oscillatory in opposite senses in the two solutions. These solutions were obtained by starting from the same initial guess and using different numbers of backsolves for each new Jacobian evaluation. Multiple solutions have also been obtained for higher M_∞ (though the oscillations in the shock are less pronounced) and when starting from different initial guesses. In all cases the solutions found were "close together," though many other solutions may also exist.

Appendix. Derivation of characteristic compatibility conditions. Some background on characteristic compatibility equations may be found in [31]. For their application to fluid dynamics, see [32]. Consider equations (1) and the time-dependent transformation $X = X(r, \theta, t)$, $Y = Y(r, \theta, t)$, $\tau = t$. Let

$$Q = \begin{bmatrix} \rho \\ u \\ v \\ e \end{bmatrix}.$$

Then, using the chain rule, (1) becomes

$$Q_\tau + B_1 Q_X + C_1 Q_Y + D_0 = 0, \tag{A.1}$$

where $B_1 = X_t I + X_r B_0 + X_\theta C_0$, $C_1 = Y_t I + Y_r B_0 + Y_\theta C_0$, $B_0 = A^{-1}B$, $C_0 = A^{-1}C$, $D_0 = A^{-1}(H + 2F/r + \cot \theta G)$ and $A = [\partial U/\partial Q]$, $B = [\partial F/\partial Q]$, $C = [\partial G/\partial Q]$. (Here subscripts τ, X, Y, t, r, θ indicate partial differentiation.) Specifically,

$$B_0=\begin{bmatrix} u & \rho & 0 & 0 \\ k_1/\rho & u & 0 & k_2/\rho \\ 0 & 0 & u & 0 \\ 0 & p/\rho & 0 & u \end{bmatrix}, \qquad C_0=\frac{1}{r}\begin{bmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ k_1/\rho & 0 & v & k_2/\rho \\ 0 & 0 & p/\rho & v \end{bmatrix}$$

and

$$D_0=\frac{1}{r}\begin{bmatrix} \rho(2u+v\cot\theta) \\ -v^2 \\ uv \\ (p/\rho)(2u+v\cot\theta) \end{bmatrix}$$

where $k_1=(\partial p/\partial\rho)_e$, $k_2=(\partial p/\partial e)_\rho$.

The characteristic matrix for (A.1) is

$$A^*=\lambda_1 I+\lambda_2 B_1+\lambda_3 C_1=\begin{bmatrix} \sigma & \Lambda_2\rho & \Lambda_3\rho/r & 0 \\ k_1\Lambda_2/\rho & \sigma & 0 & k_2\Lambda_2/\rho \\ k_1\Lambda_3/\rho r & 0 & \sigma & k_2\Lambda_3/\rho r \\ 0 & \Lambda_2 p/\rho & \Lambda_3 p/\rho r & \sigma \end{bmatrix},$$

where

$$\sigma=\Lambda_1+\Lambda_2 u+\Lambda_3\frac{v}{r}, \quad \Lambda_1=\lambda_1+\lambda_2 X_t+\lambda_3 Y_t, \quad \Lambda_2=\lambda_2 X_r+\lambda_3 Y_r, \quad \Lambda_3=\lambda_2 X_\theta+\lambda_3 Y_\theta.$$

The characteristic condition is $\det A^*=\sigma^2(\sigma^2-(\Lambda_2^2+\Lambda_3^2/r^2)c^2)$ where $c^2=k_1+k_2 p/\rho^2$ is the square of the speed of sound. Corresponding to the four characteristic conditions $\sigma_{1,2}=0$, $\sigma_{3,4}=\pm(\Lambda_2^2+\Lambda_3^2/r^2)^{1/2}c$ are four independent left null vectors (defined by $lA^*=0$) $\bar{l}_1=(0,\Lambda_3/r,-\Lambda_2,0)$, $\bar{l}_2=(-p,0,0,\rho^2)$, $\bar{l}_{3,4}=(-k_1\sigma_{3,4}, \rho\Lambda_2 c^2, \rho\Lambda_3 c^2/r, -k_2\sigma_{3,4})$. The four characteristic compatibility conditions are obtained by left multiplying eqns. (A.1) by $\bar{l}_i$, $i=1,2,3,4$. These conditions are rather complicated and thus are not written out. The conditions for $i=1,2$ correspond to the particle path while those for $i=3,4$ are associated with the Mach conoid. The ray directions on the particle path are given by $dX/d\tau=\bar{A}$ and $dY/d\tau=\bar{B}$ and those on the Mach conoid by

$$\frac{dX}{d\tau}=\bar{A}-\frac{c^2(\Lambda_2 X_r+\Lambda_3 X_\theta/r^2)}{\sigma} \quad \text{and} \quad \frac{dY}{d\tau}=\bar{B}-\frac{c^2(\Lambda_2 Y_r+\Lambda_3 Y_\theta/r^2)}{\sigma}$$

where $\bar{A}=X_t+uX_r+vX_\theta/r$ and $\bar{B}=Y_t+uY_r+vY_\theta/r$. These directions play a crucial role in determining the computational boundary treatment, as various choices of Λ_2 and Λ_3 will correspond to rays reaching the boundary in the positive time direction from either inside or outside of the computational field. Those from inside are called "admissible" and provide compatibility conditions which are used to advance the flow variables at the boundaries. Those from outside the field are "inadmissible" and must be replaced by boundary conditions. The total number of admissible compatibility conditions plus the number of boundary conditions equals the number of unknowns [32].

At the body $r=b$, the choice $\Lambda_2=-1$, $\Lambda_3=b_\theta$ renders the $i=1,2,4$ conditions admissible. The $i=3$ condition corresponds to a ray coming from outside the computational field and is replaced by the boundary condition of flow tangency at the wall. It can easily be shown that the $i = 2$ condition is a statement that entropy is conserved along particle paths. Also, using the definition (3) of the total enthalpy Ω, the $i = 1, 2$ conditions can be combined to give an equation for Ω_τ which may be used to replace the $i = 1$ condition. This equation for Ω_τ is identically satisfied in the steady state.

At the shock $r = s$, the choice $\Lambda_2 = 1$, $\Lambda_3 = -s_\theta$ gives the $i = 4$ condition as the only admissible one. The other compatibility conditions are replaced by the Rankine–Hugoniot jump relations.

Acknowledgment. The authors are grateful to John B. Bell for several helpful discussions.

REFERENCES

[1] G. R. Shubin, A. B. Stephens and H. M. Glaz, *Steady shock tracking and Newton's method applied to one-dimensional duct flow*, J. Comp. Phys., 39 (1981), pp. 364–374.

[2] A. Rizzi, *Solution by Newton's method to the steady transonic Euler equations*, Proc. of the Sixth International Conference on Numerical Methods in Fluid Dynamics, Lecture Notes in Physics 90, Springer-Verlag, New York, 1979, pp. 460–467.

[3] B. Fornberg, *A numerical study of steady viscous flow past a circular cylinder*, J. Fluid Mech., 98 (1980), pp. 819–855.

[4] B. Gustafsson and P. Wahlund, *Finite difference methods for computing the steady flow about blunt bodies*, J. Comp. Phys., 36 (1980), pp. 327–346.

[5] V. V. Rusanov, *A blunt body in a supersonic stream*, Ann. Rev. Fluid Mech., 8 (1976), pp. 377–404.

[6] P. Kutler, S. R. Chakravarthy and C. P. Lombard, *Supersonic flow over ablated nosetips using an unsteady implicit numerical procedure*, AIAA Paper 78-213, Huntsville, Alabama, Jan. 1978.

[7] G. Moretti and M. Abbett, *A time-dependent computational method for blunt body flows*, AIAA J., 4 (1966), pp. 2136–2141.

[8] G. Moretti, *The λ-scheme*, Computers and Fluids, 7 (1979), pp. 191–205.

[9] D. W. Hall, *Calculation of inviscid supersonic flow over ablated nosetips*, AIAA paper 79-0342, Huntsville, Alabama, 1979.

[10] M. D. Salas, *Flow properties for a spherical body at low supersonic speeds*, Symposium on Computers in Aerodynamics, Polytechnic Institute of New York, June 1979.

[11] W. D. Hayes and R. F. Probstein, *Hypersonic Flow Theory, vol.* 1, *Inviscid Flows*, Academic Press, New York, 1959.

[12] H. Viviand, *Conservative forms of gas dynamic equations*, La Recherche Aerospatiale, 1 (1974), pp. 65–68.

[13] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[14] E. Turkel, *Numerical methods for large-scale, time-dependent partial differential equations*, ICASE Report 79-20, Langley, VA, Aug. 1979; in Computational Fluid Dynamics, W. Kallmann, ed., Hemisphere, Washington, 1980, pp. 127–262.

[15] W. R. Briley and H. McDonald, *On the structure and use of linearized block implicit schemes*, J. Comp. Phys., 34 (1980), pp. 54–73.

[16] I. S. Duff, MA28–*A set of Fortran subroutines for sparse unsymmetric linear systems*, Harwell Report AERE-R 8730 (1979 revision), Oxfordshire, England.

[17] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*, John Wiley, New York, 1966.

[18] H. M. Markowitz, *The elimination form of the inverse and its application to linear programming*, Management Sci., 3 (1957), pp. 255–269.

[19] J. E. Dennis, Jr., and J. J. Moré, *Quasi-Newton methods, motivation and theory*, SIAM Rev., 19 (1977), pp. 46–89.

[20] C. Weiland, *Solution of the Eulerian equations for three-dimensional flows around blunt bodies*, Technical Translation ESA TT-268, Feb. 1976. (Original Report No. DLR-FB 75-60, Institut für Angewandte Gasdynamik, Cologne, Germany.)

[21] W. C. Rheinboldt, *Methods for Solving Systems of Nonlinear Equations*, CBMS Regional Conference Series in Applied Mathematics 14, Society for Industrial and Applied Mathematics, Philadelphia, PA 1974.

[22] E. ALLGOWER AND K. GEORG, *Simplicial and continuation methods for approximating fixed points and solutions to systems of equations*, SIAM Rev., 22 (1980), pp. 28–85.

[23] H. B. KELLER, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, New York 1977.

[24] R. B. KELLOGG, G. R. SHUBIN AND A. B. STEPHENS, *Uniqueness and the cell Reynolds number*, SIAM J. Numer. Anal. 17 (1980), pp. 733–739.

[25] A. B. STEPHENS AND G. R. SHUBIN, *Multiple solutions and bifurcation of finite difference approximations to some steady problems of fluid dynamics*, this Journal, 2 (1981), pp. 404–415.

[26] P. ROACHE, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1972.

[27] R. SEDNEY AND G. D. KAHL, *Interferometric study of the hypersonic blunt body problem*, Planetary and Space Sciences, vol. 4, Pergamon, New York, 1961, pp. 337–351.

[28] G. G. CHERNYI AND R. F. PROBSTEIN, *Introduction to Hypersonic Flow*, Academic Press, New York, 1961.

[29] G. BEN-DOR AND I. I. GLASS, *Nonstationary oblique shock-wave reflections: actual isopycnics and numerical experiments*, AIAA J., 16 (1978), pp. 1146–1153.

[30] W. K. OSBORNE AND J. F. W. CRANE, *Measurement of the sonic line, bow shock wave shape and stand-off distance for blunt nose bodies at $M = 6.8$*, Royal Aircraft Establishment TN AERO. 2707, Aug. 1960.

[31] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics*, vol. 2, Interscience, New York, 1962.

[32] C. P. KENTZER, *Discretization of boundary conditions on moving discontinuities*, Lecture Notes in Physics 8, Springer-Verlag, New York, 1971, p. 108.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

SMOOTH REGRADING OF DISCRETIZED DATA*

JAROSLAV KAUTSKY† AND NANCY K. NICHOLS‡

Abstract. Methods for producing nonuniform transformations, or *regradings*, of discrete data are discussed. The transformations are useful in image processing, principally for enhancement and normalization of scenes. Regradings which "equidistribute" the histogram of the data, that is, which transform it into a constant function, are determined. Techniques for smoothing the regrading, dependent upon a continuously variable parameter, are presented. Generalized methods for constructing regradings such that the histogram of the data is transformed into *any* prescribed function are also discussed. Numerical algorithms for implementing the procedures and applications to specific examples are described.

Key words. image processing, image enhancement and normalization, histogram modification, discrete transformations, regrading, equidistribution, mesh selection

1. Introduction. Digitized images are produced by grading the data points, or "pixels", of the image into a finite site of "gray levels", or *grades*. Similarly other experimental and statistical data are frequently graded into a finite set of classes. Each class, or grade, is represented by an integer value, and to each data point the value of its grade is assigned. The histogram of total occurrences of data points belonging to each grade may be formed, and with each grade we associate a *weight* equal to its number of occurrences.

In practical applications, a *regrading* of the data is frequently required, for example, for purposes of image display, enhancement or normalization. A regrading is essentially just a transformation from one set of integers into another. The most natural transformation is a *linear* regrading, in which the original grades are divided as equally as possible among the new grades. In this case, the histogram of the regraded data is basically just a scaling of the original histogram. In some situations, however, it is advantageous to determine a nonlinear regrading, such that the *histogram* of the regraded data is as flat, i.e., constant, as possible [1], [2], [3]. In this case the total occurrences of the old grades are as equally distributed over the new grades as possible, and such a regrading is called *equidistributing*. It is well known that of all transformations, the equidistributing regrading is optimal in the sense that it maximizes retained information [7]. The linear regrading, however, preserves quantitative relations between grades which are often significant for interpretation. Obviously we may also consider a range of "smoothed" regradings between these two extreme cases: linear and equidistributing. In this paper we present efficient numerical procedures for determining such regradings explicitly. The smoothed regradings are particularly useful for image enhancement when the histogram of the image data is extremely unevenly distributed.

Linear and equidistributing regradings are specific examples of histogram modification [2], [3]. *Weighted* regradings of the data, such that the histogram of the transformed data matches other prescribed *reference* functions, are also valuable, for example, for calibration of image data for multitemporal analysis. Generalized procedures for obtaining such weighted regradings are also presented.

2. Statement of the problem. We consider a set of grades represented by the integers $\{1, 2, \cdots, n\}$ and a set of nonnegative weights $\{f_1, f_2, \cdots, f_n\}$ associated with

* Received by the editors July 2, 1980, and in final revised form October 26, 1981.

† School of Mathematical Sciences, Flinders University, Bedford Park, S.A. 5042, Australia.
‡ Department of Mathematics, University of Reading, Whiteknights, Reading, RG6 2AX, England.

these grades. We define a *regrading* to be a nondecreasing map from the given set of integers into another set. In particular we have:

DEFINITION 1. If $\mathbb{Z}_k = \{1, 2, \cdots, k\}$ is the set of integers from 1 to k, and if m, n are integers, then

$$\boldsymbol{\tau} : \mathbb{Z}_n \to \mathbb{Z}_m$$

is a *regrading* if $1 \leqq i < j \leqq n$ implies

$$1 \leqq \tau_i \leqq \tau_j \leqq m \quad \forall i, j.$$

(We represent the mapping $\boldsymbol{\tau}$ by the integer-valued vector $(\tau_1, \tau_2, \cdots, \tau_n)$.)

For each regrading $\boldsymbol{\tau}$, a new (transformed) set of weights $\{g_1, g_2, \cdots, g_m\}$ defined by

$$g_i = \sum_{j \ni \tau_j = i} f_j, \qquad i = 1, 2, \cdots, m \tag{1}$$

is associated with the new grades $\{1, 2, \cdots, m\}$; that is, the new weights g_i are the sum of all the f_j for which $\tau_j = i$. Denoting

$$N = \sum_{j=1}^{n} f_j \quad \text{and} \quad M = \max_{1 \leqq j \leqq n} f_j, \tag{2}$$

we observe the following obvious relations:

$$N = \sum_{1}^{m} g_i, \tag{3}$$

$$M \leqq M_\tau \equiv \max_{1 \leqq i \leqq m} g_i, \tag{4}$$

$$N \leqq m M_\tau. \tag{5}$$

We wish to determine a regrading in which the transformed weights are equally distributed over the new grades. In that case, we must have, by (3),

$$g_i = \frac{N}{m} \quad \forall i = 1, 2, \cdots, m. \tag{6}$$

Unfortunately, because of the discrete character of our transformations, we cannot expect such a regrading to be possible, except in rare cases. In particular, if $M > N/m$, it follows from (4) that (6) cannot be satisfied. We could, for example, consider regradings $\boldsymbol{\tau}$ for which M_τ is minimal, which would give the required equidistribution when possible. In this paper we introduce, instead, another concept of equidistribution, which corresponds to the theory discussed in [4] for continuous weight functions. In § 3 we define a regrading which is "approximately" equidistributing, and give a procedure for its construction. In the following section we consider methods for smoothing the regrading, dependent upon a continuously variable parameter. In § 5 we describe the implementation of such procedures and give results for specific examples. A more general concept of *weighted* regrading is discussed in the final section.

3. Equidistributing regradings. The grades $\{1, 2, \cdots, n\}$ and the corresponding weights $\{f_1, f_2, \cdots, f_n\}$ can be represented graphically on the region $[0, n]$ by a piecewise constant function taking values f_j on the intervals $(j-1, j)$, $j = 1, 2, \cdots, n$. (This is the histogram associated with the grading.) This function may be equidistributed over m points, as in [1], simply by finding subintervals (x_{j-1}, x_j), $j = 1, 2, \cdots, m$ such that the integral of the function over each subinterval is equal to a constant (given by the total integral divided by m). Then by associating each of the old grades with a specific subinterval a transformation is defined which gives a regrading of the discrete data. We construct the transformation explicitly as follows.

Given integer n and weights $\{f_1, f_2, \cdots, f_n\}$, $f_j \geqq 0$, $\forall j$, let $F(t)$ be a piecewise linear continuous function for $t \in [0, n]$, such that

$$F(0) = 0, \qquad F(j) = \sum_{k=1}^{j} f_k, \qquad j = 1, 2, \cdots, n, \tag{7}$$

with corners at points $1, 2, \cdots, n-1$. Obviously F is nondecreasing and F' is a piecewise constant function such that

$$F'(t) = f_j \quad \text{for } t \in (j-1, j), \quad j = 1, 2, \cdots, n.$$

Thus F' represents the weights f_j and we consider the midpoints $t_j = j - \frac{1}{2}$ of the intervals $(j-1, j)$ as points representing the grades j, $j = 1, 2, \cdots, n$. Now we define breakpoints x_k, $k = 0, 1, \cdots, m$ as the largest values such that

$$F(x_k) = k\frac{F(n)}{m}. \tag{8}$$

Obviously $x_0 = 0$, $x_m = n$, and the other breakpoints x_k, $k = 1, 2, \cdots, m-1$, can easily be determined by inverse linear interpolation, as the function F is piecewise linear.

By simple inspection, a regrading which equidistributes the weights exactly is possible if (and only if) $x_1, x_2, \cdots, x_{m-1}$ are all integers. To satisfy (6) it is then necessary and sufficient to choose

$$\tau_j = k, \qquad j \text{ such that } x_{k-1} < j - \tfrac{1}{2} < x_k. \tag{9}$$

Although the cases where the exact equidistribution is possible are expected to be very few, (9) leads to a general procedure for constructing a regrading which is exact whenever possible. We make the following definition:

DEFINITION 2. Given integers n and m, and nonnegative weights $\{f_1, f_2, \cdots, f_n\}$, let the breakpoints $x_0, x_1, \cdots, x_m$ be given by (8) where F is the piecewise linear function specified by (7). The regrading $\boldsymbol{\tau}$ satisfying

$$\tau_j = k \quad \text{for all } j \text{ such that } x_{k-1} < j - \tfrac{1}{2} \leqq x_k$$

is said to "equidistribute" the weights f_j, $j = 1, 2, \cdots, n$.

Remarks. We note that another regrading where $\tau_j = k$, for all j such that $x_{k-1} \leqq j - \frac{1}{2} < x_k$, could equally well be defined, and our choice is somewhat arbitrary. Similarly, if some weights f_j vanish, $F(t)$ may be constant over some interval and (8) need not have a unique solution. In our definition we have explicitly specified the breakpoint to be the largest solution of (8); however, this choice is essentially irrelevant, since the original grades in this case have no weight attached to them and hence have no effect on the transformations anyway.

Also of interest is a *linear* regrading, in which the old grades are evenly distributed among the new grades. Such a regrading arises from equidistributing the weights $f_j \equiv f$, $\forall j = 1, 2, \cdots, n$, where f is an arbitrary positive constant. In this case, a simple calculation leads to an explicit formula for the regrading constructed by Definition 2. We obtain

$$\tau_j = \left[\!\left[1 + \left(j - \frac{1}{2}\right)\frac{m}{n} \right]\!\right]', \qquad j = 1, 2, \cdots, n, \tag{10}$$

where $[\![x]\!]'$ denotes the largest integer less than x. We observe that in the case $m=n$, the regrading given by Definition 2 thus gives

$$\tau_j = j, \qquad j = 1, 2, \cdots, n,$$

as we would expect. Similarly, in the case $n = km$, we find that exactly k old grades are mapped into each new grade by this definition. Finally, in the case $n = m+1$, we obtain a "symmetric" transformation, which maps each old grade into one new grade, except at the center of the region, where two old grades are joined to form one new one. We thus verify the suitability of our definition and conclude that the behavior of the equidistributing regrading given by Definition 2 is reasonable in these special cases.

4. Smoothing the regrading. Problems where the weights are extremely unevenly distributed are difficult to regrade sensibly, and any attempt to equidistribute the data merely shifts the largest weights fairly arbitrarily into new grades. Therefore it is natural that we now examine procedures which *smooth* the data in the extreme cases, in order to produce consistent and reasonable regradings. The methods we aim to produce will depend continuously on a parameter p, $0 \leqq p \leqq 1$, which determines the amount of smoothing to be applied. At one end of the scale, say $p=0$, the method will give the *equidistributing* regrading of Definition 2 without smoothing, and at the other end, $p=1$, it will give a *linear* regrading. To realize such procedures, we use the concepts developed in [4] and [8] for constrained equidistributing meshes.

There are essentially two approaches for smoothing the regrading:

1) For some $K_1 \geqq 1$, enforce the restriction

$$\frac{\max_k (x_k - x_{k-1})}{\min_k (x_k - x_{k-1})} \leqq K_1, \tag{11}$$

giving a quasiuniform distribution of breakpoints.

2) For some $K_2 \geqq 1$, enforce the constraint

$$\frac{1}{K_2} \leqq \frac{x_{k+1} - x_k}{x_k - x_{k-1}} \leqq K_2, \tag{12}$$

giving a locally bounded distribution of breakpoints.

Obviously, $K_i = 1$ implies the *linear* distribution of the breakpoints in either case ($i = 1, 2$), and thus a linear regrading. On the other hand, very large K_i implies no constraint on the breakpoints. In both cases, the required constraints are enforced by a suitable change in the data, called *padding* (see [4]), which is essentially a smoothing of the weight function. We state here the necessary results and derive two basic, parameter dependent procedures for the padding.

Smoothing procedure 1. Let the vector $\mathbf{f}$ represent the weights $\{f_1, f_2. \cdots, f_n\}$. To enforce restriction (11), the given weights are replaced by new (padded) weights $P_p(\mathbf{f})$, defined by

$$(P_p(\mathbf{f}))_i = \max(f_i, c(p)), \tag{13}$$

where $c(p)$ is taken to be

$$c(p) = \frac{1}{K_1} M \equiv \frac{1}{K_1} \max_i f_i. \tag{14}$$

For a gradual smoothing we may choose, for example,

$$c(p) = pM.$$

However, when the average value N/n of the data is high or low, this choice may not achieve the required result. Therefore we use, instead, an alternative definition,

$$c(p)=p\left(pM+2(1-p)\frac{N}{n}\right), \tag{15}$$

which takes into account both the maximal weight and the average weight. We consider this choice to give a more natural dependence of the smoothing on the parameter p.

Smoothing procedure 2. In [4] it is shown that constraint (12) is enforced by replacing the given weights $\mathbf{f}$ by new (padded) weights $P_p(\mathbf{f})$ defined as

$$(P_p(\mathbf{f}))_j=\max_i \frac{f_i}{1+\lambda_p f_i|i-j|}, \tag{16}$$

where λ_p satisfies

$$m \log K_2=\lambda_p \sum_{j=1}^{n} (P_p(\mathbf{f}))_j. \tag{17}$$

It is not difficult to see that for λ_p sufficiently large, (16) gives

$$(P_p(\mathbf{f}))_j=f_j,$$

while, for $\lambda_p=0$ (or sufficiently small)

$$(P_p(\mathbf{f}))_j=\max_i f_i \equiv M.$$

To obtain a gradual smoothing, as in procedure 1, we may define λ_p as a function of p by replacing K_2 in (17) by $1/p$. However, as the sum of the $(P_p(\mathbf{f}))_j$ is not available until after λ_p has been chosen, we also replace this sum by $N=\sum_{j=1}^n f_j$. The effect of this replacement is merely to shift the parameter inside the interval [0,1]. The padded data is then defined by (16) with λ_p given by

$$\lambda_p=\frac{m}{N}\log\frac{1}{p}, \qquad 0<p\leqq 1. \tag{18}$$

This padding procedure takes into account both the maximum of the weights, through (16), and the average weight, through (18). We observe that this padding is of a more sophisticated form than that of procedure 1, and we expect the resulting regrading to be essentially smoother.

5. Some examples. In this section we present and discuss numerical results of the equidistributing and smoothing procedures for several examples. Smooth regradings are obtained by padding the given weight function using smoothing procedure 1 or 2 described in § 4, and then equidistributing the padded weights by the method described in § 3. Solutions are obtained for values of the padding parameter p belonging to [0, 1]; when $p=0$, the equidistributing regrading of Definition 2 is produced, and when $p=1$ a linear regrading results. The algorithms are implemented in a simple FORTRAN subroutine. The program code is given in [5].

In Tables 1a, b, c we summarize the results of regrading data from 20 grades into 8 grades, where the original grades have a V-shaped weight function given explicitly by

$$f_j=|j-10|, \qquad j=1,2,\cdots,20.$$

The regradings obtained with the two padding procedures and various values of p are shown in Table 1a. We observe that the regradings change gradually from the equidistributing (E) to the linear (U) distribution of grades. The linear grading (U) is as uniform as possible, given that 20 old grades cannot be equally divided among the 8 new grades. (We note that the particular solution obtained is a consequence of the inequalities specified in Definition 2, and that if these are altered as discussed in § 3, then a different "uniform" solution is produced, namely: $\{1, 1, 2, 2, 2, 3, 3, 4, 4, 4, \cdots\}$.)

The new transformed weights for the regradings of Table 1a are presented in Table 1b, and the breakpoints used to define the regradings are shown in Table 1c. The weights of the equidistributing regrading are seen to be as constant as possible in this discrete case, while those of the linear regrading remain approximately V-shaped. (It should also be noted that, *except* in the linear case, the choice of inequalities in Definition 2 is irrelevant, since none of the breakpoints is equal to $j-\frac{1}{2}$ for any j.)

In Table 2 the results of another application are summarized. For this example, the weight function is such that the equidistributing regrading from 20 to 8 grades is exact. We observe that the weights of the new grades are all equal and that the breakpoints are indeed integer-valued.

To demonstrate the difficulties in equidistributing extreme weight functions, we consider some examples where the histogram of the original grades contains large separated peaks. In the case of a single peak, where the weight function is $f_j = c\delta_{jk}$, $c > 0$, $j = 1, 2, \cdots, n$ for some k, a direct application of Definition 2 shows that $\tau_k \doteq m/2$; that is, the regrading always places the old grade k in the middle of the new scale, regardless of its original position. Similarly, equidistributing a weight function with a few large peaks produces a regrading dependent only on the relative

TABLE 1a

		New grades							
			Smoothing procedure 1			Smoothing procedure 2			
Old grades	Weights	E $p=0$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	U $p=1$
1	9.0	1	1	1	1	1	1	1	1
2	8.0	2	1	1	1	2	1	1	1
3	7.0	2	2	2	2	2	2	2	1
4	6.0	3	3	2	2	3	3	2	2
5	5.0	3	3	3	2	3	3	3	2
6	4.0	3	3	3	3	3	3	3	3
7	3.0	4	4	3	3	4	4	3	3
8	2.0	4	4	4	3	4	4	4	3
9	1.0	4	4	4	4	4	4	4	4
10	0.0	4	4	4	4	4	4	4	4
11	1.0	4	4	4	5	4	4	4	5
12	2.0	4	4	5	5	4	4	5	5
13	3.0	4	5	5	5	5	5	5	5
14	4.0	5	5	5	6	5	5	5	6
15	5.0	5	5	6	6	5	5	6	6
16	6.0	6	6	6	7	6	6	6	7
17	7.0	6	6	7	7	6	6	6	7
18	8.0	7	7	7	7	7	7	7	7
19	9.0	7	7	8	8	7	7	8	8
20	10.0	8	8	8	8	8	8	8	8

TABLE 1b

New grades	New weights							
	E	Smoothing procedure 1			Smoothing procedure 2			U
	$p=0$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	$p=1$
1	9.0	17.0	17.0	17.0	9.0	17.0	17.0	24.0
2	15.0	7.0	13.0	18.0	15.0	7.0	13.0	11.0
3	15.0	15.0	12.0	9.0	15.0	15.0	12.0	9.0
4	12.0	9.0	4.0	1.0	9.0	9.0	4.0	1.0
5	9.0	12.0	9.0	6.0	12.0	12.0	9.0	6.0
6	13.0	13.0	11.0	9.0	13.0	13.0	18.0	9.0
7	17.0	17.0	15.0	21.0	17.0	17.0	8.0	21.0
8	10.0	10.0	19.0	19.0	10.0	10.0	19.0	19.0

TABLE 1c

E	Breakpoints						
	Smoothing procedure 1			Smoothing procedure 2			U
$p=0$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	$p=1$
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.44	1.54	1.83	2.34	1.49	1.55	1.70	2.5
3.17	3.44	4.25	4.95	3.32	3.46	3.86	5.0
5.63	6.31	7.38	7.56	5.96	6.35	6.88	7.5
12.67	11.50	10.50	10.17	12.07	11.56	10.94	10.0
15.42	15.01	13.63	12.78	15.19	14.97	14.29	12.5
17.25	17.05	16.39	15.38	17.14	17.03	16.69	15.0
18.72	18.63	18.38	17.93	18.67	18.63	18.49	17.5
20.0	20.0	20.0	20.0	20.0	20.0	20.0	20.0

TABLE 2

Old grades	Weights	New grades	New weights	Breakpoints
1	13.0	1	13	1.0
2	7.0	2		
3	6.0	2	13	3.0
4	5.0	3		
5	4.0	3		
6	4.0	3	13	6.0
7	4.0	4		
8	4.0	4		
9	3.0	4		
10	2.0	4	13	10.0
11	1.0	5		
12	2.0	5		
13	3.0	5		
14	3.0	5		
15	4.0	5	13	15.0
16	6.0	6		
17	7.0	6	13	17.0
18	6.0	7		
19	7.0	7	13	19.0
20	13.0	8	13	

TABLE 3
W = weights, NG = new grades

Old grades	One peak						Two equal peaks						Two unequal peaks								Three peaks			
	W	NG	W	NG	W	NG	W	NG	W	NG	W	NG	W	NG	W	NG	W	NG	W	NG	W	NG	W	NG
1	10	3	0	1	0	1	10	2	10	2	0	1	10	2	10	2	5	1	0	1	20	2	20	2
2	0	5	0	1	0	1	0	3	0	3	0	1	0	4	0	4	0	2	0	1	4	3	0	3
3	0	5	0	1	0	1	0	3	0	3	0	1	0	4	0	4	0	2	0	1	0	3	0	3
4	0	5	0	1	10	3	0	3	0	3	0	1	0	4	0	4	0	2	0	1	0	3	0	3
5	0	5	0	1	0	5	0	3	10	4	0	1	0	4	5	5	0	2	5	1	20	4	0	3
6	0	5	0	1	0	5	0	3	0	5	0	1	0	4	0	5	0	2	10	4	0	5	4	3
7	0	5	0	1	0	5	0	3	0	5	0	1	0	4	0	5	0	2	0	5	0	5	0	3
8	0	5	0	1	0	5	0	3	0	5	0	1	0	4	0	5	0	2	0	5	0	5	0	3
9	0	5	0	1	0	5	0	3	0	5	10	2	0	4	0	5	0	2	0	5	0	5	20	4
10	0	5	10	3	0	5	10	4	0	5	10	4	5	5	0	5	10	4	0	5	0	5	0	5

TABLE 4
PW = padded weights, NG = new grades.

Old grades	Weights	Smoothing procedure 1						Smoothing procedure 2					
		$p=\frac{1}{4}$		$p=\frac{1}{2}$		$p=\frac{3}{4}$		$p=\frac{1}{4}$		$p=\frac{1}{2}$		$p=\frac{3}{4}$	
		PW	NG	PW	NG	PW	NG	PW	NG	PW	NG	PW	NG
1	20.0	20.0	$\underline{1}$	20.0	$\underline{1}$	20.0	1	20.0	$\underline{1}$	20.0	$\underline{1}$	20.0	$\underline{1}$
2	0.0	2.9	2	7.2	2	12.9	$\underline{1}$	4.82	2	7.77	2	12.09	2
3	0.0	2.9	$\underline{2}$	7.2	2	12.9	2	2.74	$\underline{2}$	4.82	$\underline{2}$	8.67	2
4	0.0	2.9	3	7.2	$\underline{2}$	12.9	$\underline{2}$	1.91	3	3.49	3	6.75	$\underline{2}$
5	0.0	2.9	3	7.2	3	12.9	3	2.45	3	3.04	3	5.53	3
6	4.0	4.0	3	7.2	$\underline{3}$	12.9	$\underline{3}$	4.00	3	4.00	3	6.75	3
7	0.0	2.9	$\underline{3}$	7.2	4	12.9	4	2.74	$\underline{3}$	4.82	$\underline{3}$	8.67	$\underline{3}$
8	0.0	2.9	4	7.2	$\underline{4}$	12.9	$\underline{4}$	4.82	4	7.77	4	12.09	$\underline{4}$
9	20.0	20.0	$\underline{4}$	20.0	5	20.0	5	20.00	$\underline{4}$	20.00	$\underline{4}$		5
10	0.0	2.9	5	7.2	5	12.9	5	4.82	5	7.77	5	12.07	5

size and order of the peaks, but independent of their particular positions. Illustrations are shown in Table 3, which summarizes the results of equidistributing 10 grades, with a variety of simple extreme weight functions, into 5 grades. For a single peak the regrading always shifts the peak weight into grade 3, and for two equal peaks, the peak weights are always regraded into grades 2 and 4. If there are two unequal peaks, their new positions depend on their relative sizes, but not their original positions. Similar results are seen to hold in the case of weight functions with three peaks.

In Table 4 we show the effect of smoothing an extreme weight function containing three peaks. The equidistributing regrading for this function is given in the last column of Table 3. We observe that the weight originally in grade 1 is shifted into new grade 2. The smooth regradings however, place this peak weight, more reasonably, in grade 1. As the parameter p increases from zero, the padding increases, and the equidistributing grading is transformed into the linear grading. The peak weight in grade 4 is thus eventually moved into grade 5. We note that the *padding* of procedure 1 is constant between the peaks, covering the central peak for $p=\frac{1}{2}$, $\frac{3}{4}$, and that with $p=\frac{3}{4}$ the regrading is already linear in this case. For procedure 2, the padding drops away smoothly from each peak, the central peak being covered by the padding only for $p=\frac{3}{4}$. In this case the linear regrading is only produced when p is close to unity.

The difference in the behavior of the two smoothing procedures is more explicitly illustrated in Figures 1 and 2. A weight function with three sharp peaks on 95 grades is regraded into 9 grades. On a scale from [0, 1], the peak weights occur at points 0, 0.3 and 1.0. For smoothing procedures 1 and 2, the positions of the 10 (scaled) breakpoints $0 = x_0 < x_1 \cdots < x_9 = 1.0$ are shown in the figures as continuous functions of p, for p from 0 to 1.0.

In the smoothing produced by smoothing procedure 1, (Fig. 1) there are fairly sharp changes in the positions of the breakpoints. For small p, each breakpoint changes very little as p increases, until the constant padding reaches some critical level; then the breakpoint moves rapidly into its position in the linear distribution. By contrast, the behavior of the breakpoints produced by smoothing procedure 2 (Fig. 2) is essentially smoother. All the breakpoints move gradually from the equidistributing distribution towards positions close to the uniform distribution.

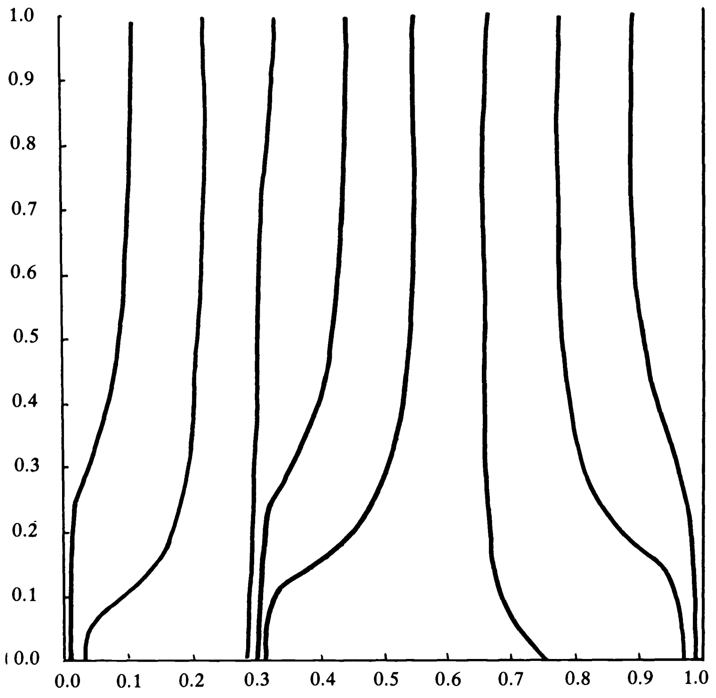

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

FIG. 1

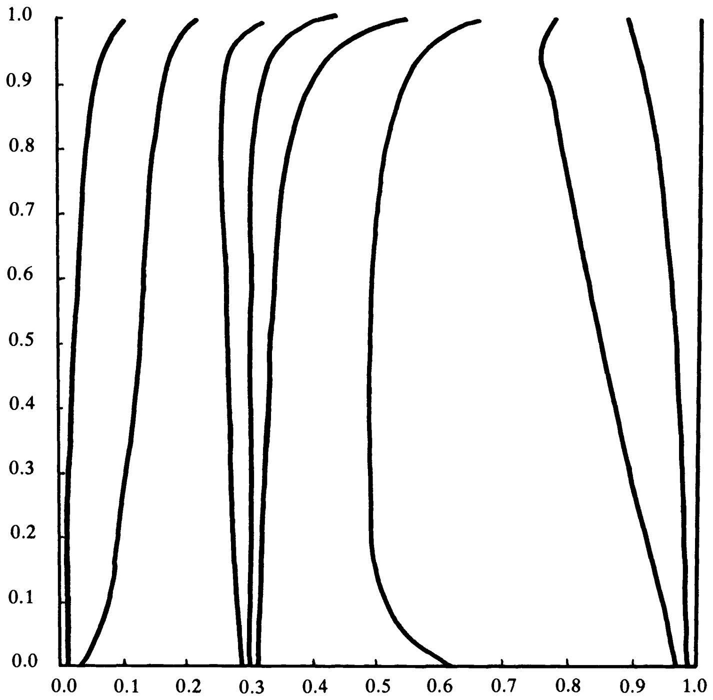

1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

FIG. 2

With smoothing procedure 1, we also observe that the uniform regrading is achieved fairly quickly (in this case for $p \geqq 0.5$); while with smoothing procedure 2, the character of the initial distribution of the points is retained for fairly large values of p ($p \leqq 0.8$, here), and the final transition to the linear regrading is somewhat abrupt. These characteristics are accentuated by the sharpness of the peaks, and for smooth weight functions the behavior of both procedures is more modulated.

Complete results for the examples described in this section are given in [5].

6. Weighted regradings. In this section we describe an application of the procedures given in §§ 3 and 4 to a more general regrading problem. In § 2 we introduced the concept of an *equidistributing* regrading as a regrading which transforms a given weight function, or histogram, into an (approximately) constant weight function. We now define a *weighted* regrading as one which transforms a given weight function into another prescribed weight function called a *reference* function. The equidistributing regrading is obviously a special case where the prescribed reference function is constant. Generally, the reference function may be prescribed on a different number of grades from that of the data being regraded.

For a more precise statement of the problem, let m, n, q be integers, and let $\{f_1, f_2, \cdots, f_n\}$ and $\{w_1, w_2, \cdots, w_q\}$ be nonnegative weights associated with sets of n and q grades, respectively. Also let $\{w_1^*, w_2^*, \cdots, w_m^*\}$ be the set of transformed weights obtained from weights w_j, $j = 1, 2, \cdots, q$, by a linear regrading from q into m grades. We seek a regrading $\boldsymbol{\tau}$ from n into m grades with respect to weights f_j, $j = 1, 2, \cdots, n$, such that the transformed weights $\{f_1^*, f_2^*, \cdots, f_m^*\}$ are (approximately) equal to w_j^*, $j = 1, 2, \cdots, m$.

To give an explicit definition of $\boldsymbol{\tau}$ we proceed as in § 3. We let F be a piecewise linear function on $[0, n]$ such that

$$(19) \qquad F(0) = 0, \qquad F(j) = \sum_{i=1}^{j} f_i, \qquad j = 1, 2, \cdots, n,$$

and

$$F'(t) = f_j \quad \text{for } t \in (j-1, j).$$

Similarly we define W to be a piecewise linear function on $[0, q]$ such that

$$(20) \qquad W(0) = 0, \qquad W(j) = \sum_{i=1}^{j} w_0, \qquad j = 1, 2, \cdots, 1,$$

and

$$W'(t) = w_j \quad \text{for } t \in (j-1, j).$$

Breakpoints $x_k \in [0, n]$ are then determined by

$$(21) \qquad F(x_m) = \frac{W(kq/m)F(n)}{W(q)}, \qquad k = 0, 1, \cdots, m,$$

and we have the following definition:

DEFINITION 3. Given integers n, q and m, and nonnegative weights $\{f_1, f_2, \cdots, f_n\}$ and $\{w_1, w_2, \cdots, w_q\}$, let the breakpoints $x_0, x_1, \cdots, x_m$ be given by (21), where F and W are specified by (19) and (20) respectively. Then the regrading $\boldsymbol{\tau}$ satisfying

$$\tau_j = k \quad \text{for all } j \text{ such that } x_{k-1} < j - \tfrac{1}{2} \leqq x_k$$

is said to *distribute* weights f_j, $j=1,2,\cdots,n$ with respect to reference weights w_j, $j=1,2,\cdots,q$, and $\boldsymbol{\tau}$ is called a *weighted regrading.*

A direct procedure for constructing the weighted regrading $\boldsymbol{\tau}$ is easy to implement using inverse linear interpolation to find the breakpoints x_k defined by (21). We note, however, that with the following theorem we may transform the weighted distribution problem into an equidistribution problem and then apply the procedure of § 3 to determine the weighted regrading.

THEOREM. *Let* $w_j>0$, $j=1,2,\cdots,q$, *and define*

$$H(x)=W^{-1}\left(\frac{F(x)W(q)}{F(n)}\right),\qquad x\in[0,n]. \tag{22}$$

Then there exist $g_j\geqq 0$, $j=1,2,\cdots,3n$, *such that*

$$\sum_{i=1}^{3n} g_i=q,\qquad \tfrac{1}{2}g_{3j-1}+\sum_{i=1}^{3j-2} g_i=H(j-\tfrac{1}{2}),\qquad j=1,2,\cdots,n. \tag{23}$$

Let $\boldsymbol{\sigma}$ *be a* $3n$ *to* m *regrading which equidistributes weights* g_j, $j=1,2,\cdots,3n$. *Then* $\boldsymbol{\tau}$ *given by*

$$\tau_j=\sigma_{3j-1},\qquad j=1,2,\cdots,n \tag{24}$$

is the weighted regrading which distributes the weights f_j, $j=1,2,\cdots,n$ *with respect to reference weights* w_j, $j=1,2,\cdots,q$.

Proof. As $w_j>0$, the inverse function of W exists and the definition (21) of the breakpoints x_k is equivalent to

$$H(x_k)=\frac{kq}{m},\qquad k=0,1,\cdots,m. \tag{25}$$

The function H is a nondecreasing piecewise linear function mapping $[0,n]$ into $[0,q]$ with corners at points j, $j=1,2,\cdots,n-1$ and at points t_i where $H(t_i)=i$, $i=1,2,\cdots,q-1$. The weighted regrading $\boldsymbol{\tau}$ of Definition 3 is uniquely determined by the relative positions of points x_k and points $j-\frac{1}{2}$, but is independent of the actual values of the breakpoints x_k as long as the relative ordering of the points is preserved.

Thus the same regrading $\boldsymbol{\tau}$ is obtained with the set of breakpoints x_k which satisfy

$$G(3x_k)=\frac{kq}{m},\qquad k=0,1,2,\cdots,m, \tag{26}$$

where G is *any* nondecreasing function from $[0,3n]$ to $[0,q]$ such that

$$G(0)=0,\quad G(3n)=q,\quad G(3(j-\tfrac{1}{2}))=H(j-\tfrac{1}{2}),\qquad j=1,2,\cdots,n. \tag{27}$$

In particular, if G is the piecewise linear function with corners at points $1,2,\cdots,3n-1$ such that

$$G'(t)=g_j,\qquad t\in(j-1,j),\quad j=1,2,\cdots,3n, \tag{28}$$

then G satisfies (27) by the definition of the weights g_j given in the theorem. Furthermore, the regrading $\boldsymbol{\sigma}$ from $[0,3n]$ to $[0,m]$ which equidistributes the weights g_j, $j=1,2,\cdots,3n$ is determined by breakpoints $3x_k$ satisfying (26). A direct application of Definition 2 then gives that $x_{k-1}<j-\frac{1}{2}\leqq x_k$ implies $\tau_j=\sigma_{3j-1}=k$, and hence $\boldsymbol{\tau}$ is the required weighted regrading.

The existence of the weights g_j, $j=1,2,\cdots,3n$ is shown by explicit construction.

We may choose, for example,

$$g_1 = H(\tfrac{1}{2}), \qquad g_{3n} = q - H(n-\tfrac{1}{2}),$$
$$g_{3j-1} = 0, \qquad j = 1, 2, \cdots, n,$$
$$g_{3j} = g_{3j+1} = \tfrac{1}{2}(H(j+\tfrac{1}{2}) - H(j-\tfrac{1}{2})), \qquad j = 1, 2, \cdots, n-1,$$

and the theorem is proved.

We conclude that a generalized *weighted* regrading may be obtained by the equidistributing procedure of § 3. Unfortunately the weighted regradings cannot be determined *exactly* by an equidistributing regrading from n to m grades; however, *approximations* to the weighted regradings could be obtained by equidistributing an appropriate choice of n weights based on the theorem. Such procedures are discussed further in [5].

We observe that a variety of *smooth weighted* regradings could be obtained by applying the padding procedures of § 4 to the weights g_j, $j = 1, 2, \cdots, 3n$, defined in the theorem, and equidistributing the padded weights. The weights g_j satisfying (23) are not uniquely determined, however, and the resulting smoothed regradings are affected by the choice of these weights.

A conceptually simpler method for obtaining a range of smooth regradings between a *weighted* regrading and a *linear* regrading is based on the fact that if a given weight function is distributed with respect to itself, the resulting regrading is linear. Thus a smoothing is achieved by reshaping either the reference function or the original weight function, or both, so that they are ultimately equal, possibly to another given function. The simplest such reshaping uses (one-parameter dependent) linear combinations of the given weights:

Smoothing procedure 3. Let the vectors **f**, **y** represent, respectively, the nonnegative weights f_j, y_j, $j = 1, 2, \cdots, n$ and let vectors **w**, $\tilde{\mathbf{y}}$, represent the weights w_j, $\tilde{y}_j$, $j = 1, 2, \cdots, q$. For $p \in [0, 1]$ replace **f** and **w** by new weights

$$(29) \qquad \begin{aligned} (\mathbf{f}_p)_j &= (1-p)f_j + py_j, \qquad j = 1, 2, \cdots, n, \\ (\mathbf{w}_p)_j &= (1-p)w_j + p\tilde{y}_j, \qquad j = 1, 2, \cdots, q. \end{aligned}$$

If $q = n$, then $\tilde{y}_j = y_j$, $j = 1, 2, \cdots, n$, is taken, and **y** is arbitrary. If $q \neq n$, then $\tilde{y}_j = c$, $j = 1, 2, \cdots, q$ and $y_j = c$, $j = 1, 2, \cdots, n$, is taken, where c is an arbitrary constant with $c > 0$. (A more general choice is possible, but **y** and $\tilde{\mathbf{y}}$ must together satisfy certain constraints. The generalizations are discussed further in [5].)

A *smoothing* of the weighted regrading of **f** with respect to reference w is given by the regrading from n into m grades which distributes the weights $(\mathbf{f}_p)_j$, $j = 1, 2, \cdots, n$, with respect to weights $(\mathbf{w}_p)_j = 1, 2, \cdots, q$. Clearly, when $p = 0$ the weighted regrading of Definition 3 is obtained, and when $p = 1$, the linear regrading from n into m grades results.

Remarks. 1) In the case $q = n$, the choice of $\mathbf{y} \equiv \tilde{\mathbf{y}}$ is arbitrary and may be taken, for example, to be either **f** or **w**. In the former case $\mathbf{w}_0 \equiv \mathbf{w}$, $\mathbf{w}_1 = \mathbf{f}$, and $\mathbf{f}_p \equiv \mathbf{f}$, $\forall p$; that is, **w** is reshaped linearly into **f**. In the latter case, **f** is reshaped linearly into **w**, and $\mathbf{f}_0 \equiv \mathbf{f}$, $\mathbf{f}_1 \equiv \mathbf{w}$, $\mathbf{w}_p \equiv \mathbf{w}$, $\forall p$.

2) If the reference weights are all constant, i.e., $w_j = c$, $j = 1, 2, \cdots, q$, then procedure 3 gives an additional method to those of § 4 for smoothing the regrading which *equidistributes* the weights f_j, $j = 1, 2, \cdots, n$.

The algorithms for determining smooth weighted regradings are implemented in simple FORTRAN subroutines (see [5]). As an example, we consider a weighted

TABLE 5a

Old grades	Weights	Reference weights	New grades $p=0$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	$p=1$
1	9.0	0.0	2	1	1	1	1
2	8.0	1.0	3	2	2	1	1
3	7.0	2.0	3	3	2	2	1
4	6.0	3.0	4	3	3	2	2
5	5.0	4.0	4	4	3	3	2
6	4.0	5.0	4	4	4	3	3
7	3.0	6.0	4	4	4	4	3
8	2.0	7.0	4	4	4	4	3
9	1.0	8.0	4	4	4	4	4
10	0.0	9.0	4	4	4	4	4
11	1.0	10.0	5	5	5	5	5
12	2.0	9.0	5	5	5	5	5
13	3.0	8.0	5	5	5	5	5
14	4.0	7.0	5	5	5	5	6
15	5.0	6.0	5	5	5	6	6
16	6.0	5.0	5	5	6	6	7
17	7.0	4.0	6	6	6	7	7
18	8.0	3.0	6	6	7	7	7
19	9.0	2.0	7	7	7	8	8
20	10.0	1.0	7	8	8	8	8

TABLE 5b

New grades	Reference weights	New weights $p=0$	$p=\frac{1}{4}$	$p=\frac{1}{2}$	$p=\frac{3}{4}$	$p=1$
1	3.0	0.0	9.0	9.0	17.0	24.0
2	7.0	9.0	8.0	15.0	13.0	11.0
3	18.0	15.0	13.0	11.0	9.0	9.0
4	17.0	21.0	15.0	10.0	6.0	1.0
5	27.0	21.0	21.0	15.0	10.0	6.0
6	13.0	15.0	15.0	13.0	11.0	9.0
7	12.0	19.0	9.0	17.0	15.0	21.0
8	3.0	0.0	10.0	10.0	19.0	19.0

regrading from 20 into 8 grades which distributes the V-shaped weight function given in § 5 with respect to a Λ-shaped reference function defined explicitly by

$$w_j = 10 - |11 - j|, \qquad j = 1, 2, \cdots, 20.$$

The results are summarized in Tables 5a, b. The regradings obtained with smoothing procedure 3 for various values of p and $\mathbf{y} \equiv \mathbf{f}$ are shown in Table 5a, and the transformed weights are shown in Table 5b. The weighted regrading gives a new histogram of the required shape, and a gradual change into the linear regrading is obtained by the smoothing. Further examples and detailed results are presented in [5].

7. Conclusions. Simple explicit methods for histogram modification are presented here. The methods construct weighted regradings, or transformations, of discrete data such that the histogram of data occurrences is transformed into a prescribed function. Equidistributing regradings which flatten the histogram, i.e., transform it into a constant function, are examined in detail. Methods for producing a range of "smooth" regradings between a weighted regrading and a linear transformation are also discussed. The use of smoothing mollifies the difficulties which arise in regrading data with extremely unevenly distributed histograms.

The technique of histogram modification is widely used for image processing. The methods described here are currently being used by CSIRO Division of Land Use Research for calibration, enhancement and normalization of LANDSAT image data. Applications and preliminary results are published in [6]. The smoothing techniques described here have proved particularly valuable for identifying and enhancing special features. Further results will be presented in a forthcoming paper.

Acknowledgments. The authors wish to thank Dr D. B. Jupp of CSIRO, Canberra, Australia for bringing to their attention the problems of discrete regradings and the need for smoothing techniques.

REFERENCES

[1] R. M. HARALICK, *Automatic remote sensor image processing*, Topics in Applied Physics, 11 (1976), pp. 5–63.

[2] R. A. HUMMEL, *Histogram modification techniques*, Comput. Gr. Image Process., 4 (1975), pp. 209–224.

[3] ———, *Image enhancement by histogram modification*, Comput. Gr. Image Process., 6 (1977), pp. 184–195.

[4] J. KAUTSKY AND N. K. NICHOLS, *Equidistributing meshes with constraints*, SIAM J. Sci. Stat. Comp., 1 (1980), pp. 499–511.

[5] ———, *Smooth regrading of discretized data*, Department of Mathematics Numerical Analysis Rpt. 1/80, University of Reading, England, 1980.

[6] J. KAUTSKY, D. L. B. JUPP AND N. K. NICHOLS, *Image enhancement by smoothed histogram modification*, Proc. 2nd Australian Remote Sensing Conference, LANDSAT 81, Canberra, 1981, pp. 6.6.1–6.6.5.

[7] J. MAX, *Quantizing for minimum distation*, Trans. IRE, IT-6 (1960), pp. 7–12.

[8] V. PEREYRA AND E. G. SEWELL, *Mesh selection for discrete solution of boundary problems in ordinary differential equations*, Num. Math., 23 (1975), pp. 261–268.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

A LAGRANGE EXTRAPOLATION ALGORITHM FOR SEQUENCES OF APPROXIMATIONS TO MULTIPLE INTEGRALS*

ALAN C. GENZ†

Abstract. An algorithm for multivariable Lagrange interpolation is described and applied to the problem of extrapolating sequences of approximations to multiple integrals. The new algorithm is then compared with a recursive extrapolation algorithm in terms of required time and storage, and stability. A Fortran subroutine is given for computing extrapolated sequences of approximations to multiple integrals using the new algorithm.

Key words. multiple integration, Lagrange interpolation, extrapolation.

1. Introduction. The purpose of this paper is to describe an algorithm for multidimensional interpolation and the use of the algorithm for the extrapolation of sequences of approximations to multidimensional integrals. This will be compared with the use of a multidimensional recursive interpolation algorithm described by McKinney [4]. In this introduction the motivation for the use of the algorithm is given, along with some basic definitions.

Let $\{A_{\mathbf{m}}\}$ be a sequence of real numbers with index $\mathbf{m}=(m_1, m_2, \cdots, m_n)$ where $m_i=0, 1, 2, \cdots$. Let $|\mathbf{m}|=\sum_{i=1}^n m_i$ and assume that we have some ordering of the integer n-tuples $\mathbf{m}$ such that if $|\mathbf{m}|>|\mathbf{k}|$ then $\mathbf{m}$ comes after $\mathbf{k}$ in the chosen ordering. We denote $(0, 0, \cdots, 0)$ by $\mathbf{0}$.

We are concerned here with sequences $\{A_{\mathbf{m}}\}$ which have the asymptotic form

$$A_{\mathbf{m}}=A+\sum_{1\leq|\mathbf{k}|\leq d} c_{\mathbf{k}}\mathbf{x}_{\mathbf{m}}^{\mathbf{k}}+\sum_{|\mathbf{k}|=d+1} O(\mathbf{x}_{\mathbf{m}}^{\mathbf{k}}),$$

where $\mathbf{x}_{\mathbf{m}}=(x_{m_1}, x_{m_2}, \cdots, x_{m_n})$ and $\mathbf{x}_{\mathbf{m}}^{\mathbf{k}}=\prod_{i=1}^n x_{m_i}^{k_i}$, for some positive mesh sequence $\{x_m\}$ decreasing with limit zero, so that A will be the limit of the sequence $\{A_{\mathbf{m}}\}$.

The sequence of this type which we consider in this paper is obtained by applying the product midpoint rule to a sufficiently smooth function $f(\mathbf{x})$ in order to estimate

$$I(f)=\int_0^1\int_0^1\cdots\int_0^1 f(\mathbf{x})\,dx_1\,dx_2\cdots d_{x_n}.$$

In this case we use $A_{\mathbf{m}}=M_{\mathbf{m}}(f)$, where

$$M_{\mathbf{m}}(f)=\prod_{i=1}^n (m_i+1)^{-1}\sum_{k_1=0}^{m_1}\sum_{k_2=0}^{m_2}\cdots\sum_{k_n=0}^{m_n} f\left(\frac{2k_1+1}{2m_1+2}, \frac{2k_2+1}{2m_2+2}, \cdots, \frac{2k_n+1}{2m_n+2}\right),$$

and then $A=I(f)$ with $x_m=(1+m)^{-2}$.

The use of McKinney's recursive interpolation algorithm to extrapolate a sequence $\{A_{\mathbf{m}}\}$ was described in [1], where the strategy is similar to the one used when the Neville–Aitken interpolation algorithm is used in the Romberg integration method for one-variable functions. Because a similar method using a different interpolation algorithm will be discussed in § 2, we describe in some detail McKinney's algorithm and its application to extrapolation.

We use McKinney's ordering, denoted by $\subset$, for the sequence $\{\mathbf{m}\}$. This ordering gives $\mathbf{m}$ as the successor to $\mathbf{m}'$ using

$$\mathbf{m}=(m_1'+1, m_2'-1, m_3', \cdots, m_n')\quad \text{if } m_2'\neq 0,$$

* Received by the editors June 10, 1980, and in revised form April 15, 1981.

† Mathematical Institute, University of Kent, Canterbury, Kent, CT2 7NF, England.

or

$$\mathbf{m} = (0, 0, \cdots, 0, m'_j + 1, m'_{j+1} - 1, m'_{j+2}, \cdots, m'_n) \quad \text{if } m_2 = m_3 = \cdots = m_j = 0.$$

We define polynomials $\phi_{\mathbf{m}}(\mathbf{y})$ by

$$\phi_{\mathbf{m}}(\mathbf{y}) = \prod_{i=1}^{n} \prod_{l=0}^{m_i-1} (y_i - x_l),$$

using the convention that the inner product is equal to one if the upper index limit is negative. Clearly the degree of $\phi_{\mathbf{m}}$ is $|\mathbf{m}|$; it can also be shown that $\phi_{\mathbf{m}}(\mathbf{x_k}) = 0$ for any $\mathbf{k} \subset \mathbf{m}$.

Now if $g(\mathbf{x})$ is any function defined for the sequence $\{\mathbf{x_m}\}$, then a sequence of interpolating polynomials $\{R_{\mathbf{m}}(\mathbf{x})\}$ can be defined using

$$R_{\mathbf{m}}(\mathbf{x}) = R_{\mathbf{m}'}(\mathbf{x}) + a_{\mathbf{m}}\phi_{\mathbf{m}}(\mathbf{x}), \tag{1.1}$$

starting with $R_0(\mathbf{x}) = g(\mathbf{x_0})$ and defining $a_{\mathbf{m}}$ by

$$a_{\mathbf{m}} = \frac{g(\mathbf{x_m}) - R_{\mathbf{m}'}(\mathbf{x_m})}{\phi_{\mathbf{m}}(\mathbf{x_m})}. \tag{1.2}$$

Clearly, $R_{\mathbf{m}}(x)$ is a polynomial of degree at most $|\mathbf{m}|$ and induction easily shows that $R_{\mathbf{m}}(\mathbf{x_k}) = g(\mathbf{x_k})$ for $\mathbf{k} \subseteq \mathbf{m}$.

In order to eliminate $a_{\mathbf{m}}$ we expand (1.2) to give

$$a_{\mathbf{m}} = \frac{g(\mathbf{x_m}) - \sum_{\mathbf{k}\subset\mathbf{m}} a_{\mathbf{k}}\phi_{\mathbf{k}}(\mathbf{x_m})}{\phi_{\mathbf{m}}(\mathbf{x_m})},$$

and then substitute $a_{\mathbf{k}}$ obtained from (1.1) in the form

$$a_{\mathbf{k}} = \frac{R_{\mathbf{k}}(\mathbf{x}) - R_{\mathbf{k}'}(\mathbf{x})}{\phi_{\mathbf{k}}(\mathbf{x})},$$

into the sum, to give finally

$$a_{\mathbf{m}} = \left(\frac{g(\mathbf{x_m}) - \sum_{\mathbf{k}\subset\mathbf{m}} (R_{\mathbf{k}}(\mathbf{x}) - R_{\mathbf{k}'}(\mathbf{x}))\phi_{\mathbf{k}}(\mathbf{x_m})}{\phi_{\mathbf{k}}(\mathbf{x})} \right) \Big/ \phi_{\mathbf{m}}(\mathbf{x_m}),$$

where $R_{0'} = 0$.

For extrapolation, we are only interested in $\mathbf{x} = \mathbf{0}$, so we simplify notation by using $R_{\mathbf{m}} = R_{\mathbf{m}}(\mathbf{0})$ and $A_{\mathbf{m}} = g(\mathbf{x_m})$, and defining

$$\rho_{\mathbf{m},\mathbf{k}} = \frac{\phi_{\mathbf{k}}(\mathbf{x_m})}{\phi_{\mathbf{k}}(\mathbf{0})} = \prod_{i=1}^{n} \prod_{l=0}^{k_i-1} \left(1 - \frac{x_{m_i}}{x_l}\right).$$

This gives the recursive extrapolation formula in its final form as

$$R_{\mathbf{m}} = R_{\mathbf{m}'} + \frac{A_{\mathbf{m}} - \sum_{\mathbf{k}\subset\mathbf{m}} (R_{\mathbf{k}} - R_{\mathbf{k}'})\rho_{\mathbf{m},\mathbf{k}}}{\rho_{\mathbf{m},\mathbf{m}}}. \tag{1.3}$$

If we define $P_d = R_{\mathbf{m}}$ when $\mathbf{m} = (d, 0, 0, \cdots, 0)$, then when $\{A_{\mathbf{m}}\} = \{M_{\mathbf{m}}(f)\}$, P_d is an approximation to $I(f)$ of polynomial degree $2d + 1$.

The algorithm given by formula (1.3) is relatively straightforward to use, but unfortunately requires rapidly increasing amounts of storage and time as d increases. There are $\binom{n+d}{d}$ elements in the sequence $\{R_{\mathbf{m}}\}$ which are needed to obtain P_d and the form of (1.3) requires all elements to be stored for possible future use. For increasing d, $\binom{n+d}{d}$ grows like d^n, so storage could be a significant problem on many computers when n is larger than two or three. Furthermore, it can be shown that a careful implementation of (1.3) which takes account of the zero terms in the sum (they occur when $m_i < k_i$ for at least one i) requires roughly $C\binom{2n+d}{d}$ arithmetic operations ($C=2$ or 3) to find P_d given $\{A_{\mathbf{m}}\}$. When $\{A_{\mathbf{m}}\} = \{M_{\mathbf{m}}(f)\}$ the number of integrand evaluations needed for $\{M_{\mathbf{m}}(f)\}$ with $|\mathbf{m}| \leqq d$ is given by

$$\sum_{|\mathbf{m}|\leqq d} \prod_{i=1}^{n} (m_i+1) = \binom{2n+d}{d},$$

so the extrapolation requires the same order amount of time as do the integrand evaluations. Other applications of (1.3) could give rise to situations where the extrapolation process dominates.

These time and space complexity considerations lead to the investigation of methods which use symmetry to reduce the time and space required for the computation of P_d. In the next section we present a generalization of the Lagrange interpolation method, which may be applied to this type of extrapolation problem, and then describe the resulting algorithm which exploits the symmetry of the problem to significantly reduce the time and space required to compute P_d. A more efficient, symmetrized version of McKinney's algorithm is described in § 3, and in the final section the algorithms are compared and a brief description is given for the Fortran subroutine, listed at the end of this paper, which uses the new algorithm to compute extrapolated approximations to multiple integrals.

2. Multivariable Lagrange interpolation.

We wish to find polynomials $\Phi_{\mathbf{m}}^{(d)}(\mathbf{x})$ of total degree d, when $0 \leqq |\mathbf{m}| \leqq d$, satisfying

$$\Phi_{\mathbf{m}}^{(d)}(\mathbf{x}_{\mathbf{k}}) = \delta_{\mathbf{m},\mathbf{k}} \quad \text{for } 0 \leqq |\mathbf{k}| \leqq d,$$

where

$$\delta_{\mathbf{m},\mathbf{k}} = \begin{cases} 0, & \mathbf{m} \neq \mathbf{k}, \\ 1, & \mathbf{m} = \mathbf{k}. \end{cases}$$

Then the multiple Lagrange interpolating polynomial for any function $g(\mathbf{x})$ defined on $\{\mathbf{x}_{\mathbf{k}}\}$ is given by

$$P_d(\mathbf{x}) = \sum_{|\mathbf{m}|\leqq d} g(\mathbf{x}_{\mathbf{m}})\Phi_{\mathbf{m}}^{(d)}(\mathbf{x}). \tag{2.1}$$

We will construct $\Phi_{\mathbf{m}}^{(d)}(\mathbf{x})$ from a divided difference representation. We assume that $\{x_m\}$ is a sequence of distinct real numbers and let $[\mathbf{x}_{\mathbf{k}}] = (x_0, x_1, \cdots, x_{k_1}; x_0, x_1, \cdots, x_{k_2}; \cdots; x_0, x_1, \cdots, x_{k_n})$. Then $D[\mathbf{x}_{\mathbf{k}}]$, the multivariable divided difference operator, may be explicitly given in the form

$$D[\mathbf{x}_{\mathbf{k}}]g = \sum_{j_1=0}^{k_1} \sum_{j_2=0}^{k_2} \cdots \sum_{j_n=0}^{k_n} g(\mathbf{x}_{\mathbf{j}}) \prod_{i=1}^{n} \prod_{l=0,\neq j_i}^{k_i} (x_{j_i} - x_l)^{-1}.$$

If we apply $D[\mathbf{x}_{\mathbf{k}}]$ to $\delta_{\mathbf{m},\mathbf{k}}$ we find

$$D[\mathbf{x}_{\mathbf{k}}]\delta_{\mathbf{m},\mathbf{k}} = \begin{cases} \prod_{i=1}^{n} \prod_{l=0,\neq m_i}^{k_i} (x_{m_i} - x_l)^{-1} & \text{if } k_i \geqq m_i \text{ for all } i, \\ 0 & \text{otherwise.} \end{cases}$$

We then express $\Phi_{\mathbf{m}}^{(d)}(\mathbf{y})$ in its multivariable divided difference form (by generalizing the two-dimensional formula in [3]) as

$$\Phi_{\mathbf{m}}^{(d)}(\mathbf{y}) = \sum_{|\mathbf{k}|\leqq d}\left(\prod_{i=1}^{n}\prod_{l=0}^{k_i-1}(y_i-x_l)\right)D[\mathbf{x_k}]\,\delta_{\mathbf{m},\mathbf{k}}.$$

The only nonzero terms in the sum are those with $k_i \geqq m_i$ for all i, so we make a slight change of index and write $\Phi_{\mathbf{m}}^{(d)}(\mathbf{y})$ as

$$\Phi_{\mathbf{m}}^{(d)}(\mathbf{y}) = \sum_{|\mathbf{k}|\leqq d-|\mathbf{m}|}\prod_{i=1}^{n}\frac{\prod_{l=0}^{m_i+k_i-1}(y_i-x_l)}{\prod_{l=0,\neq m_i}^{m_i+k_i}(x_{m_i}-x_l)}. \tag{2.2}$$

For the purposes of extrapolation we are only interested in $P_d(\mathbf{0})$, so we simplify notation by using $P_d \equiv P_d(\mathbf{0})$ and $\Phi_{\mathbf{m}}^{(d)} \equiv \Phi_{\mathbf{m}}^{(d)}(\mathbf{0})$. From the symmetry in (2.2) we can see that $\Phi_{\mathbf{m}}^{(d)} = \Phi_{\mathbf{k}}^{(d)}$ whenever $\mathbf{k}$ is a permutation of $\mathbf{m}$. Therefore, the number of distinct values in the sequence $\{\Phi_{\mathbf{m}}^{(d)}\}$ with $0 \leqq |\mathbf{m}| \leqq d$ is at most the number, which we call $N_d^{(n)}$, of distinct n-partitions of the integers $0, 1, 2, \cdots, d$. Let $\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_{N_d^{(n)}}$ be these partitions as they occur in the chosen ordering, and let $Q_{\mathbf{m}}$ be the set of permutations of a partition $\mathbf{m}$. P_d can now be expressed in the form

$$P_d = \sum_{j=1}^{N_d^{(n)}} \Phi_{\mathbf{m}_j}^{(d)} S_{\mathbf{m}_j},$$

where

$$S_{\mathbf{m}_j} = \sum_{\mathbf{k}\in Q_{\mathbf{m}_j}} A_{\mathbf{k}}.$$

Finally we note that $\Phi_{\mathbf{m}}^{(d)}$ depends on $\Phi_{\mathbf{m}}^{(d-1)}$ through the formula

$$\Phi_{\mathbf{m}}^{(d)} = \Phi_{\mathbf{m}}^{(d-1)} + \sum_{|\mathbf{k}|=d-|\mathbf{m}|}\prod_{i=1}^{n}\frac{\prod_{l=0}^{m_i+k_i-1}x_l}{\prod_{l=0,\neq m_i}^{m_i+k_i}(x_l-x_{m_i})}. \tag{2.3}$$

One algorithm for computing P_d from P_{d-1} consists of the following two stages:

i) For $j = 1, 2, \cdots, N_{d-1}^{(n)}$ compute $\Phi_{\mathbf{m}}^{(d)}$ from $\Phi_{\mathbf{m}}^{(d-1)}$ using (2.3) and accumulate the sum

$$D_{d-1} = \sum_{j=1}^{N_{d-1}^{(n)}} \Phi_{\mathbf{m}_j}^{(d)} S_{\mathbf{m}_j}.$$

ii) For $j = N_{d-1}^{(n)}+1, N_{d-1}^{(n)}+2, \cdots, N_d^{(n)}$ calculate $\Phi_{\mathbf{m}_j}^{(d)}$, obtain $S_{\mathbf{m}_j}$ and accumulate the final sum

$$P_d = D_{d-1} + \sum_{j=N_{d-1}^{(n)}+1}^{N_d^{(n)}} \Phi_{\mathbf{m}_j}^{(d)} S_{\mathbf{m}_j}.$$

A second algorithm, which does not require saving the numbers $\Phi_{\mathbf{m}_j}^{(d)}$, gives P_d directly as

$$P_d = P_{d-1} + \sum_{j=1}^{N_d^{(n)}} S_{\mathbf{m}_j} \sum_{\mathbf{k}=d-|\mathbf{m}_j|}\prod_{i=1}^{n}\frac{\prod_{l=0}^{m_{ij}+k_i-1}x_l}{\prod_{l=0,\neq m_{ij}}^{m_{ij}+k_i}(x_l-x_{m_{ij}})}. \tag{2.4}$$

We illustrate the use of the first algorithm with the integral

$$I(f) = \int_0^1\int_0^1\int_0^1 e^{x_1x_2x_3}\,dx_1\,dx_2\,dx_3 = 1.1465.$$

The various intermediate results for $d = 0, 1, 2$, are given in Table 2.1 (to five digits).

TABLE 2.1
Extrapolation example

j	$\mathbf{m}_j$	$S_{\mathbf{m}_j}$	$\Phi^{(d)}_{\mathbf{m}1}$	$\Phi^{(d)}_{\mathbf{m}2}$	$\Phi^{(d)}_{\mathbf{m}3}$	$\Phi^{(d)}_{\mathbf{m}4}$	P_d	d
1	(0, 0, 0)	1.1331	1.0000				1.1331	0
2	(0, 0, 1)	3.4061	−3.0000	1.3333			1.1420	1
3	(0, 0, 2)	3.4073						
4	(0, 1, 1)	3.4148	3.4583	−4.6222	1.7778	2.0250	1.1457	2

3. Symmetrized recursive extrapolation.

We use the formula (1.1) to define

$$b_{\mathbf{m}} = R_{\mathbf{m}} - R_{\mathbf{m}'} = \frac{A_{\mathbf{m}} - \sum_{\mathbf{k}\subset\mathbf{m}} (R_{\mathbf{k}} - R_{\mathbf{k}'})\rho_{\mathbf{m},\mathbf{k}}}{\rho_{\mathbf{m},\mathbf{m}}}$$

and note that

$$P_d = \sum_{|\mathbf{m}|\leq d} (R_{\mathbf{m}} - R_{\mathbf{m}'}) = \sum_{|\mathbf{m}|\leq d} b_{\mathbf{m}} = \sum_{j=1}^{N_d^{(n)}} B_{\mathbf{m}_j}, \tag{3.1}$$

with $B_{\mathbf{m}}$ defined by

$$B_{\mathbf{m}} = \sum_{\mathbf{i}\in Q_{\mathbf{m}}} b_{\mathbf{i}} = \frac{S_{\mathbf{m}} - \sum_{\mathbf{i}\in Q_{\mathbf{m}}} \sum_{\mathbf{k}\subset\mathbf{i}} \rho_{\mathbf{i},\mathbf{k}} b_{\mathbf{k}}}{\rho_{\mathbf{m},\mathbf{m}}}. \tag{3.2}$$

Because $\rho_{\mathbf{m},\mathbf{k}}$ is zero whenever $m_l < k_l$ for at least one l, the double sum in (3.2) may be written in more detail as

$$\sum_{\mathbf{i}\in Q_{\mathbf{m}}} \sum_{k_1=0}^{i_1} w_{i_1,k_1} \sum_{k_2=0}^{i_2} w_{i_2,k_2} \cdots \sum_{k_n=0}^{i_n}{}' w_{i_n,k_b} b_{\mathbf{k}}, \tag{3.3}$$

where the prime indicates that the last term in the sum is excluded, and the weights $w_{m,k}$ are given by

$$w_{m,k} = \prod_{l=0}^{k-1} \left(1 - \frac{x_m}{x_l}\right). \tag{3.4}$$

The multiple sum uses the same set of weights for each $\mathbf{i}\in Q_{\mathbf{m}}$, so if a particular $b_{\mathbf{k}}$ occurs with a particular weight for some $\mathbf{i}$, then all $b_{\mathbf{l}}$, $\mathbf{l}\in Q_{\mathbf{k}}$ must occur at least once with the same weight for some other $\mathbf{i}\in Q_{\mathbf{m}}$. However, every $b_{\mathbf{l}}$ used in the sum for $B_{\mathbf{k}}$ with $\mathbf{k}\subset\mathbf{m}$ does not necessarily occur for each $\mathbf{i}\in Q_{\mathbf{m}}$. But we would like to write the sum (3.3) in terms of $B_{\mathbf{k}}$, so we use $\langle\mathbf{m}\rangle$ to denote the number of distinct permutations of the components of $\mathbf{m}$. Then the sum (3.3) becomes

$$\langle\mathbf{m}\rangle \sum_{k_1=0}^{m_1} w_{m_1,k_1} \sum_{k_2=0}^{m_2} w_{m_2,k_2} \cdots \sum_{k_n=0}^{m_n}{}' w_{m_n,k_n} \frac{B_{\mathbf{k}}}{\langle\mathbf{k}\rangle}.$$

A symmetrized formula for $B_{\mathbf{m}_j}$ is then

$$B_{\mathbf{m}_j} = \left(S_{\mathbf{m}_j} - \langle\mathbf{m}_j\rangle \sum_{\mathbf{k}\subset\mathbf{m}_j} \rho_{\mathbf{m}_j,\mathbf{k}} B_{\mathbf{k}}/\langle\mathbf{k}\rangle\right)\Big/\rho_{\mathbf{m}_j,\mathbf{m}_j}. \tag{3.5}$$

This formula is compact but is unfortunately somewhat difficult to use because the sum is not taken over $B_{\mathbf{k}_j}$. This presents counting problems because only the numbers $B_{\mathbf{k}_j}$ are stored, and so for each $\mathbf{k}$ in the sum we have to determine for which j, $\mathbf{k}\in Q_{\mathbf{k}_j}$.

An alternative formula with more straightforward counting but requiring more arithmetic is

$$(3.6)\qquad B_{\mathbf{m}_j} = \left(S_{\mathbf{m}_j} - \sum_{\mathbf{i}\in Q_{\mathbf{m}_j}} \sum_{j=1}^{N^{(n)}_{|\mathbf{m}_j|-1}} \rho_{\mathbf{i}_j,\mathbf{k}_j} B_{\mathbf{k}_j} \right) \Big/ \rho_{\mathbf{i}_j,\mathbf{i}_j}.$$

The new recursive extrapolation algorithm for computing P_d from P_{d-1} involves using (3.5) or (3.6) to compute $B_{\mathbf{m}_j}$ for $j = N^{(n)}_{d-1} + 1, \cdots, N^{(n)}_d$ and accumulating the sum

$$P_d = P_{d-1} + \sum_{j=N^{(n)}_{d-1}+1}^{d} B_{\mathbf{m}_j}.$$

4. Comparison of the algorithms. Both of the new algorithms have roughly the same storage requirements, which will be dominated, for large n and d by the array space necessary to store the sequences $\{S_{\mathbf{m}_j}\}$ or $\{B_{\mathbf{m}_j}\}$. In order to compute P_d we need an array of size $N^{(n)}_d$. With d fixed, $N^{(n)}_d$ has the same value for all $n \geqq d$, and in order to give some indication of the maximum array space needed for a given d, $N^{(n)}_d$ is listed in Table 4.1 for $d = n = 2, 3, \cdots, 10$. We also list for comparison purposes $\binom{n+d}{d}$,

TABLE 4.1
Space required for new and old algorithms

d	2	3	4	5	6	7	8	9	10
$N^{(d)}_d$	4	7	12	19	30	45	67	97	139
$\binom{2d}{d}$	6	20	70	252	924	3432	12870	48620	184756

the space required for the original recursive extrapolation algorithm. The space requirements are clearly less for the symmetrized algorithms, and more so as n and d increase.

We now consider the time requirements for the two new algorithms, distinct from the additional time necessary to compute elements in the sequence $\{A_{\mathbf{m}}\}$. It is difficult to give a precise analysis of this because of the variety of operations involved in the computation of P_d using either algorithm. If we assume that the counting is done efficiently then the time for both algorithms will be proportional to the time required for the arithmetic, which usually can be done with multiplications and additions. There is usually one addition with every multiplication, so we provide a rough analysis which is based on counting the number of multiplications.

For the recursive algorithm the multiplications necessary to compute the coefficients $\rho_{\mathbf{m},\mathbf{k}}$ will be the most important. In order to reduce the total number of multiplications the weights $w_{m,k}$, defined by (3.4) in the previous section, should be precomputed and stored at the beginning of the computation. If we assume this is done then (3.5) requires roughly $\prod_{i=1}^{n}(m_i+1)$ multiplications if the sum is done in nested form and P_d requires a number of multiplications, which we call $R^{(n)}_d$, given by

$$R^{(n)}_d = \sum_{j=1}^{N^{(n)}_d} \prod_{i=1}^{n} (m_{ij}+1),$$

where $\mathbf{m}_j = (m_{1j}, m_{2j}, \cdots, m_{nj})$. We note that $R^{(n)}_d = R^{(d)}_d$ for all $n > d$.

The Lagrange-based algorithm is dominated by the computation of the numbers $\Phi_{\mathbf{m}_j}^{(d)}$. From (2.2)

$$\Phi_{\mathbf{m}}^{(d)} = \sum_{k_1=0}^{d-|\mathbf{m}|} w'_{m_1,k_1} \sum_{k_2=0}^{d-|\mathbf{m}|-k_1} w'_{m_2,k_2} \cdots \sum_{k_{n-1}=0}^{d-|\mathbf{m}|-k_1-\cdots-k_{n-2}} w'_{m_{n-1},k_{n-1}} w'_{m_n,k_n},$$

with

$$w'_{m,k} = \prod_{l=0}^{k+m-1} x_l \Big/ \prod_{l=0,\neq m}^{k+m} (x_l - x_m).$$

If the weights $w'_{m,k}$ have been precomputed and stored, and nested multiplication is used then the computation of $\Phi_{\mathbf{m}}^{(d)}$ requires roughly $\binom{n-1+d-|\mathbf{m}|}{d-|\mathbf{m}|}$ multiplications and P_d requires a number of multiplications, which we call $L_d^{(n)}$, given by

$$L_d^{(n)} = \sum_{j=1}^{N_d^{(n)}} \binom{n-1+d-|\mathbf{m}_j|}{d-|\mathbf{m}_j|}.$$

In Table 4.2 we list $R_d^{(n)}$ and $L_d^{(n)}$ for $n+d \leqq 12$, $1 \leqq d \leqq 9$ and $2 \leqq n \leqq 10$, with $L_d^{(n)}$ given above $R_d^{(n)}$ for each pair (n, d). For large values of n and d the total time required for either extrapolation algorithm will be roughly proportional to the associated multiplication number $R_d^{(n)}$ or $L_d^{(n)}$. We see from the table that $R_d^{(n)} < L_d^{(n)}$ except for small values of n. For d sufficiently large it can be shown that eventually $L_d^{(n)} < R_d^{(n)}$. Anyway, the numbers in the table give a clear indication of the efficiency of either algorithm for a range of values of n and d most likely to be of practical interest, when compared with the multiplication number $\binom{2n+d}{d}$ (see Table 4.4) required for the original recursive extrapolation algorithm.

TABLE 4.2
Required numbers of multiplications $L_d^{(n)}$ and $R_d^{(n)}$

$d\backslash n$	2	3	4	5	6	7	8	9	10
1	3	4	5	6	7	8	9	10	11
	3	3	3	3	3	3	3	3	3
2	7	11	16	22	29	37	46	56	67
	10	10	10	10	10	10	10	10	10
3	13	25	41	63	92	129	175	231	298
	20	28	28	28	28	28	28	28	28
4	22	50	92	155	247	376	551	782	1080
	42	62	78	78	78	78	78	78	78
5	34	91	187	343	590	966	1517	2299	3379
	70	124	164	196	196	196	196	196	196
6	50	155	353	701	1292	2258	3775	6074	
	120	245	353	433	497	497	497	497	
7	70	250	628	1345	2643	4902	8677		
	180	427	677	893	1053	1181	1181		
8	95	386	1065	2451	5116	10025			
	275	719	1294	1794	2226	2546			
9	125	575	1735	4278	9457				
	385	1179	2256	3406	4406				

The time taken by either algorithm when applied to the problem of extrapolating sequences of product midpoint rule approximations to multiple integrals will certainly be dominated by the integrand function evaluations.

The stability of all three algorithms when used for multiple integration is expected to be poor, because it is known to be poor for $n = 1$, when they reduce to Romberg integration with mesh sequence $h_m = 1/(m+1)$. A simple test was carried out to determine the seriousness of this expected instability. We used a series of ten trials for each of a range of values of n and d, with the Lagrange extrapolation algorithm applied to the test integrand $f(\mathbf{x}) = 1 + 10^{-10}r$, where r was a uniformly distributed (pseudo) random number chosen from $[-1, 1]$, and a new r provided for each integrand evaluation. In theory $I(f) = 1$, and in practice we would expect a stable algorithm to give a result accurate to at least ten decimal digits. In Table 4.3 we give the average

TABLE 4.3
Digits lost because of instability

$d\backslash n$	2	3	4	5	6	7	8	9	10
1	−.1	.3	.2	.4	.0	.6	.5	.8	.7
2	.0	.6	.7	.8	.7	1.1	1.1	1.6	1.5
3	.4	.7	1.2	1.1	1.4	1.2	1.7	2.2	2.0
4	.2	1.2	1.4	1.6	1.7	2.0	2.0	2.6	2.7
5	.7	1.5	1.7	1.9	1.9	2.6	2.4	3.0	3.2
6	1.1	1.6	1.8	2.3	2.5	2.8	2.8	3.5	
7	1.4	1.7	2.0	2.6	2.9	2.9	3.2		
8	1.7	1.6	2.4	2.9	3.1	3.2			
9	2.0	2.0	2.6	3.0	3.5				

number of decimal digits lost from the expected number, ten. Similar results were obtained using the recursive algorithm in its original and symmetrized forms. The averages listed all had associated standard deviations in the range .3–.9 and the computation was carried out on a computer with approximately fourteen decimal digits accuracy.

The results in Table 4.3 indicate that for a range of values of n and d of practical interest, up to about four decimal digits could be lost using any of the extrapolation algorithms discussed here with integrands which are reasonably smooth. As most scientific calculations are carried out with ten to sixteen decimal digits precision, this instability should usually not be a problem.

Finally, in order to place the integration rules described here in a somewhat wider context, we compare the number of integrand evaluations required for the integration rules generated by the sequence $\{P_d\}$, with the number required by product Gauss–Legendre rules of degree $2d+1$ and the more recently described fully symmetric rules of degree $2d+1$ described by Keast [2]. In Table 4.4 we give three numbers for each value of n and d. The first is the number of integrand evaluations required for the Keast rules, the second is the number required for P_d and the third is the number $(d+1)^n$ required for a product Gauss rule. For specific values of n and d, all three numbers are for multiple integration rules of degree $2d+1$.

Clearly the Keast rules are more efficient than the rules generated by the extrapolation methods discussed in this paper, when efficiency is measured in terms of polynomial

TABLE 4.4
Numbers of integrand evaluations required for the three rule types

$d\backslash n$	2	3	4	5	6	7	8	9	10
1	5	7	9	11	13	15	17	19	21
	5	7	9	11	13	15	17	19	21
	4	8	16	32	64	128	256	512	1024
2	9	19	33	51	73	99	129	163	201
	15	28	45	66	91	120	153	190	231
	9	27	81	243	729	2187	6561	19683	59049
3	17	45	97	181	305	477	705	997	1361
	35	84	165	286	455	680	969	1330	1771
	16	64	256	1024	4096	16384	65536	262144	1048576
4	25	77	193	421	825	1485	2497	3973	6041
	70	210	495	1001	1820	3060	4845	7315	10626
	25	125	625	3125	15625	78125	390625	1953125	9565625
5	41	151	417	983	2089	4103	7553	13159	21865
	126	462	1285	3003	6188	11628	20349	33649	53130
	36	216	1296	7776	46656	279936	1659616	10077696	60466176
6	49	223	737	1975	4625	9871	19649	36967	
	210	924	3003	8008	18564	38760	74613	134596	
	49	343	2401	16807	117649	823543	5764801	40353607	
7	73	369	1329	3897	19913	22753	48353		
	330	1716	6453	19448	50388	116280	245157		
	64	512	4096	32768	262144	2097152	16777216		
8	81	465	1953	6489	18353	46177			
	495	3003	12870	43758	125970	319770			
	81	729	6561	59049	531441	4782969			
9	113	731	3201	11211	33649				
	715	5005	24310	92378	293930				
	100	1000	10000	100000	1000000				

integrating power per function evaluation. However, the extrapolation generated rules are more efficient for many values of n and d than the product Gauss rules. They also form nested families of easily generated rules which could be useful for automatic multiple numerical integration routines and comparisons with results from other methods.

The Fortran subroutine INTLAG listed at the end of this paper computes the sequence $\{P_d\}$ for $d = 0, 1, \cdots$ MAXORD-1 using the algorithm given at the end of § 2. The routine is written in ANSI (66) standard Fortran with all arguments defined in the comments at the beginning of the subroutine. A subroutine using Lagrange extrapolation is somewhat less efficient for many values of n and d, than one based on formula (3.5) but, as was discussed at the end of § 3, (3.5) is difficult to implement efficiently. The Lagrange derived method also works directly with the symmetric sums $S_{\mathbf{m}_j}$, and minor modifications could easily be made to the subroutine if a user wanted to obtain and save the weights $\Phi^{(d)}_{\mathbf{m}_j}$. Except for one possible change of REAL to DOUBLE PRECISION advisable for computers with fewer than about eight decimal digits single precision, the subroutine should run without modification on any computer with a standard Fortran compiler.

We conclude this section with a short example program illustrating the use of INTLAG when applied to the integral

$$I(f)=\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}\frac{1}{4+x_1+x_2+x_3}\,dx_1\,dx_2\,dx_3=2.15214283.$$

EXAMPLE PROGRAM WITH RESULTS

```
      EXTERNAL FUNINT
      INTEGER IFAIL,J,MAXORD,MINORD,NOUT
      REAL ACTERR,ESTERR
      REAL FINVLS(8),A(3),B(3),SYMSMS(200)
      DATA NOUT/6/
      WRITE(NOUT,99999)
      DO 20 J=1,3
         A(J)=-1
         B(J)=1
   20    CONTINUE
      MINORD=0
      DO 40 MAXORD=2,8
         CALL INTLAG(3,A,B,FUNINT,MINORD,MAXORD,FINVLS,200,SYMSMS,IFAIL)
         ACTERR=ABS(FINVLS(MAXORD)-2.1521428326D0)
         ESTERR=ABS(FINVLS(MAXORD)-FINVLS(MAXORD-1))
         WRITE(NOUT,99998) MAXORD,FINVLS(MAXORD),ESTERR,ACTERR
   40    CONTINUE
99999 FORMAT(4(1X/),40H          INTLAG EXAMPLE PROGRAM RESULTS//3X,
     * 60HMAXORD  ESTIMATED VALUE  ESTIMATED ACCURACY  ACTUAL ACCURACY)
99998 FORMAT(3X,I4,3X,F14.8,1X,F17.8,2X,F17.8)
      STOP
      END


      REAL FUNCTION FUNINT(NUMVAR,Z)
      INTEGER NUMVAR
      REAL Z(NUMVAR)
      FUNINT=1.0/(4.0+Z(1)+Z(2)+Z(3))
      RETURN
      END
```

```
          INTLAG EXAMPLE PROGRAM RESULTS

MAXORD  ESTIMATED VALUE  ESTIMATED ACCURACY  ACTUAL ACCURACY
   2       2.12698413         .12698413          .02515871
   3       2.14677249         .01978836          .00537035
   4       2.15089424         .00412176          .00124859
   5       2.15183772         .00094347          .00030512
   6       2.15206551         .00022780          .00007732
   7       2.15212269         .00005718          .00002014
   8       2.15213748         .00001478          .00000536
```

FORTRAN SUBROUTINE

```
      SUBROUTINE INTLAG(NUMVAR, LOWER, UPPER, FUNCTN, MINORD, MAXORD,
     * FINVLS, NUMSMS, SYMSMS, IFAIL)
C  MULTIDIMENSIONAL GENERALISED ROMBERG INTEGRATION SUBROUTINE
C
C   THIS SUBROUTINE COMPUTES A SEQUENCE OF APPROXIMATIONS TO A MULTIPLE
C   INTEGRAL USING AN EXTRAPOLATION METHOD BASED ON MULTIVARIABLE
C   LAGRANGE INTERPOLATION, AS DESCRIBED IN THE PAPER ;  A LAGRANGE
C   EXTRAPOLATION ALGORITHM FOR SEQUENCES OF APPROXIMATIONS TO MULTIPLE
C   INTEGRALS,  BY A. GENZ, MATHEMATICAL INSTITUTE, UNIVERSITY OF KENT,
C   CANTERBURY, KENT CT2 7NF, ENGLAND
C
C**************  PARAMETERS FOR INTLAG  ********************************
C  NUMVAR   INTEGER NUMBER OF VARIABLES, MUST EXCEED 1 BUT NOT EXCEED 16
C  LOWER    REAL LOWER INTEGRATION LIMITS ARRAY WITH DIMENSION(NUMVAR)
C  UPPER    REAL UPPER INTEGRATION LIMITS ARRAY WITH DIMENSION(NUMVAR)
C  FUNCTN   EXTERNALLY DECLARED USER DEFINED REAL FUNCTION INTEGRAND.
C           IT MUST HAVE PARAMETERS (NUMVAR,X), WHERE X IS A REAL ARRAY
C           WITH DIMENSION NUMVAR.
C  MINORD   INTEGER MINIMUM ORDER PARAMETER.  ON ENTRY MINORD SPECIFIES
C           THE CURRENT HIGHEST ORDER APPROXIMATION TO THE INTEGRAL,
C           AVAILABLE IN THE ARRAY FINVLS.  FOR THE FIRST CALL OF INTLAG
C           MINORD SHOULD BE SET TO 0.  OTHERWISE A PREVIOUS CALL IS
C           ASSUMED WHICH COMPUTED FINVLS(1), ... , FINVLS(MINORD).
C           ON EXIT MINORD IS SET TO MAXORD.
C  MAXORD   INTEGER MAXIMUM ORDER PARAMETER, MUST BE GREATER THAN MINORD
C           AND NOT EXCEED 16. THE SUBROUTINE COMPUTES FINVLS(MINORD+1),
C           FINVLS(MINORD+2),..., FINVLS(MAXORD).
C  FINVLS   REAL ARRAY OF DIMENSION(MAXORD).  UPON SUCCESSFUL EXIT
C           FINVLS(1), FINVLS(2),..., FINVLS(MAXORD) ARE APPROXIMATIONS
C           TO THE INTEGRAL.  FINVLS(J) WILL BE AN APPROXIMATION OF
C           POLYNOMIAL DEGREE 2J-1.
C  NUMSMS   INTEGER LENGTH OF ARRAY SYMSMS, MUST BE AT LEAST THE SUM OF
C           THE NUMBER OF DISTINCT PARTITIONS OF LENGTH AT MOST NUMVAR
C           OF THE INTEGERS 0,1,...,MAXORD-1.  AN UPPER BOUND FOR NUMSMS
C           WHEN NUMVAR+MAXORD IS LESS THAN 19 IS 200
C  SYMSMS   REAL WORKING STORAGE ARRAY WITH DIMENSION (NUMSMS). ON EXIT
C           SYMSMS(J) CONTAINS THE SUM OF ALL PRODUCT MIDPOINT INTEGRAL
C           APPROXIMATIONS ASSOCIATED WITH THE JTH DISTINCT PARTITION OF
C           THE INTEGERS 0,1,...,MAXORD-1.
C  IFAIL     INTEGER FAILURE OUTPUT PARAMETER
C           IFAIL=0 FOR SUCCESSFUL TERMINATION OF THE SUBROUTINE
C           IFAIL=1 WHEN NUMSMS IS TOO SMALL FOR THE SUBROUTINE TO
C                   CONTINUE.  IN THIS CASE FINVLS(1), FINVLS(2),...,
C                   FINVLS(J) ARE RETURNED, WHERE J IS MAXIMUM VALUE OF
C                   MAXORD COMPATIBLE WITH THE GIVEN VALUE OF NUMSMS.
C           IFAIL=2 WHEN PARAMETERS NUMVAR,MINORD OR MAXORD ARE OUT OF
C                   RANGE
C***********************************************************************
C***  FOR DOUBLE PRECISION CHANGE REAL TO DOUBLE PRECISION
C      IN THE NEXT STATEMENT
      REAL FINVLS(MAXORD), FLOATL, FUNCTN, H(16), HISQRD, HLSQRD,
     * HX(16), INTVAL, INTWT, LOWER(NUMVAR), ONE, PHISUM(16), PROINT,
     * SYMSMM, SYMSMS(NUMSMS), TWO, UPPER(NUMVAR), WT, WTPROD(16,16),
     * X(16), ZERO
      INTEGER D, I, IFAIL, IHALF, IMNUSL, IXCHNG, J, K(16), K1, KI,
     * L, LXCHNG, M(16), M1, MAXORD, MI, MINORD, MODOFM, MP(16), MPI,
C    * MPL, MSUM, NUMSMS, NUMVAR
C***  PARAMETER CHECKING AND INITIALISATION
      IFAIL = 2
      IF (NUMVAR.GT.16 .OR. NUMVAR.LT.2) RETURN
      IF (MINORD.LT.0 .OR. MINORD.GE.MAXORD) RETURN
      IF (MAXORD.GT.16) RETURN
      IFAIL = 1
      ZERO = 0
      ONE = 1
      TWO = 2
      D = MINORD
C***  CALCULATE MESH SEQUENCE AND PRECOMPUTED WEIGHTS
      DO 10 L=1,MAXORD
        FLOATL = L
        H(L) = ONE/FLOATL
   10 CONTINUE
      DO 30 L=1,MAXORD
        HLSQRD = H(L)**2
        WT = ONE
        DO 20 I=1,MAXORD
          HISQRD = H(I)**2
          IF (I.LT.L) WT = WT*HISQRD/(HISQRD-HLSQRD)
          IF (I.GT.L) WT = WT*H(I-1)**2/(HISQRD-HLSQRD)
          WTPROD(L,I) = WT
   20   CONTINUE
   30 CONTINUE
```

```
C
C***   BEGIN LOOP FOR EACH D
C       FOR EACH D FIND ALL DISTINCT PARTITIONS M WITH MOD(M)<=D
C
   40 J = 1
      INTVAL = ZERO
      MODOFM = 0
      DO 50 I=1,NUMVAR
        K(I) = 0
        PHISUM(I) = ZERO
        M(I) = 0
   50 CONTINUE
   60 IF (J.GT.NUMSMS) RETURN
      IF (MODOFM.LT.D) GO TO 150
C
C*****  WHEN MOD(M)=D FIND ALL PERMUTATIONS MP OF M
C         AND COMPUTE INTEGRAL FOR EACH MP
      DO 70 I=1,NUMVAR
        MP(I) = M(I)
   70 CONTINUE
      SYMSMM = ZERO
C
C*******  COMPUTE PRODUCT INTEGRAL FOR PERMUTATION MP
   80 PRDINT = ZERO
      INTWT = ONE
      DO 90 I=1,NUMVAR
        MPI = MP(I) + 1
        HX(I) = H(MPI)*(UPPER(I)-LOWER(I))
        INTWT = INTWT*HX(I)
        X(I) = LOWER(I) + HX(I)/TWO
   90 CONTINUE
  100 PRDINT = PRDINT + FUNCTN(NUMVAR,X)
      DO 110 I=1,NUMVAR
        X(I) = X(I) + HX(I)
        IF (X(I).LT.UPPER(I)) GO TO 100
        X(I) = LOWER(I) + HX(I)/TWO
  110 CONTINUE
C*******  END INTEGRATION LOOP FOR MP
C
      SYMSMM = SYMSMM + INTWT*PRDINT
C
C*******  FIND NEXT DISTINCT PERMUTATION OF M
C           AND LOOP BACK TO COMPUTE NEXT INTEGRAL
      DO 140 I=2,NUMVAR
        IF (MP(I-1).LE.MP(I)) GO TO 140
        MPI = MP(I)
        IXCHNG = I - 1
        IF (I.EQ.2) GO TO 130
        IHALF = IXCHNG/2
        DO 120 L=1,IHALF
          MPL = MP(L)
          IMNUSL = I - L
          MP(L) = MP(IMNUSL)
          MP(IMNUSL) = MPL
          IF (MPL.LE.MPI) IXCHNG = IXCHNG - 1
          IF (MP(L).GT.MPI) LXCHNG = L
  120   CONTINUE
        IF (MP(IXCHNG).LE.MPI) IXCHNG = LXCHNG
  130   MP(I) = MP(IXCHNG)
        MP(IXCHNG) = MPI
        GO TO 80
  140 CONTINUE
C*****  END LOOP FOR PERMUTATIONS OF M AND ASSOCIATED INTEGRALS
C
      SYMSMS(J) = SYMSMM
C
C*****  CALCULATE WEIGHT FOR PARTITION M
  150 M1 = M(1) + 1
      K1 = D - MODOFM + M1
  160 PHISUM(1) = WTPROD(M1,K1)
      DO 170 I=2,NUMVAR
        MI = M(I) + 1
        KI = K(I) + MI
        PHISUM(I) = PHISUM(I) + WTPROD(MI,KI)*PHISUM(I-1)
        PHISUM(I-1) = ZERO
        K1 = K1 - 1
        K(I) = K(I) + 1
        IF (K1.GE.M1) GO TO 160
        K1 = K1 + K(I)
        K(I) = 0
  170 CONTINUE
      INTVAL = INTVAL + PHISUM(NUMVAR)*SYMSMS(J)
      PHISUM(NUMVAR) = ZERO
C
C***  FIND NEXT PARTITION M AND LOOP BACK TO COMPUTE
C       ASSOCIATED INTEGRALS AND/OR WEIGHT
      J = J + 1
      MSUM = M(1)
      DO 200 I=2,NUMVAR
        MSUM = MSUM + M(I)
        IF (M(1).LE.M(I)+1) GO TO 190
        M(1) = MSUM - (I-1)*(M(I)+1)
        DO 180 L=2,I
          M(L) = M(I) + 1
 180    CONTINUE
        GO TO 60
 190    M(I) = 0
 200  CONTINUE
      M(1) = MSUM + 1
      MODOFM = MODOFM + 1
      IF (MODOFM.LE.D) GO TO 60
C
C***  END LOOP FOR EACH D
      IF (D.GT.0) INTVAL = FINVLS(D) + INTVAL
      FINVLS(D+1) = INTVAL
      D = D + 1
      IF (D.LT.MAXORD) GO TO 40
C
C***  SET FAILURE PARAMETER AND RETURN
      IFAIL = 0
      MINORD = MAXORD
      RETURN
      END
```

REFERENCES

[1] A. C. GENZ, *Some extrapolation methods for the numerical calculation of multiple integrals*, Software for Numerical Mathematics, D. J. Evans, ed., Academic Press, New York, 1978, pp. 162–172.

[2] E. ISAACSON AND H. B. KELLER, *Analysis of Numerical Methods*, John Wiley, New York, 1966.

[3] P. KEAST, *Some fully symmetric quadrature formulae for product spaces*, J. Inst. Maths. Applics., 23 (1979), pp. 251–264.

[4] E. H. MCKINNEY, *Generalized recursive multivariate interpolation*, Math. Comp., 26 (1972), pp. 723–735.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

ARC-LENGTH CONTINUATION AND MULTI-GRID TECHNIQUES FOR NONLINEAR ELLIPTIC EIGENVALUE PROBLEMS*

TONY F. C. CHAN† AND H. B. KELLER‡

Abstract. We investigate multi-grid methods for solving linear systems arising from arc-length continuation techniques applied to nonlinear elliptic eigenvalue problems. We find that the usual multi-grid methods diverge in the neighborhood of singular points of the solution branches. As a result, the continuation method is unable to continue past a limit point in the Bratu problem. This divergence is analyzed and a modified multi-grid algorithm has been devised based on this analysis. In principle, this new multi-grid algorithm converges for elliptic systems arbitrarily close to singularity and has been used successfully in conjunction with arc-length continuation procedures on the model problem. In the worst situation, both the storage and the computational work are only about a factor of two more than the unmodified multi-grid methods.

Key words. multi-grid, arc-length continuation, nonlinear elliptic eigenvalue problems, singular systems

1. Introduction.

Many problems of computational interest can be formulated as

$$G(u, \lambda) = 0, \tag{1.1}$$

where u represents the "solution" (i.e., flow field, displacements, etc.) and λ is a real physical parameter (i.e., Reynold's number, load, etc.) It is required to find the solution for some λ-intervals, that is, a path of solutions, $[u(\lambda), \lambda]$. In this paper, we use a class of continuation based on parametrizing the solution branches by arc-length, say $[u(s), \lambda(s)]$. A main advantage of these arc-length continuation methods is that most singular points on the solution branches can be handled without much difficulty. Equations of the form (1.1) are called nonlinear *elliptic* eigenvalue problems if the operator G with λ fixed is an elliptic differential operator [2]. For nonlinear *elliptic* eigenvalue problems, a major portion of the computational work in the arc-length continuation methods is spent in solving large *linear* elliptic systems. In this paper, we investigate the use of multi-grid [4] methods for solving these linear systems. It turns out that a straightforward implementation of the multi-grid methods fails in the neighborhood of the singular points and this usually prevents continuation past limit points. This failure is analyzed and a modified multi-grid method based on this analysis is devised. Even for very singular systems, the new multi-grid algorithm performs satisfactorily and never requires more than about twice the storage and computational work as the unmodified algorithm.

The arc-length continuation methods will be described in § 2 and the multi-grid methods in § 3. In § 4, computational results for a model problem are presented, together with a description of the difficulties encountered by the multi-grid method near a limit point. The behavior of the multi-grid method near singular points will be analyzed in § 5. The modified multi-grid algorithms designed to overcome these difficulties are described in § 6. The paper ends with a summary in § 7.

* Received by the editors April 1, 1981.

† Computer Science Department, Yale University, New Haven, Connecticut 06520. This author's work was supported by the U.S. Department of Energy under contract EY-76-S-03-070 while he was at California Institute of Technology and by the Office of Naval Research under grant N00014-80-0076 under subcontract from Florida State University while he was at Yale.

‡ Applied Mathematics Department, California Institute of Technology, Pasadena, California 91125. This author's work was supported by the U.S. Department of Energy under contract EY-76-S-03-070 and by the U.S. Army Research Office under contract DAAG 29-78-C-0011.

2. Newton's method and continuation techniques. In this paper we are concerned with methods for computing a family or path of solutions of (1.1). The methods we employ will be based on some version of Newton's method.

2.1. Newton's method. Given a value of λ and an initial guess u^0 for the solution $u(\lambda)$, we perform the following steps repeatedly until $\|\delta u^i\| < \varepsilon$ is satisfied:

$$G_u^i \delta u^i = -G(u^i, \lambda), \tag{2.1}$$

$$u^{i+1} = u^i + \delta u^i. \tag{2.2}$$

In the above, subscripts denote partial derivatives and so G_u denotes the Jacobian of the operator G (with respect to u). This procedure will generally converge quadratically when it does converge. However, as is well known, in many instances it will fail to converge when the initial guess is not "close" to the true solution.

2.2. Natural continuation. A plausible procedure for overcoming this convergence difficulty and also for determining the dependence of u on λ is to start at a known solution (u_0, λ_0) on the solution curve and use it as initial guess for a Newton-type iteration to find the solution for a neighboring point on the solution curve with λ close to λ_0. The procedure is then repeated. We can improve on this by computing the derivative, u_λ, at a known solution and use it to get a better initial guess for the next value of λ in a predictor-corrector fashion. We call this a natural continuation procedure because it corresponds to parametrizing the solution curve by λ, the naturally occurring parameter. A specific form of this is the more or less well-known

Euler–Newton continuation procedure. Given a known solution (u_0, λ_0), *we compute the solutions at nearby values of* λ *as follows*:

1. *First compute the derivative* u_λ *at* (u_0, λ_0) *from*

$$G_u u_\lambda = -G_\lambda. \tag{2.3}$$

2. *Perform an Euler predictor step*:

$$u^0 = u_0 + u_\lambda (\lambda - \lambda_0). \tag{2.4}$$

3. *Use* u^0 *as initial guess in Newton's method,*

$$G_u^i(u^{i+1} - u^i) = -G(u^i, \lambda), \tag{2.5}$$

until convergence.

4. *Use* $(u(\lambda), \lambda)$ *as the new* (u_0, λ_0) *and go back to Step* 1.

Note that the computation of the derivative u_λ does not cause much computational overhead because we usually have the factorization of the Jacobian G_u computed already in the Newton step. Using such a predictor-corrector method will often allow us to take a much bigger step in λ and thus reduce the overall cost of determining the dependence of u on λ.

Unfortunately, this procedure needs some modification in order to handle general nonlinear systems because of the possibility of existence of nonunique solutions. The nonuniqueness usually manifests itself in the form of existence of "singular" points where the Jacobian G_u is singular (see Fig. 2.1). Points such as point A in Fig. 2.1 are called limit points (or turning points) and points such as point B are called bifurcation points. These singular points are further characterized by the conditions that $G_\lambda \notin \text{Range}\,(G_u)$ at a limit point and that $G_\lambda \in \text{Range}\,(G_u)$ at a bifurcation point [12].

The difficulties that a natural continuation procedure will encounter at singular points are threefold. First of all, since G_u is singular at these points, Newton's method

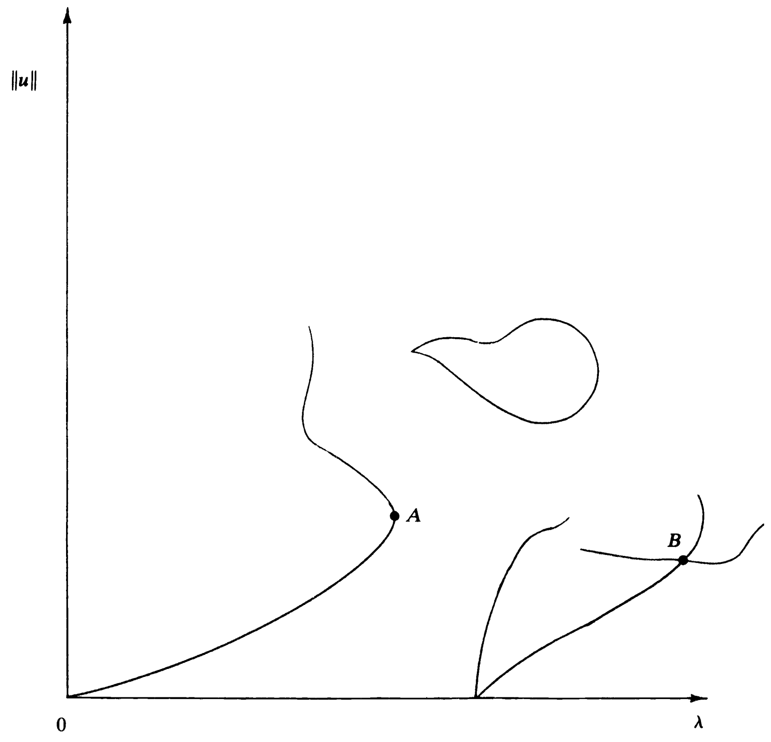


FIG. 2.1. *A typical bifurcation diagram.*

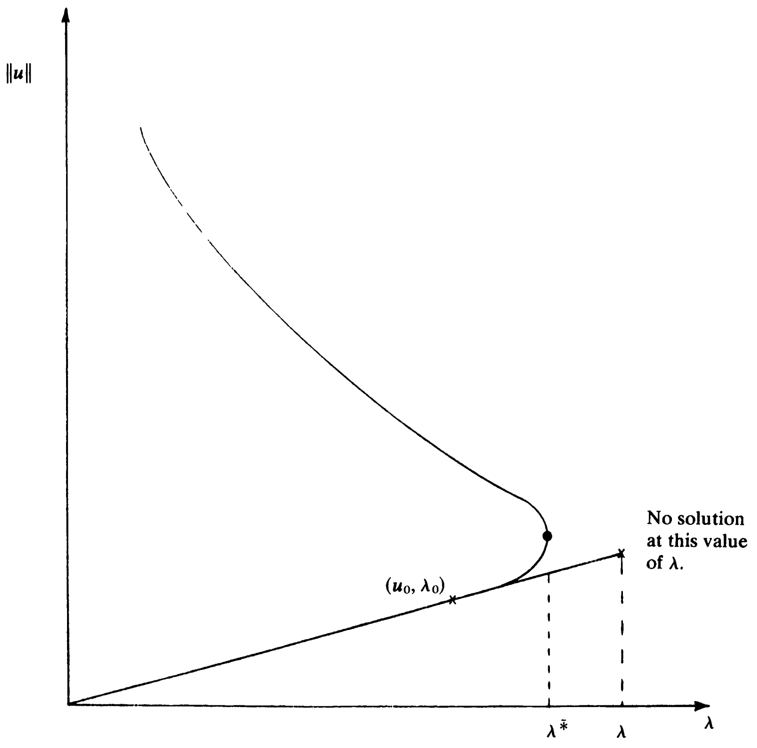


FIG. 2.2. *Failure of natural continuation near limit points.*

will at best be linearly convergent, making it much more costly to compute the solution. Moreover, near a limit point, there may not exist a solution for a given value of λ (see Fig. 2.2) and hence the iterations must fail to converge. Lastly, we need some mechanism for switching branches at a bifurcation point.

2.3. Arc-length continuation. In the pseudo arc-length continuation approach [12], these difficulties are overcome by not parametrizing the solution u by λ. Instead, we parametrize the solution branches using an arc-length parameter s, and specify how far along the current solution branch we want to march.

To be more specific, we let s be the arc-length parameter, and treat $u(s)$ and $\lambda(s)$ as functions of s. We can compute the "tangent" $[\dot{u}(s_0), \dot{\lambda}(s_0)]$ (where the dots denote differentiation with respect to s) of a known solution at $s = s_0$ from the following two equations:

$$G_u\dot{u}_0 + \dot{\lambda}_0 G_\lambda = 0, \tag{2.6}$$

$$\|\dot{u}_0\|^2 + |\dot{\lambda}_0|^2 - 1 = 0. \tag{2.7}$$

Equation (2.6) is obtained from differentiating $G(u, \lambda) = 0$ with respect to s and (2.7) imposes the arc-length condition. We could theoretically generate the solution curve by integrating the initial value problem obtained by solving (2.6), (2.7) for $\dot{u}(s)$ and $\dot{\lambda}(s)$. Although this process is subject to the usual instabilities inherent in solving initial value problems approximately, it can be an extremely effective procedure. Indeed our pseudo arc-length continuation procedure can be viewed as a method for stabilizing Euler integration of (2.6), (2.7).

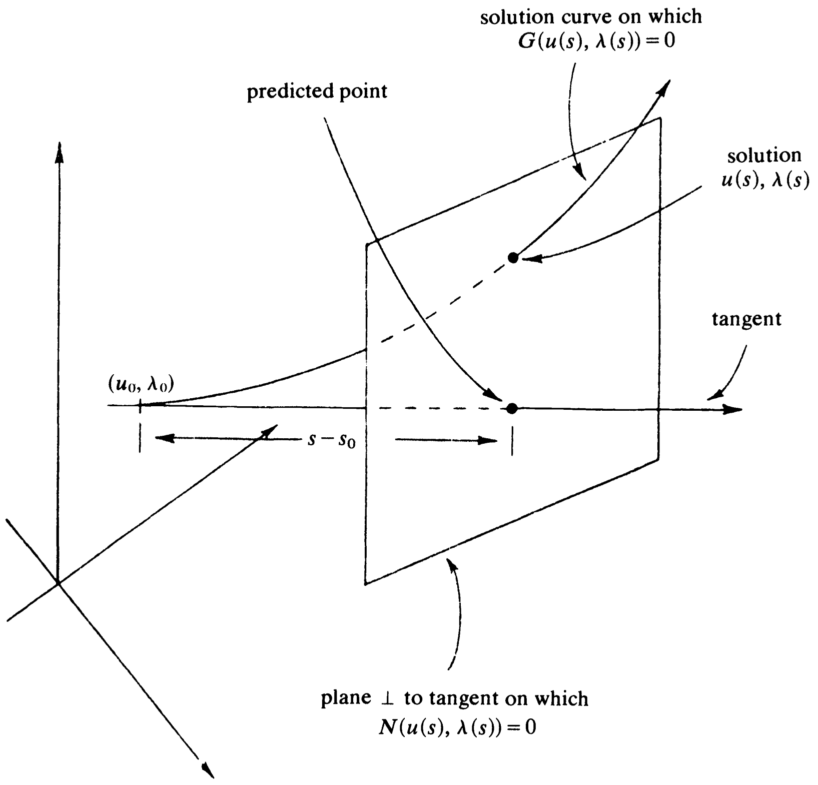


FIG. 2.3. *Pseudo arc-length continuation.*

In the pseudo arc-length continuation procedure, we advance from s_0 to s along the tangent to the solution branch and require the new solution $u(s)$ and $\lambda(s)$ to satisfy

$$N(u(s),\lambda(s)) \equiv \dot{u}_0^T(u(s)-u(s_0))+\dot{\lambda}_0(\lambda(s)-\lambda(s_0))-(s-s_0)=0. \tag{2.8}$$

In addition we require, of course,

$$G(u(s),\lambda(s))=0. \tag{2.9}$$

Equation (2.8) is the linearization of (2.7), and as indicated forces the new solution to lie on a hyperplane perpendicular to the tangent vector to the solution curve at s_0 and at a distance $(s-s_0)$ from it. Equation (2.9) requires $u(s)$ and $\lambda(s)$ to lie on the true solution curve (Fig. 2.3). We now solve the coupled system (2.8) and (2.9) for $u(s)$ and $\lambda(s)$, given the step size $(s-s_0)$ (efficient strategies for choosing the step size are discussed in [23]). We use Newton's method, in which case we have to solve the following linear system at each iteration:

$$A\begin{bmatrix}\delta u\\ \delta\lambda\end{bmatrix}=\begin{bmatrix}G_u & G_\lambda\\ N_u^T & N_\lambda\end{bmatrix}\begin{bmatrix}\delta u\\ \delta\lambda\end{bmatrix}=-\begin{bmatrix}G\\ N\end{bmatrix}. \tag{2.10}$$

It can be shown that at limit points, where G_u is singular and $G_\lambda \notin \text{Range}(G_u)$, the linear system in (2.10) is nonsingular (see [12]) and therefore Newton's method for the coupled system (2.8) and (2.9) is well defined. Hence limit points present no problem and even quadratic convergence is achievable.

At bifurcation points, where G_u is singular and $G_\lambda \in \text{Range}(G_u)$, things are more complicated. In the simplest case of only one branch bifurcating from the main branch (simple bifurcation), an additional higher order condition involving G_{uu}, $G_{u\lambda}$ and $G_{\lambda\lambda}$ has to be satisfied. It can be shown [12] that this condition, together with (2.6) and (2.7) and the left and right null vectors of G_u, enable two solutions for $(\dot{u}_0, \dot{\lambda}_0)$ to be computed at a simple bifurcation point, with one solution corresponding to each branch. Using the appropriate pair of $(\dot{u}_0, \dot{\lambda}_0)$ in (2.8) allows branch switching. In [7] a more detailed study of the singular behavior and branch switching at bifurcation is given.

In order to solve the linear system in (2.10) by direct methods, several approaches are possible. One way is to perform Gaussian elimination on the inflated matrix A, with some form of pivoting to ensure stability. But this approach completely ignores the sparse structure which is usually found in G_u arising from nonlinear elliptic eigenvalue problems. In order to take advantage of the structure in G_u, Keller [12] suggested the following block-elimination procedure:

Algorithm BE (block-elimination)

Solve

$$G_u y = G_\lambda \tag{2.11}$$

and

$$G_u z = -G. \tag{2.12}$$

Set

$$\delta\lambda = (-N_u^T z - N)/(N_\lambda - N_u^T y) \tag{2.13}$$

and

$$\delta u = z - \delta\lambda y. \tag{2.14}$$

Note that only systems with the coefficient matrix G_u have to be solved, so structures in G_u can be exploited. Moreover, only one factorization of G_u is needed. It has been shown [27] that even when G_u is becoming singular, Algorithm BE produces iterates that converge quadratically at limit points.

Continuation methods of various forms and levels of sophistication have been widely used in the engineering literature. For a recent survey of numerical methods for bifurcation problems, see for example [18]. The approach taken here is due to Keller [12], and has recently been applied to other problems in fluid mechanics [5], [6], [15], [16], [25], [27]. A related approach suggested by Abbott [1] corresponds (in a loose way) to applying Algorithm BE to the matrix A with the last column permuted into the first n columns so that the corresponding coefficient matrix in equations (2.11) and (2.12) becomes nonsingular even at limit points. However, as has already been pointed out, any structure or symmetry in G_u is lost in the process, and hence that approach seems unsuitable for large elliptic systems in two or three dimensions.

3. Multi-grid methods.

3.1. Introduction. The class of multi-grid (MG) methods that we use here is based on work by Bakhvalov [3], Brandt [4], Federenko [8], Hackbush [10] and Nicolaides [19]. We shall only briefly describe here the particular MG algorithms that we have used for linear elliptic problems that arise in our treatment of nonlinear elliptic eigenvalue problems.

The particular way in which we use the MG idea is to use a hierarchy of grids, rather than a single one, in order to speed up the convergence rate of the solution process. The MG process has some very desirable theoretical properties: for certain elliptic operators on an $n \times n$ grid, it computes the approximate solution to truncation error accuracy in $O(n^2)$ arithmetic operations and $O(n^2)$ storage. It seems natural to consider the use of MG methods for solving nonlinear eigenvalue problems. MG methods have been applied to solution of linear eigenvalue problems by Hackbush [11] and McCormick [17].

3.2. The Cycle C MG algorithm. The particular MG algorithm that has been used in this study is based on the "Cycle C" algorithm described in Brandt [4]. This is an algorithm for iteratively solving the discrete equations approximating a linear elliptic problem on a given grid, through interaction with a hierarchy of coarser grids, taking advantage of the fact that the different discretizations on the different grids are all approximations to the same continuous problem. We note that there are other MG algorithms [4] proposed for implementing continuation procedures outside of the context of the pseudo arc-length framework. Some potential problems with these related algorithms are discussed in § 3.4. We do not know how well such MG algorithms perform and we hope to carry out our own investigation on such related methods in the future. In this paper, MG algorithms are used to solve the fine grid discrete equations that arise in the pseudo arc-length continuation procedure.

Consider a hierarchy of grids $(G^0, G^1, \cdots, G^M)$, with G^M being the finest one, defined on a domain Ω with corresponding mesh sizes $(h_0 > h_1 > \cdots > h_M)$, and all approximating the same linear elliptic problem:

$$LU = F \quad \text{on } \Omega, \qquad U = 0 \quad \text{on } \partial\Omega. \tag{3.1}$$

The discrete equation on a grid G^k is written as:

$$L^k U^k = F^k \quad \text{on } G^k, \qquad U^k = 0 \quad \text{on } \partial\Omega. \tag{3.2}$$

We are primarily interested in obtaining the approximating solution U^M on the finest grid, and we shall start with an initial guess on G^M and apply a standard relaxation procedure such as the Gauss–Seidel procedure. It is well known that the error is reduced rapidly in the first few iterations but then the reduction rate becomes very slow. By a frequency analysis, it can be shown that the fast reduction occurs when the residual (or the error) in the current iterate has large harmonics on the scale of the grid, the so-called high frequencies. Now at a stage in the iterative process where the error reduction rate slows down, let the current iterate be u^M. Define the error v^M in the iterate as $v^M = U^M - u^M$. Then the error v^M satisfies the following equation:

$$L^M v^M = F^M - L^M u^M = R^M \quad \text{on } G^M, \qquad v^M = 0 \quad \text{on } \partial G^M. \tag{3.3}$$

The residual R^M is computable, and hence the original problem of solving for U^M can be reduced to an equivalent one of solving (3.3) for v^M. There seems to be no obvious advantage in using (3.3) rather than continuing with the original relaxation procedure with u^M. However, if the error v^M and the residual R^M are *smooth* relative to G^M, that is, if their high frequency components have been smoothed out by the relaxation procedure, then we can *approximate* the solution of (3.3) on a *coarser* grid, say G^{M-1}, by solving:

$$\begin{aligned} L^{M-1} v^{M-1} = F^{M-1} &\equiv I_M^{M-1} R^M \quad \text{on } G^{M-1}, \\ v^{M-1} = 0 &\quad \text{on } \partial G^{M-1}, \end{aligned} \tag{3.4}$$

After this problem is solved we can interpolate the solution v^{M-1} onto G^M to get:

$$\text{new } u^M = \text{old } u^M + w_{M-1} I_{M-1}^M v^{M-1}, \tag{3.5}$$

where w_{M-1} is an interpolation factor, normally taking the value unity, and I_i^j stands for some interpolation operator from G^i to G^j. The solution process for equation (3.4) on G^{M-1} usually costs considerably less than the cost of solving equation (3.3) on G^M. If v^M is indeed smooth (relative to G^M), then G^{M-1} should provide adequate resolution for v^M and hence $I_{M-1}^M v^{M-1}$ should be a good approximation for v^M. This principle of transferring to a coarser grid when convergence slows down can be applied *recursively*. Thus for example, we can start with a zero initial guess for v^{M-1} in equation (3.4) and apply the Gauss–Seidel relaxation procedure to the iterates on G^{M-1}. When the convergence slows down, we can again transfer to the next coarser grid G^{M-2}, and so on. One can view the whole process as each grid smoothing just those frequencies in the error that are high relative to its own mesh size, each doing its job efficiently because these high frequencies are precisely those that are efficiently smoothed out by relaxation procedures.

The control of when to transfer between grids can follow a fixed strategy or an adaptive one. A fixed strategy could be of the following kind (see Nicolaides [19]): perform p relaxation sweeps on each grid G^k before transferring to a coarser grid G^{k-1}, and perform q relaxation sweeps before interpolating back to a finer grid G^{k+1}. An adaptive strategy could be as follows (see Brandt [4]): transfer to a coarser grid when the ratio of the residual norm of current iterate to the residual norm a sweep earlier is greater than some tolerance η, and transfer to a finer grid when the ratio of the residual norm of current iterate to the residual norm on the next finer grid is less than another tolerance δ. For simple problems like Poisson's equation on a square, the overall MG efficiency is very insensitive to which particular strategy is used and what values are used for (p, q) or (η, δ). We shall refer to the above particular fixed strategy the (p, q) strategy and the adaptive strategy the (η, δ) strategy.

3.3. Indefinite problems. In the Cycle C algorithm just described, convergence on the lowest (coarsest) grid G^0 is obtained by repeated relaxation sweeps. For positive definite matrices, convergence on G^0 can be guaranteed. For indefinite problems, however, convergence on G^0 cannot be obtained by repeated relaxation sweeps, because the components of the error that correspond to eigenfunctions with negative eigenvalues will grow as a result of relaxation sweeps (see the analysis in § 5). Therefore, for indefinite problems, a direct solution (e.g., Gaussian elimination) must be employed on the coarsest grid. If this coarsest grid is fine enough, it will also provide corrections to those growing components of the iterates on all finer grids. However, too fine a grid for G^0 will increase the cost of the direct solution procedure. Hence a little care must be taken regarding the size of the coarsest grid for indefinite problems. Fortunately, for "not too indefinite" problems G^0 can be chosen coarse enough so that the direct solution on G^0 will not affect the overall efficiency of the MG procedure seriously. Since indefinite problems occur frequently in nonlinear elliptic eigenvalue problems and, in particular, in our model problem, we shall use such a direct solution on G^0 whenever necessary.

3.4. Continuation methods. Brandt [4] suggested using continuation methods in conjunction with the MG procedure. His main idea is to use coarse grids for continuation, with little work and crude accuracy, and only use the finer grids at the final continuation step to achieve higher accuracy. We have not pursued this idea here. We believe that it will work as long as we stay away from singular points. Around a limit point, however, the solution branches corresponding to different grids may look like the situation in Fig. 3.1. If we continue on the coarse grid to λ^* and try to refine

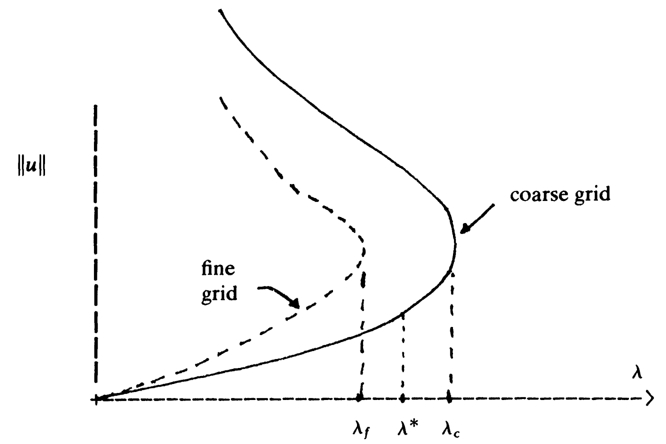


FIG. 3.1. *Limit points for different grids.*

using the finer grid, while keeping λ^* fixed, we cannot hope to obtain a fine grid solution because λ^* is larger than the fine grid limit point λ_f (i.e., no fine grid solution exists for $\lambda > \lambda_f$). In the opposite case, there is no coarse grid solution at λ^* so we cannot get started on that grid. Hence, in general, we have to be extremely careful in using MG methods and continuation around singular points.

4. Application to the Bratu problem.

4.1. Bratu's problem. As a typical example of a nonlinear elliptic eigenvalue problem, we consider the Bratu problem:

$$G(u,\lambda)=\Delta u+\lambda e^{u}=0 \quad \text{on } \Omega, \qquad u=0 \quad \text{on } \partial\Omega. \tag{4.1}$$

Equation (4.1) arises in many physical problems, for example, in chemical reactor theory, radiative heat transfer, and in modelling the expansion of the universe. The domain Ω is the unit interval $[0, 1]$ in R^1, or the unit square $[0,1]\times[0,1]$ in R^2, or the unit cube $[0,1]\times[0,1]\times[0,1]$ in R^3. There are no bifurcation points in this problem; all the singular points are limit points. The behavior of the solution near the singular points has been studied numerically [1], [26] and theoretically [14], [20], [21], [24]. Typical solution diagrams are shown in Fig. 4.1. For both the one- and

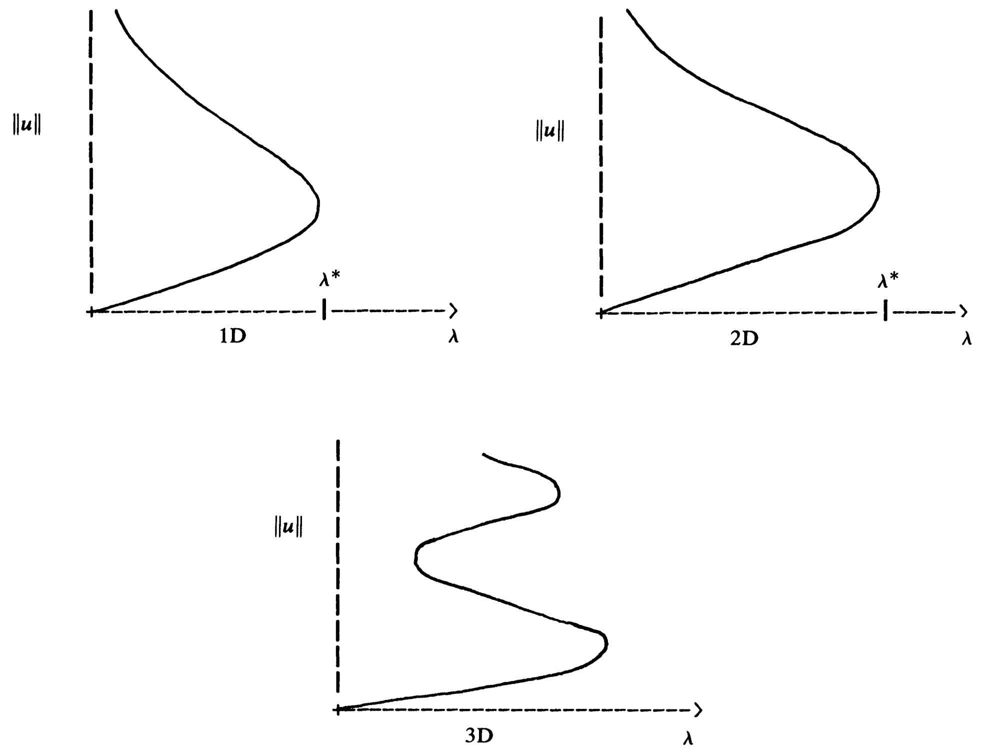


FIG. 4.1. *Solution for the Bratu problem.*

two-dimensional cases, the problem has exactly one limit point, whereas the three-dimensional case has infinitely many limit points (if Ω is a sphere). From now on we only consider the two-dimensional case, with Ω the unit square. For this case, the value of λ^* and the corresponding $\|u\|_\infty$ at the limit point are given by: $\lambda^* \cong 6.81$ and $\|u\|_\infty = u(0.5, 0.5) \cong 1.39$. For $\lambda > \lambda^*$, equation (4.1) has no solution, and for $\lambda < \lambda^*$, it has exactly two solutions.

4.2. Arc-length continuation with direct methods. We first apply the arc-length continuation method of § 2 to (4.1) using direct methods. For this problem, a trivial solution is $(u = 0, \lambda = 0)$. We can thus start at this trivial solution on the lower branch and march along the solution branch, past the limit point, and continue on to the upper solution branch. Since the only singular point in this problem is a limit point, this in principle presents no problem to the arc-length continuation procedure, although the step size might have to be reduced and controlled appropriately near the limit point. If desired, the limit point can be accurately determined by other related techniques [1], [13].

The derivatives of the operator G in equation (4.1) that are needed for the arc-length continuation technique are:

$$G_u = \Delta + \lambda e^u, \tag{4.2}$$

$$G_\lambda = e^u. \tag{4.3}$$

Now if we approximate the Laplacian operator by the standard five-point stencil on a uniform grid, the operator G_u will be approximated by the usual block tridiagonal matrix and the operator G_λ by a column vector.

In the application of the arc-length continuation technique, we will have to repeatedly solve linear systems of equations with the matrix given by G_u. The solution of these linear systems is the central part of the arc-length continuation method. Hence, an efficient linear system solver is crucial to the overall performance of the continuation technique. In this section, we present some computational results for Bratu's problem using a direct method (Gaussian elimination) of solution of the linearized difference equations. For large problems, this would be prohibitively expensive. However, the results here are intended to demonstrate the performance of the continuation procedure independent of the linear algebra method employed. In the next section, we shall investigate the use of multi-grid methods for solving the linear equations. It should be pointed out that G_u is generally not separable, and therefore we cannot use fast Poisson solvers directly even on rectangular domains. Moreover, this matrix is indefinite on the upper branch, and hence iterative methods like successive-over-relaxation cannot be used directly.

We present some of our computed results in Table 4.1 and Fig. 4.2. Only the behavior of the solution branch near the limit point for a few relatively coarse discretizations is presented. This is to be compared with the values: $\lambda^* = 6.80811698$ and $u(.5, .5) = 1.3916603$ for a grid with $h = \frac{1}{24}$ with the nine-point finite difference operator as computed by Abbott [1] and to the easily obtainable exact solution $(\lambda^* = 18/e \cong 6.62183, u^* = 1)$ for the case $h = \frac{1}{3}$. As expected, the step size $\partial s = s - s_0$ had to be suitably controlled near the limit point, but otherwise we encountered no difficulty in continuing past the limit point.

4.3. Arc-length continuation with multi-grid methods. In this section we discuss the use of MG methods, rather than direct methods, for solving the linear equations that arise in the continuation procedure. The MG method that we use was described in § 3 and Gauss–Seidel is the smoothing relaxation process. Since the Jacobian matrix G_u becomes indefinite on the upper branch, we use a direct method on the coarsest grid in the neighborhood of the limit point and on the upper branch.

We started the continuation procedure with the trivial solution $(u = 0, \lambda = 0)$, with $h = \frac{1}{4}$ on the coarsest grid, and a total of four levels of grids, making the finest grid with $h = \frac{1}{32}$. As expected, the MG method worked fine and we were able to

TABLE 4.1.
Computed results for Bratu's problem near limit point

h	λ	u	$\dot{\lambda}$	
	6.000000	0.619061	0.9841	
	6.485170	0.809435	0.9165	
$\frac{1}{3}$	6.572858	0.883052	0.7948	
	6.621830	0.999899	2.8889E−4	←limit point
	6.614022	1.04937	−0.4307	
	6.500000	1.00456	0.9632	
	6.689007	1.14350	0.9041	
$\frac{1}{24}$	6.802681	1.34995	0.2965	
	6.805499	1.39043	−1.1732E−4	←limit point
	6.805485	1.39368	−0.0125	

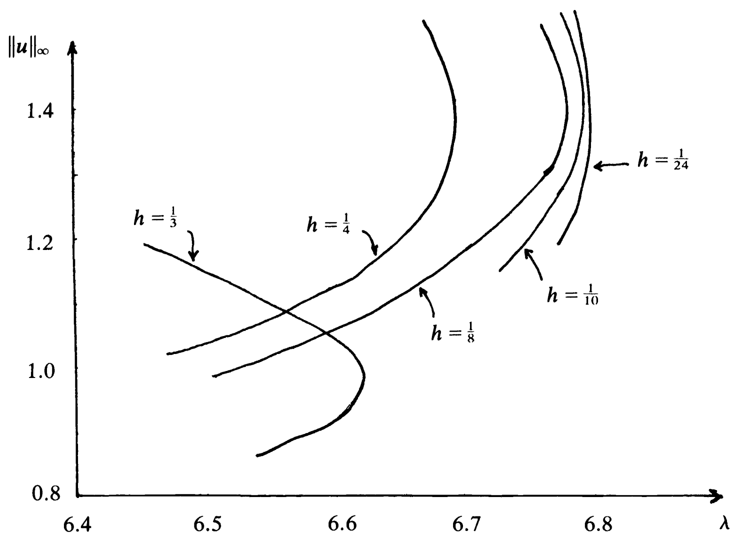


FIG. 4.2. *Computed results for Bratu's problem near limit point.*

continue up to very close to the limit point, at $\lambda \cong 6.804$ on the lower branch. However, we noticed that the convergence of the MG method deteriorates as we move in towards the limit point. For example, the number of equivalent relaxation sweeps on the finest grid required to reduce the residual norm by an order of magnitude, which is a convenient way of measuring the efficiency of MG methods, went from about 5 at $\lambda = 0$ to about 20 at $\lambda = 6.803$ and to *divergence* at $\lambda = 6.805$. The divergence occurred in the MG method and not in the Newton iteration. It is *not* due to the possible indefiniteness of the Jacobian matrix on the finest grid. This can occur near the limit point after a large Euler-predictor step. We performed other tests starting on the upper branch, away from the limit point, where the Jacobian matrix is indefinite, and here the MG method performed as efficiently as on the lower branch. From our experience, this divergence is strictly a phenomenon associated with the limit point, and to the best of our knowledge, has never been discussed or analyzed in the literature. We study this effect in § 5.

The exact value of λ at which this divergence first occurs varies slightly with the size of the coarsest grid h_0, but is quite independent of the other parameters of the Cycle C algorithm (e.g., η and δ). In all the cases we have run, this divergence made it impossible to continue past the limit point. Therefore, a remedy is needed. Before we can find one, we must understand the reason for the divergence.

5. Analysis of multi-grid methods for near-singular systems. For the present analysis, we assume that the linear operator L is self-adjoint and has the complete set of orthonormal eigenfunctions $\{\xi_1, \xi_2, \cdots\}$ with corresponding *real* eigenvalues $\{\mu_1 \leqq \mu_2, \cdots\}$. The operator G_u in the Bratu problem clearly satisfies the above hypothesis. Thus the solution U to $LU = F$ can be written as:

$$U = \sum_{i=1}^{\infty} a_i \xi_i, \qquad a_j = \langle \xi_j, F \rangle, \quad j = 1, 2, \cdots. \tag{5.1}$$

We assume that the discrete approximations L^k to the continuous L are *symmetric*. Thus they have *real* eigenvalues $\{\mu_1^k \leqq \mu_2^k \leqq \cdots \leqq \mu_{N_k}^k\}$ and a complete set of orthonormal eigenvectors $\{\xi_1^k, \xi_2^k, \cdots, \xi_{N_k}^k\}$. Here N_k is the dimension of the matrix representing L^k. For most reasonable approximations, and certainly for the five-point formula used for the Bratu problem on a rectangle, this is true.

Assume that after iterating (relaxing) on the grid G^k, convergence has slowed down and a transfer to the next coarser grid is desired. Let the current iterate be u^k, and the corresponding "correction" be v^k so that $U^k = u^k + v^k$ where U^k satisfies $L^k U^k = F^k$. The correction problem is given (as in § 3) by:

$$L^k v^k = R^k = F^k - L^k u^k \quad \text{in } G^k, \qquad v^k = 0 \quad \text{on } \partial G^k. \tag{5.2}$$

This is approximated on G^{k-1} by

$$L^{k-1} v^{k-1} = I_{k-1}^k R^k \quad \text{in } G^k, \qquad v^{k-1} = 0 \quad \text{on } \partial G^{k-1}. \tag{5.3}$$

Using the eigenvector expansion of v^k in (5.2) we get

$$v^k = \sum_{i=1}^{N_k} a_i^k \xi_i^k, \tag{5.4}$$

where

$$a_i^k = \frac{\langle R^k, \xi_i^k \rangle}{\mu_i^k}, \qquad i = 1, \cdots, N_k. \tag{5.5}$$

Suppose now that (5.3) is solved *exactly* (by either direct solution or Cycle C or any other means) on G^{k-1}. The solution v^{k-1} is then

$$v^{k-1} = \sum_{i=1}^{N_{k-1}} a_i^{k-1} \xi_i^{k-1}, \tag{5.6}$$

where

$$a_i^{k-1} = \frac{\langle I_k^{k-1} R^k, \xi_i^{k-1} \rangle}{\mu_i^{k-1}}. \tag{5.7}$$

The key idea in the MG method is that if v^k and R^k are smooth enough, they can be well approximated on G^{k-1}. Thus it is important for efficiency considerations that[1]

$$I_{k-1}^k v^{k-1} \cong v^k. \tag{5.8}$$

Using (5.4) and (5.6), this is equivalent to:

$$\sum_{i=1}^{N_{k-1}} a_i^{k-1} I_{k-1}^k \xi_i^{k-1} \cong \sum_{i=1}^{N_k} a_i^k \xi_i^k. \tag{5.9}$$

This will be the case if

$$\begin{array}{llll} (5.10) & \text{(a)} & I_{k-1}^k \xi_i^{k-1} \cong \xi_i^k, & 1 \leqq i \leqq N_{k-1}, \\ (5.11) & \text{(b)} & a_i^{k-1} \cong a_i^k, & 1 \leqq i \leqq N_{k-1}, \\ (5.12) & \text{(c)} & a_i^k \cong 0, & i > N_{k-1}. \end{array}$$

Conditions (5.10) and (5.11) ensure that the coarse grid correction v^{k-1} improves the *lower* modes of the iterate u^k. Condition (5.12) is essentially the smoothness required of v^k on G^k (i.e., negligible higher modes).

Now condition (5.10) is satisfied for the low frequency eigenfunctions of the continuous operator L if the grids G^k and G^{k-1} are both fine enough to resolve these eigenfunctions. This holds in many cases since the lower eigenfunctions of most second-order elliptic operators over smooth domains are very smooth. For the Bratu problem, the eigenfunctions are very close to products of sines and cosines (the eigenfunctions of the Laplacian operator) and so the lower modes are easily resolved by very coarse grids. Condition (5.11), on the other hand, turns out to be violated if the operator L^k is near singular. This is what caused the divergence of the Cycle C algorithm in the arc-length continuation procedure as we approach the limit point (see § 4.3). We shall analyze this case next.

From (5.5) and (5.7), condition (5.11) becomes

$$\frac{\langle I_k^{k-1} R^k, \xi_i^{k-1} \rangle}{\mu_i^{k-1}} \cong \frac{\langle R^k, \xi_i^k \rangle}{\mu_i^k}, \qquad i \leqq i \leqq N_{k-1}. \tag{5.13}$$

We claim that if condition (5.10) is satisfied, and if the transfer from G^k to G^{k-1} is done only after the residual R^k has been smoothed, then the numerators in (5.13) will have approximately the same value. To show this, we expand R^k as

$$R^k = \sum_{i=1}^{N_k} r_i \xi_i^k, \tag{5.14}$$

where

$$r_i = \langle R^k, \xi_i^k \rangle. \tag{5.15}$$

Thus the numerator on the right-hand side of (5.13) is precisely r_i. To estimate the numerator on the left hand side of (5.13), we proceed as follows:

$$I_k^{k-1} R^k = \sum_{i=1}^{N_k} r_i I_k^{k-1} \xi_i^k = \sum_{i=1}^{N_{k-1}} r_i I_k^{k-1} \xi_i^k + \sum_{i=N_{k-1}+1}^{N_k} r_i I_k^{k-1} \xi_i^k. \tag{5.16}$$

[1] We shall use the $\cong$ symbol to mean rather loosely "approximately equal to". The meaning should be clear by context. Also, we shall assume that the interpolation factor w_{k-1} in equation (3.5) is equal to one unless stated otherwise.

Now if condition (5.10) holds, its converse

$$I_k^{k-1}\xi_i^k \cong \xi_i^{k-1}, \qquad 1 \leqq i \leqq N_{k-1} \tag{5.17}$$

also holds. Also, if R^k has been smoothed on G^k, then r_i [for $N_{k-1} < i \leqq N_k$] must be small compared with r_i [for $1 \leqq i \leqq N_{k-1}$]. Alternatively (5.12) assumes $a_i^k = r_i/\mu_i^k \cong 0$ for $i > N_{k-1}$. Therefore, we can approximate in (5.16) by dropping the second sum on the right-hand side to get

$$I_k^{k-1}R^k \cong \sum_{i=1}^{N_{k-1}} r_i\xi_i^{k-1}. \tag{5.18}$$

Hence

$$\langle I_k^{k-1}R^k, \xi_i^{k-1}\rangle \cong r_i, \qquad 1 \leqq i \leqq N_{k-1}. \tag{5.19}$$

Therefore, from (5.15) and (5.19), we have, as claimed earlier,

$$\langle I_k^{k-1}R^k, \xi_i^{k-1}\rangle \cong \langle R^k, \xi_i^k\rangle \quad \text{for } 1 \leqq i \leqq N_{k-1}. \tag{5.20}$$

The relations in (5.20) imply that condition (5.13) will be true if

$$\frac{\mu_i^k}{\mu_i^{k-1}} \cong 1, \; 1 \leqq i \leqq N_{k-1}. \tag{5.21}$$

Actually, these conditions need to be *strengthened* in order to guarantee that the visit to G^{k-1} actually *improves* the accuracy of u^k. This can be seen as follows. The error in the iterate u^k *before* the transfer to G^{k-1} is given by

$$\text{old error} \equiv v^k = \sum_{i=1}^{N^k} a_i^k\xi_i^k. \tag{5.22}$$

From (3.5), the new error in u^k *after* coming back from a visit to G^{k-1} is given by

$$\text{new error} = v^k - w_{k-1}I_{k-1}^k v^{k-1}. \tag{5.23}$$

In view of (5.4) and (5.6), the above gives

$$\begin{aligned} \text{new error} &\cong \sum_{i=1}^{N_{k-1}} (a_i^k - w_{k-1}a_i^{k-1})\xi_i^k + \text{higher modes} \\ &\cong \sum_{i=1}^{N^{k-1}} \left(1 - \frac{w_{k-1}a_i^{k-1}}{a_i^k}\right) a_i^k\xi_i^k + \text{higher modes}. \end{aligned} \tag{5.24}$$

From (5.5), (5.7) and (5.20), we have

$$\frac{a_i^{k-1}}{a_i^k} \cong \frac{\mu_i^k}{\mu_i^{k-1}},$$

and therefore we can write the new error in (5.24) as

$$\text{new error} \cong \sum_{i=1}^{N_{k-1}} \left(1 - \frac{w_{k-1}\mu_i^k}{\mu_i^{k-1}}\right) a_i^k\xi_i^k + \text{higher modes}. \tag{5.25}$$

For obvious efficiency and convergence considerations, the new error should preferably be less than the old error, at least for the lower modes. In other words, condition (5.21) should be strengthened to

$$\left|1 - \frac{w_{k-1}\mu_i^k}{\mu_i^{k-1}}\right| < 1, \tag{5.26}$$

i.e.,

$$0 < \frac{w_{k-1}\mu_i^k}{\mu_i^{k-1}} < 2 \quad \text{for } 1 \leqq i \leqq N_{k-1}. \tag{5.27}$$

Now if the ratios of eigenvalues in (5.21) are not close to unity, the interpolation factors, w_{k-1}, should be chosen so that condition (5.27) is satisfied. Otherwise the new error can be larger than the old error in some modes.

It should be pointed out that, in general, condition (5.27) is not *necessary* for the convergence of the "Cycle C" algorithm. This is the case, for instance, if L and the L^k's are all positive definite. Then Gauss–Seidel sweeps on any grid G^k will reduce the amplitude of *every* mode present in the error. In such cases, convergence on any grid can be achieved by merely doing enough relaxation sweeps. Then it is not necessary for the next coarser grid to provide any improvement on the current iterate, although it would obviously improve the efficiency of the overall algorithm if it does so. In fact, the MG method derives its efficiency from the very fact that the coarser grids *do* provide improvements in the current iterate u^k in the lower modes. These are precisely those modes that have poor convergence rates for the relaxation sweeps on G^k. Thus, even in the positive definite case, it is important (from an efficiency viewpoint) that conditions (5.27) hold, at least for small i's.

If the operator L and the L^k's are indefinite the situation is different because some modes will grow if we simply perform relaxation sweeps on a fixed grid. Such modes have to be corrected by going to coarser grids and using a direct method on the coarsest grid. Further, the interpolation factors, w_{k-1}, should be chosen such that condition (5.27) is satisfied for these modes. Condition (5.27) has been suggested by Brandt [4] for indefinite problems. However, as we show later, most nonlinear eigenvalue problems with limit points and bifurcation points abound with indefinite operators, but they do not cause difficulties in the sense of violating condition (5.27). Essentially only one mode causes problems on each G^k and it is the mode that corresponds to the eigenvalue that is nearest zero as the singular point is approached. Merely including the interpolation factors so that condition (5.27) is satisfied turns out to be very inefficient. Further, it is not clear that such factors, w_{k-1}, can be found at all in this case.

Another source of difficulty is that the process of interpolating v^{k-1} into G^k introduces high frequency errors. That is, the exact relation corresponding to (5.10) is:

$$I_{k-1}^k \xi_i^{k-1} = \xi_i^k + \sum_{j=1}^{N_k} b_{ij}^k \xi_j^k, \qquad i = 1, 2, \cdots, N_{k-1}, \quad \text{for } 1 \leqq i \leqq N_{k-1}, \tag{5.28}$$

and the coefficients b_{ij}^k may be large for $j > N_{k-1}$. This would result in a violation of (5.12). Fortunately, these high frequency errors are very efficiently smoothed out by the subsequent relaxation sweeps on G^k, and thus these errors are automatically corrected.

For elliptic operators which are "far" from being singular and with a reasonable grid system $\{G^k\}$ condition (5.27) can be assured. For example, if L is the negative Laplacian, $-\Delta$, on a unit square with Dirichlet boundary conditions, then it is known (e.g., [9]) that the eigenvalues of L are given by

$$\mu_{m,n} = (m\pi)^2 + (n\pi)^2. \tag{5.29}$$

The corresponding eigenfunctions are:

$$\xi_{m,n} = \sin(m\pi x)\sin(n\pi y). \tag{5.30}$$

These eigenfunctions evaluated at the discrete interior grid points of a uniform mesh on the unit square give the eigenfunctions of the discrete 5-point approximations, $L^k = -\Delta_h$, with h being the uniform mesh size. The eigenvalues of L^k are, with $\delta x = \delta y = h_k$,

$$\mu^k_{m,n} = \frac{4[\sin^2(m\pi h_k/2) + \sin^2(n\pi h_k/2)]}{h_k^2}. \tag{5.31}$$

Some of these eigenvalues are tabulated in Table 5.1 for various mesh sizes, h_k. The ratios $\mu^k_{m,n}/\mu^{k-1}_{m,n}$ are given in Table 5.2. We see from Table 5.2 that condition (5.27)

TABLE 5.1.
$\mu^k_{m,n}$ *for* $-\Delta_{h_k}$

$k=$	0	1	2	3	∞
(m, n)	$h_0=\frac{1}{2}$	$h_1=\frac{1}{4}$	$h_2=\frac{1}{8}$	$h_3=\frac{1}{16}$	$h_\infty=0$
1, 1	16.0	18.745	19.487	19.676	19.739
2, 1	NA	41.37258	47.238	48.812	49.348
1, 2	NA	41.37258	47.238	48.812	49.348
2, 2	NA	64.0	74.981	77.947	78.957
3, 1	NA	NA	88.760	96.126	98.696
1, 3	NA	NA	88.760	96.126	98.696
3, 2	NA	NA	116.507	125.261	128.305
2, 3	NA	NA	116.507	125.261	128.305
3, 3	NA	NA	158.033	172.575	177.653

TABLE 5.2.
Ratios $\mu^k_{m,n}/\mu^{k-1}_{m,n}$ *for* $-\Delta_{h_k}$

(m, n)	$h_k=\frac{1}{4}, h_{k-1}=\frac{1}{2}$	$h_k=\frac{1}{8}, h_{k-1}=\frac{1}{4}$	$h_k=\frac{1}{16}, h_{k-1}=\frac{1}{8}$
1, 1	1.17	1.04	1.01
2, 1	NA	1.14	1.03
1, 2	NA	1.14	1.03
2, 2	NA	1.17	1.04
3, 1	NA	NA	1.08
1, 3	NA	NA	1.08
3, 2	NA	NA	1.08
2, 3	NA	NA	1.08
3, 3	NA	NA	1.09

is satisfied, with $w_{k-1} \cong 1$, for all lower modes shown. These ratios are very close to unity, even for the case where the coarsest grid has only one interior point. We have seen from condition (5.11) that this closeness to unity is very desirable and this fact partly explains the well-documented success of MG methods for the Laplacian operator.

Near the limit point of the Bratu problem, the operator $L \equiv G_u = \Delta + \lambda e^u$ behaves very much like a shifted Laplacian operator. Clearly, if the factor e^u were replaced by a constant, α say, then G_u is replaced by the Laplacian operator with a shift $\alpha\lambda$.

Then the eigenvalue ratio $\mu_{1,1}^{k}/\mu_{1,1}^{k-1}$, valid for $\alpha\lambda = 0$, is replaced by:

$$\frac{\mu_{1,1}^{k} - \alpha\lambda}{\mu_{1,1}^{k-1} - \alpha\lambda}. \tag{5.32}$$

Since $0 < u < 1.4$, the factor e^{u} does not vary much and we assume this approximation to be valid for some $\alpha > 0$. The situation is depicted graphically in Fig. 5.1 for the grid system that was used for Table 5.1. As the shift $\alpha\lambda$ approaches the group of eigenvalues corresponding to the (1, 1) mode from below, the ratios in (5.31) increase. As $\alpha\lambda$ continues to increase, the ratio of eigenvalues will become greater than 2, then increase towards $+\infty$, jump to $-\infty$ discontinuously, and start increasing from $-\infty$ to 1. The situation is depicted in Fig. 5.2.

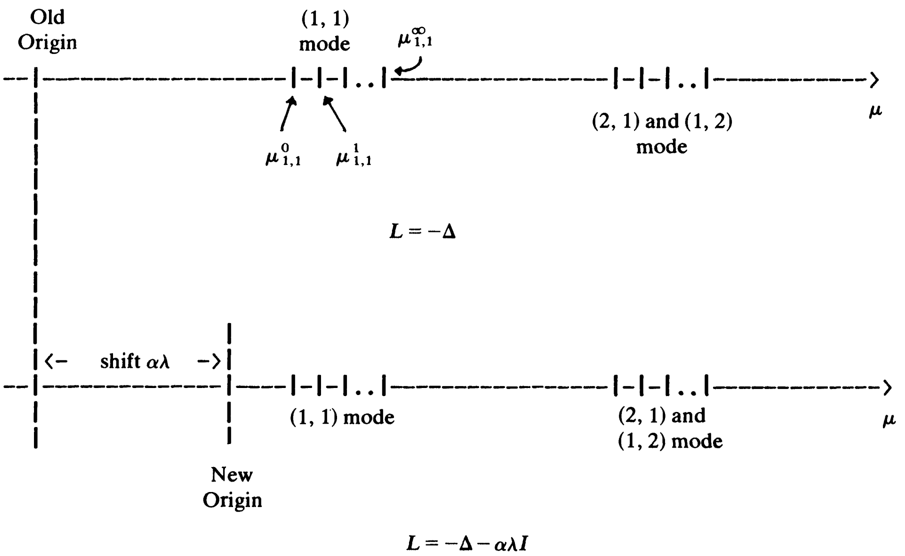


FIG. 5.1. *Spectrum of shifted Laplacian.*

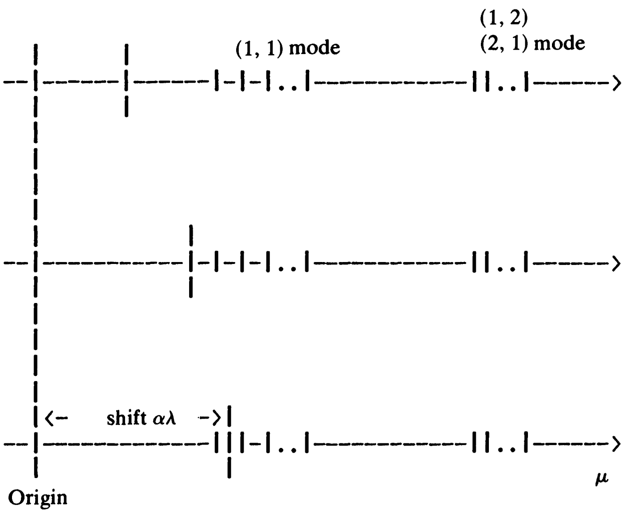


FIG. 5.2. *Spectrum near singular point.*

We thus see, under the above assumptions, that condition (5.27) is first violated by the lowest mode (i.e., the (1, 1) mode) on the two coarsest grids G^0 and G^1. In fact the lowest eigenvalues for the Bratu problem computed at the first point on the solution branch where Cycle C diverged, yields the ratio almost exactly 2. On the other hand, even at this point, condition (5.27) is satisfied by the (1, 1) modes on the finer grids. In other words, the divergence of Cycle C is seen to be caused by *one* near-singular grid out of the whole hierarchy of grids present. The mode that becomes singular at the limit point of the Bratu problem is the (1, 1) mode, and this occurs first on the G^0 grid. As the limit point is approached, L^k on some of these grids may even become indefinite, while others (the finer grids) may still be positive definite. Essentially, the near-singular grid causes the (1, 1) mode component of the correction v^{k-1}, when viewed as an approximation to v^k, to have the right *direction*, but the wrong *magnitude*. This phenomenon is not limited to the Bratu problem. The only thing special about this problem is that it is the eigenvalue of the (1, 1) mode that becomes zero at the limit point. For other problems, the eigenvalue of the operator L that becomes zero as the singular point is approached might correspond to other modes. Although the singular point in the Bratu problem is a limit point, we can expect the same behavior at a bifurcation point.

Having now understood the cause of the divergence of the MG method, in the next section we shall discuss some modifications to the basic Cycle C algorithm that are designed to overcome such difficulties.

6. Remedies and new algorithms. In this section we discuss approaches that have been devised to overcome the difficulties with the MG method near singular points. The first goal is to modify the basic Cycle C algorithm so that it will converge for values of λ close enough to the limit point so that the arc-length continuation procedure can take us past the limit point onto the upper solution branch. A more ambitious goal is to modify Cycle C further so that it will converge arbitrarily close to the singular point. Such an algorithm, when used in conjunction with the arc-length continuation technique for tracing solution branches, will make the overall algorithm much more robust. Moreover, such an algorithm may prove to be useful for locating singular points accurately, either using an arc-length continuation based procedure [13], or some other procedure that uses the operator G_u near the singular point [22]. We shall see that the first goal is relatively easy to achieve, whereas the second goal is much more difficult. However, we have devised a Cycle C based algorithm that has performed very well when applied very close to the limit point. The approaches that we have tried and that lead to the final algorithm will be discussed in this section. We shall describe them in the sequence that they were tried.

Before we proceed, however, we have to explain a few general strategies that were used. First of all, Gauss–Seidel and many other relaxation schemes are not very effective in smoothing the lower modes, especially modes with near-zero eigenvalues. Hence, these modes must be eliminated by means other than relaxation, even on the coarsest grid. Therefore, unless stated otherwise, we shall use a direct solution on the coarsest grid even though the operators L^k's may be *positive definite*. This does not affect the overall efficiency very much because the coarsest grid has so few points that direct solution is very fast and efficient.

Another strategy concerns the treatment of the mode that causes the divergence, that is, the mode with a near-zero eigenvalue, say ξ_1. In all the algorithms that are discussed, this mode is treated *separately* from the other modes. To do this, it is essential to have approximations to this mode and to its corresponding eigenvalues, say $\bar{\xi}_1^k$ and $\bar{\mu}_1^k$, respectively. Here we have to strike a balance between accuracy and efficiency. If we compute the ξ_1^k exactly, then we can completely eliminate the ξ_1^k error components on each grid. Thus, the problem on G^k can be reduced to one in which a_1^k is zero (see (5.25)). When this is done, we do not need to satisfy condition (5.27) for this mode. On the other hand, the work involved in computing accurate approximations to μ_1^k and ξ_1^k for each k would be at least as much as solving the original linear system. Our compromise has been to compute an approximation $\bar{\xi}_1^0$ to ξ_1 on the coarsest grid, G^0, by a few steps of inverse iteration with zero shift (since the eigenvalue we want is near zero). This is very inexpensive since G^0 is quite coarse and the LU factors of L^0 are already available. Then we interpolate $\bar{\xi}_1^0$ onto the finer grids. To eliminate the high frequency errors introduced in these interpolations, we do two things: (1) use higher order interpolation, e.g., cubic instead of linear; (2) smooth the interpolated eigenfunctions by performing a few relaxation sweeps on $L^k\xi_1^k = 0$. Estimates of the eigenvalues, $\bar{\mu}_1^k$, are then computed using the Rayleigh quotients: $\langle \bar{\xi}_1^k, L^k\bar{\xi}_1^k\rangle$. We view this as a preprocessing phase of the algorithm and the extra work is usually minimal compared to the overall work. Furthermore, since the eigenfunctions (not the eigenvalues) do not change very much in the neighborhood of the singular points, we can use the same approximation for different linearized operators L^k. The storage required to store these eigenfunctions is less than twice the size of the finest grid.

We use the (η, δ) adaptive version of the Cycle C algorithm, unless otherwise stated. The first modified algorithm is the following.

6.1. Under- and over-interpolation. The idea is to choose w_{k-1} in (3.5) for interpolation onto G^k, such that condition (5.27) is satisfied for ξ_1. Clearly the value

$$w_{k-1} = \frac{\bar{\mu}_1^{k-1}}{\bar{\mu}_1^k} \tag{6.1}$$

is in some sense optimal since it eliminates the ξ_1 term in (5.25). For the case discussed in § 4.3, this modification allows the computation to continue past the point $\lambda = 6.804$, where divergence of Cycle C first occurred. In fact (with a little luck) we succeeded in continuing around the limit point onto the upper branch. Here the eigenfunction ξ_1 no longer presented difficulties for the MG algorithm. For some of these cases μ_1^0 is actually negative and therefore (6.1) yields a negative value for w_1. In this case the transfer from G^0 to G^1 violates condition (5.27) for all modes *other* than ξ_1. The errors in these modes must be reduced by extra relaxation sweeps on G^1. In other words G^0 only provides a proper correction on G^1 for the ξ_1 mode, all higher modes are treated incorrectly during the transfer. The efficiency of the algorithm thus suffers. This effect is especially pronounced if some factors w_k are either very large or negative or (worse) both. The algorithm is very sensitive to the parameters (η, δ) and thus is not robust. It can even diverge if the higher modes are not reduced fast enough on G^k after the transfer from G^{k-1}.

Even worse, the above algorithm will not work for indefinite problems in which some intermediate eigenvalue is near zero. For example, if the spectra of the L^k are similar to those in Fig. 6.1, the interpolation factors w_k are controlled by the ξ_1^k belonging to eigenvalues μ_1^k near zero. On the other hand, the eigenfunctions ξ_{-1}^k require that condition (5.27) be satisfied because these modes cannot be liquidated by relaxation. Conflicts can occur when ξ_1^k requires w_k to be negative while ξ_{-1}^k requires w_k to be positive. Indefinite problems of this type occur frequently in nonlinear eigenvalue problems. Mere under- or over-interpolation must run into difficulties for such problems, near the singular points.

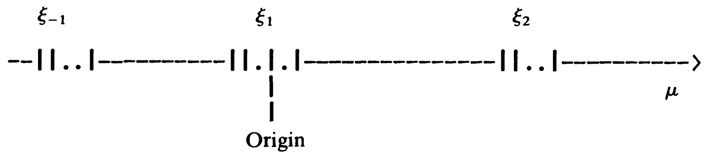


FIG. 6.1. *Intermediate eigenvalue near zero.*

The above considerations make it clear that the eigenfunction with the near-zero eigenvalue must be isolated and treated differently from the other eigenfunctions. We use the approximate eigenfunctions that are computed in the preprocessing phase for this purpose in the following procedure.

6.2. Under- and over-interpolating the singular eigenfunction only. We use an interpolation different from that in (3.5). Specifically if

$$v^{k-1} = \sum_{i=1}^{N_{k-1}} a_i^{k-1} \xi_i^{k-1} \tag{6.2}$$

on G^{k-1}, we interpolate it onto G^k by

$$v^k = w_{k-1} a_1^{k-1} I_{k-1}^k \xi_1^{k-1} + I_{k-1}^k \sum_{i=2}^{N_{k-1}} a_i^{k-1} \xi_i^{k-1}. \tag{6.3}$$

Further w_{k-1} is chosen to satisfy (6.1). Since we only have an approximation to ξ_1^k, we use, instead of (6.3):

$$v^k = I_{k-1}^k [v^{k-1} - \langle v^{k-1}, \bar{\xi}_1^{k-1} \rangle \bar{\xi}_1^{k-1}] + w_{k-1} \langle v^{k-1}, \bar{\xi}_1^{k-1} \rangle I_{k-1}^k \bar{\xi}_1^{k-1}. \tag{6.4}$$

In practice, this performed much better than indiscriminate under- and over-interpolation described in § 6.1. It was the more efficient when both procedures worked. In many cases when (6.1) yields large and/or negative values for w_k, only the current scheme converges. In principle, it will also work for indefinite problems like that depicted in Fig. 6.1. The efficiency in most cases was very respectable: in the range of 6–10 units per order of magnitude reduction in the residual. It is also quite insensitive to the parameters (η, δ). Thus, it can be used very efficiently and reliably with the arc-length continuation procedure for tracing out solution branches.

Unfortunately, this improved algorithm fails when the magnitude of w_k becomes too large. This occurs when L^k is very nearly singular, that is, with μ_1^k very close to zero. Since we only have an approximation $\bar{\xi}_1^k$ to ξ_1^k, large factors w_k in (6.4) introduce very large errors in the other modes. Moreover, the estimates $\bar{\mu}_1^k$ using Rayleigh quotients tend to be too large (relatively) when μ_1^k is very small. Then (6.1) gives a value of w_k that is too small. Both of the above result in lower efficiency and reliability. In extreme cases, this makes the algorithm impractical. To overcome this difficulty, we devise an algorithm that will work even if one of the operators L^k is very nearly singular. For this we employ the idea of skipping a grid.

6.3. Skipping the singular grid. The previous algorithm fails if the operator is very nearly singular on one of the grids, say G^k. The idea here is to simply delete this grid from the hierarchy of grids used by the MG algorithm. If the remaining grids are not as singular as the deleted grid it would seem that the algorithm described in § 6.2 should work. However, calculations show that skipping a grid can cause other problems. When G^k is skipped, the mesh changes more drastically from G^{k-1} to G^{k+1}, and hence the interpolation in (6.4) (now I_{k-1}^{k+1} instead of I_{k-1}^{k}) introduces larger errors into the higher modes on G^{k+1}. These high-frequency errors can cause divergence of the MG process unless controlled properly by the parameters (η, δ). A large value of η, say between 0.8 and 0.9, makes the algorithm more robust but involves more work than for a smaller value of η, say 0.5. We encountered a case where, with all else the same, the new skipping algorithm converges for $\eta = 0.9$ but diverges for $\eta = 0.6$. Granted with $\eta = 0.9$ the algorithm may be very reliable, such sensitivity to one parameter is very undesirable. Therefore, we considered the following modification.

6.4. Skipping the singular grid for the singular eigenfunction only. The idea is to skip the singular grid G^k for ξ_1 only, and to keep it for smoothing the other modes. In the actual implementation, we modify the algorithm described in § 6.2 to use

$$w_{k-1} = \frac{\bar{\mu}_1^{k-1}}{\bar{\mu}_1^{k+1}} \tag{6.5}$$

for ξ_1 and $w_{k-1} = 1$ for all other modes to transfer from G^{k-1} to G^k and, after a few smoothing sweeps on G^k, transfer to G^{k+1} with $w_k = 1$ for all modes. Note that we do not try to solve the G^k equations for v^k. Trying to do that would result in large magnification of the ξ_1^k component in v^k, since μ_1^k is near zero. This would in turn cause problems during the transfer to G^{k+1}.

In addition, we have experimented with using a mixture of the adaptive (η, δ) strategy with the nonadaptive (p, q) strategy (cf. § 3.2). We have found an (η, q) strategy that is as good as any other we have tried. In this strategy, we use η to control when we terminate relaxation on a certain grid and go on to a coarser grid, and use q to control how many sweeps to do on a grid after transfer from a coarser grid before interpolating onto a finer grid. A typical set of parameters that worked well is $(\eta = 0.6, q = 2)$. The resulting algorithm is fairly insensitive to actual values of η and q and is quite robust. It is also quite efficient. It consistently achieved an efficiency of less than about 12 units per order of magnitude reduction in the residual for most problems that we have encountered. Some of these problems have very singular grids which presented difficulties for all of the previous algorithms.

7. Summary. In this paper, we study arc-length continuation techniques and multi-grid techniques for solving nonlinear elliptic eigenvalue problems. We have applied these techniques to solve a model nonlinear elliptic eigenvalue problem (the Bratu problem). We have found that as long as we stay away from singular points, the two techniques combined to give a very powerful and efficient procedure for tracing solution branches. Near singular points, however, the standard multi-grid method has difficulty converging on the linearized elliptic systems that arise in the continuation procedure. One consequence is that we cannot continue past the limit point in the model problem. This divergence is successfully analyzed and several modified multi-grid algorithms have been designed based on this analysis. The best of these modified algorithms performs efficiently and reliably arbitrarily close to the singular points. This enables the continuation procedure to continue past the limit point with no difficulty. It seems reasonable that this modified multi-grid algorithm can be useful in more general situations where nearly singular elliptic systems arise, such as in inverse iteration [11], [17].

REFERENCES

[1] J. P. Abbott, *An efficient algorithm for the determination of certain bifurcation points*, J. Comput. Appl. Math., 4 (1978), pp. 19–27.

[2] H. Amann, *Fixed point equations and nonlinear eigenvalue problems in ordered Banach spaces*, SIAM Rev., 18 (1976), pp. 620–709.

[3] N. S. Bakhvalov, *Convergence of a relaxation method with natural constraints on an elliptic operator*, Z. Vyčisl. Mat. i Mat. Fiz., 6 (1966), pp. 861–885. (In Russian.)

[4] A. Brandt, *Multi-level adaptive solution to boundary value problems*, Math. Comp., 31 (1977), pp. 333–390.

[5] T. F. Chan, *Numerical computation of large amplitude internal solitary waves*, Computer Science Report # 198, Yale University, New Haven, CT, Feb. 1981.

[6] B. Chen and P. Saffman, *Numerical evidence for the existence of new types of gravity waves of permanent form on deep water*, Stud. Appl. Math., 62 (1980), pp. 1–21.

[7] D. W. Decker and H. B. Keller, *Path following near bifurcation*, Comm. Pure Appl. Math., 34 (1981), pp. 149–175.

[8] R. P. Federenko, *A relaxation method for solving elliptic difference equations*, Z. Vyčisl. Mat. i Mat. Fiz., 1 (1961), pp. 922–927. (In Russian.)

[9] G. E. Forsythe and W. R. Wasow, *Finite difference methods for partial differential equations*, John Wiley, New York, 1960.

[10] W. Hackbush, *On the multi-grid method applied to difference equations*, Computing, 20 (1978), pp. 291–306.

[11] ———, *On the computation of approximate eigenvalues and eigenfunctions of elliptic operators by means of a multi-grid method*, SIAM J. Numer. Anal., 16 (1979), pp. 201–215.

[12] H. B. Keller, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359–384.

[13] ———, *Global homotopies and Newton methods*, in Recent Advances in Numerical Analysis, Carl de Boor and Gene Golub, eds., Academic Press, New York, 1978, pp. 73–94.

[14] H. B. Keller and D. S. Cohen, *Some positone problems suggested by nonlinear heat generation*, J. Math. and Mech., 16 (1967), pp. 1361–1376.

[15] H. B. Keller and R. Schreiber, *Accurate solutions for the driven cavity*, in preparation.

[16] M. Lentini and H. B. Keller, *The von Karman swirling flows*, SIAM J. Appl. Math., 38 (1980), pp. 52–64.

[17] S. F. McCormick, *A mesh refinement method for $Ax = \lambda Bx$*, manuscript.

[18] H. D. Mittelmann and H. Weber, *Numerical methods for bifurcation problems—A survey and classification*, in Bifurcation Problems and their Numerical Solution, Workshop on Bifurcation Problems and their Numerical Solution, January 15–17, Dortmund, 1980, pp. 1–45.

[19] R. A. Nicolaides, *On multiple grid and related techniques for solving discrete elliptic systems*, J. Comp. Phys., 19 (1975), pp. 418–431.

[20] S. V. Parter, *Mildly nonlinear elliptic partial differential equations and their numerical solution*. I, Numer. Math., 7 (1965), pp. 113–128.

[21] ———, *Maximal solutions of mildly nonlinear elliptic equations*, in Numerical Solution of Nonlinear Differential Equations, D. Greenspan, ed., John Wiley, New York, 1966, pp. 213–238.

[22] W. C. Rheinboldt, *Numerical methods for a class of finite dimensional bifurcation problems*, SIAM J. Numer. Anal., 15 (1978), pp. 1–11.

[23] ———, *Solution fields of nonlinear equations and continuation methods*, SIAM J. Numer. Anal., 17 (1980), pp. 221–237.

[24] J. B. Rosen, *Approximate solution and error bounds for quasilinear elliptic boundary value problems*, SIAM J. Numer. Anal., 8 (1970), pp. 80–103.

[25] S. Rosenblat and R. Szeto, *Multiple solutions of nonlinear boundary value problems*, Stud. Appl. Math., 63 (1980), pp. 99–117.

[26] R. B. Simpson, *A method for the numerical determination of bifurcation states of nonlinear systems of equations*, SIAM J. Numer. Anal., 12 (1975), pp. 439–451.

[27] R. K. H. Szeto, *The flow between rotating coaxial disks*, Ph.D. thesis, California Institute of Technology, Pasadena, CA, 1978.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

PIECEWISE ANALYTICAL PERTURBATION SERIES SOLUTIONS OF THE RADIAL SCHRÖDINGER EQUATION: ONE-DIMENSIONAL CASE*

MITCHELL D. SMOOKE†

Abstract. We develop a piecewise analytical perturbation series method (PAPSM) for solving the radial Schrödinger equation. The method centers around seeking a perturbation series solution when the coefficient function of the radial Schrödinger equation is approximated by piecewise constant polynomials. We perform a number of numerical experiments designed to evaluate the accuracy and efficiency of PAPSM as compared to Gordon's piecewise analytical solution method. The calculations are performed for both scattering and bound state problems.

Key words. Schrödinger equation, piecewise analytical, perturbation series, two-point boundary value problem.

1. Introduction.

We want to consider the following one-dimensional two-point boundary value problem:

$$\begin{aligned} &-u''(r)+p(r)u(r)=0, \qquad 0<r<\infty,\\ &u(0)=0, \\ &u(r)\to u_\infty \quad \text{as } r\to\infty, \end{aligned} \tag{1.1}$$

where $p(r)$, the potential function, has the following properties:

$$\begin{aligned} &r^2|p(r)|\to\infty \quad \text{as } r\to 0,\\ &|p(r)|\to \text{constant} \quad \text{as } r\to\infty. \end{aligned} \tag{1.2}$$

The problem stated in (1.1) and (1.2) occurs in a variety of physical contexts. Of particular importance to us is the relation between (1.1) and (1.2) and problems encountered in quantum chemistry. Using the proper changes of variables, Messiah [24] shows that one can write the radial Schrödinger equation in a form similar to the differential equation in (1.1). Moreover, elastic scattering [25] and bound state eigenvalue problems [29] can be formulated as two-point boundary value problems similar to (1.1). For these, $p(r)$ is defined by

$$p(r)=-E+\frac{l(l+1)}{r^2}+v(r), \tag{1.3}$$

for some energy E (specified in the scattering problem and to be determined in the bound state problem), specified orbital angular momentum quantum number l and interaction potential $v(r)$.

If $p(r)$ is of such a form that an analytical solution to (1.1) cannot be obtained, then a numerical technique must be used. This requires solving the problem on a mesh

$$\mathcal{M}=\{0<\alpha=R_0<R_1<\cdots<R_M=\beta<\infty\}, \tag{1.4}$$

where we set $h_k=R_k-R_{k-1}$, $k=1,2,\cdots,M$. By imposing boundary conditions at α and β we can reduce the problem in (1.1) to a finite domain. This leads us to consider

* Received by the editors August 15, 1980, and in final form October 6, 1981. This work was supported by the National Science Foundation under contract NSF-MPS75-15469.

† Applied Mathematics Division, Sandia National Laboratories, Livermore, California 94550.

the two-point boundary value problem

$$\begin{aligned} &-u''+pu=0, \qquad \alpha<r<\beta, \\ &c_1u(\alpha)+c_2u'(\alpha)=c_3, \qquad |c_1|+|c_2|\neq 0, \\ &c_4u(\beta)+c_5u'(\beta)=c_6, \qquad |c_4|+|c_5|\neq 0, \end{aligned} \tag{1.5}$$

for constants c_i, $i=1,2,\cdots,6$. We remark that in solving (1.5) on the mesh $\mathcal{M}$, the points R_k, $k=0,1,\cdots,M$, are generally not equally spaced.

Although there is a variety of methods for solving the elastic scattering problem [5], [16], [21], [26] and the bound state eigenvalue problem [4], [6]–[7], [11], [19], we focus our attention here on a method studied by Gordon et al. [12], [14]–[15], [28], Canosa and De Oliveira [9], Ixaru et al. [2], [17]–[18], Riehl, Diestler and Wagner [27], Luthey [23], and more recently by Smooke [30]. The method involves replacing the coefficient function $p(r)$ by low degree piecewise polynomials $\tilde{p}(r)$ (constant, linear and quadratic) so that the resulting zeroth order equation can be solved analytically. The rationale behind this kind of approximation lies in the fact that, for a number of problems of physical interest, the scale of variation of $u(r)$ is often small compared to the scale of variation of $p(r)$. As a result, the approximating scheme allows larger mesh intervals, or equivalently, a smaller number of mesh points than if, for example, a conventional finite difference scheme had been used. The method is computationally efficient providing the complexity of the calculation hasn't been shifted to evaluation of the functions comprising the analytical solution of the zeroth order equation.

While all of the authors mentioned in connection with the piecewise analytical solution method (PASM) have considered zeroth order approximations to (1.5) only Ixaru et al. [2], Riehl, Diestler and Wagner [27] and sometimes Gordon [15] have included a first perturbation correction in their approximation to (1.5).

The approach we shall take will be to rewrite (1.5) in the form

$$-u''+\tilde{p}u=-(p-\tilde{p})u, \tag{1.6}$$

and seek a perturbation series solution to u on the premise that $(p-\tilde{p})u$ is small compared to terms on the left hand side of (1.6). As in the zeroth order work of Gordon and others, we will require that our perturbation series solution be determined analytically. This will lead us to consider piecewise constant polynomial approximations to $p(r)$ with $(p-\tilde{p})$ expressed in some polynomial-exponential combination. We will then be able to obtain a piecewise analytical perturbation series solution by the method of undetermined coefficients.

The motivation for our piecewise analytical perturbation series method (PAPSM) lies in the fact that by taking our approximate solution as a piecewise constant zeroth order solution plus a number of perturbation corrections to this zeroth order solution, we will obtain a more accurate approximation to u than if we had not included any perturbation corrections. This will manifest itself in a smaller number of mesh intervals needed to solve the problem numerically. Hence, providing the complexity of evaluating the perturbation corrections is not too prohibitive, we will reduce the overall cost of the calculation.

The paper is organized as follows: in the next section we reformulate the two-point boundary value problem in (1.5) to obtain a piecewise analytical perturbation series solution for piecewise constant polynomial approximations to $p(r)$. In § 3 we discuss the numerical algorithms used in implementing our piecewise analytical perturbation series method in both scattering and bound state problems. Finally in § 4 we present our numerical results and compare the piecewise analytical perturbation series method with the zeroth order piecewise analytical solution method of Gordon [14].

2. Formulation of the piecewise analytical perturbation series method. To develop the formalism necessary to solve (1.5) by a piecewise perturbation series solution, we first let $\tilde{p}(r)$ be a piecewise polynomial approximation to $p(r)$ and imbed our problem in the family of problems

$$-u''+\tilde{p}u=-\varepsilon(p-\tilde{p})u, \tag{2.1}$$

parametrized by ε. The problem we want to solve is for the case $\varepsilon=1$. We know the solution for $\varepsilon=0$ and $\tilde{p}$ a low degree piecewise polynomial. This is the zeroth order problem studied in [2], [9], [12], [14]–[15], [17]–[18], [23], [27]–[28]. From the theory of differential equations, we expect the solution to (2.1) to be analytic in ε in some disk about the origin. We will thus seek a power series solution in ε and hope that the disk convergence of this series includes $\varepsilon=1$. This will be true if $(p-\tilde{p})$ is sufficiently small and sufficiently smooth.

We define

$$u(r|\varepsilon)=\sum_{j=0}^{\infty}\varepsilon^j v_j(r), \tag{2.2}$$

where we employ the convention that $v_j(r)$ will be referred to as the jth order perturbation correction. The procedure we will follow will be to substitute the expression for $u(r|\varepsilon)$ and its second derivative into (2.1) and then to equate like powers of ε. If we then apply the boundary conditions of (1.5) to v_0 and corresponding homogeneous boundary conditions ($c_3=c_6=0$) to v_j, $j=1,2,\cdots$ we have the following two-point boundary value problems:

$$\begin{aligned}&-v_0''+\tilde{p}v_0=0, &&\alpha<r<\beta,\\ &c_1v_0(\alpha)+c_2v_0'(\alpha)=c_3, &&|c_1|+|c_2|\neq 0,\\ &c_4v_0(\beta)+c_5v_0'(\beta)=c_6, &&|c_4|+|c_5|\neq 0,\end{aligned} \tag{2.3}$$

where $v_0(R_k^+)=v_0(R_k^-)$ and $v_0'(R_k^+)=v_0'(R_k^-)$ for $k=1,2,\cdots,M$,

$$\begin{aligned}&-v_j''+\tilde{p}v_j=-(p-\tilde{p})v_{j-1}, &&\alpha<r<\beta,\\ &c_1v_j(\alpha)+c_2v_j'(\alpha)=0, &&|c_1|+|c_2|\neq 0,\\ &c_4v_j(\beta)+c_5v_j'(\beta)=0, &&|c_4|+|c_5|\neq 0,\end{aligned} \tag{2.4}$$

where $v_j(R_k^+)=v_j(R_k^-)$ and $v_j'(R_k^+)=v_j'(R_k^-)$ for $k=1,2,\cdots,M$ and $j=1,2,\cdots$. As usual, we terminate the representation in (2.2) at some finite upper limit n, and seek a perturbation series solution to (1.5) up through and including the first n perturbation corrections.

In each subinterval the general solution of (2.3) can be expressed as a linear combination of two linearly independent basis functions

$$v_0(r)=\alpha_0v_0^+(r)+\beta_0v_0^-(r), \tag{2.5}$$

for some constants α_0 and β_0. The general solution to (2.4) is composed of a homogeneous solution $v_{h,j}$ and a particular solution $v_{p,j}$, $j=1,2,\cdots,n$. Since the homogeneous solution is expressed as a linear combination of the two zeroth order basis functions, v_0^+ and v_0^-, we have

$$v_j=\alpha_jv_0^++\beta_jv_0^-+v_{p,j}, \tag{2.6}$$

for constants α_j and β_j, $j = 1, 2, \cdots, n$. If we assume that the mesh is given, the two constants parameterizing the solution of each perturbation correction in each subinterval can be determined by forming and solving a $2M \times 2M$ block bidiagonal system of linear equations relating the α_j's and β_j's in one subinterval to those in adjacent subintervals. This essentially reflects the fact that we require each perturbation correction v_j, $j = 0, 1, \cdots, n$, to be C^1 continuous at each mesh node.

In determining the zeroth order solution to (1.5) we saw that in each subinterval we could represent the solution as a linear combination of two linearly independent basis functions v_0^+ and v_0^-. However, once we allow the first perturbation correction to be included in the approximate solution, it becomes unclear as to how to represent the solution to (1.5) in a similar way. However, to try to do so is appealing, not only from a mathematical viewpoint but from a conceptual one as well.

To achieve this type of representation we re-examine the general solution of (1.5). The two linearly independent basis functions, $u^+(r)$ and $u^-(r)$ have the property that

$$-u''^{\pm} + pu^{\pm} = 0. \tag{2.7}$$

Our strategy will be to rewrite (2.7) as

$$-u''^{\pm} + \tilde{p}u^{\pm} = -(p - \tilde{p})u^{\pm}, \tag{2.8}$$

and seek a perturbation series solution to both u^+ and u^-. Hence we define

$$u^{\pm}(r) = \sum_{j=0}^{\infty} v_j^{\pm}(r). \tag{2.9}$$

As before, we will terminate the expansions in (2.9) at some finite upper limit n. These expressions will be referred to as modified basis functions. Note that we have not written (2.8) and (2.9) in terms of the formal expansion parameter ε because we are interested in the case $\varepsilon = 1$ and the expansion parameter merely serves as a mnemonic aid in deriving the perturbation equations.

If we substitute the expressions for u^+ and u^- and their second derivatives into (2.8) and then order the equations much the same as was done in (2.3) and (2.4), we have the following system of ordinary differential equations:

$$(2.10)\quad \begin{array}{ll} \text{(a)} & -v_0''^{\pm} + \tilde{p}v_0^{\pm} = 0, \\ \text{(b)} & -v_1''^{\pm} + \tilde{p}v_1^{\pm} = -(p - \tilde{p})v_0^{\pm}, \\ & \vdots \\ \text{(c)} & -v_n''^{\pm} + \tilde{p}v_n^{\pm} = -(p - \tilde{p})v_{n-1}^{\pm}. \end{array}$$

Note that by splitting the solution of (1.5) into two linearly independent basis functions and by then seeking a perturbation series approximation to each of them, we can no longer explicitly write out a two-point boundary value problem with known boundary conditions for $v_j^{\pm}$, $j = 0, 1, 2, \cdots, n$. However, by systematically solving for $v_0^{\pm}, v_1^{\pm}, \cdots, v_n^{\pm}$ we can express our perturbation solution in each subinterval as a linear combination of the two modified basis functions. We have

$$\bar{u}(r) = \alpha \sum_{j=0}^{n} v_j^+ + \beta \sum_{j=0}^{n} v_j^-, \tag{2.11}$$

for constants α and β. If we again assume that the mesh $\mathcal{M}$ is given, we can determine the two constants parameterizing our solution in each subinterval by requiring that the approximate solution be C^1 continuous at each mesh node. This amounts to again forming and solving a $2M \times 2M$ block bidiagonal system of linear equations relating α and β in one subinterval to those in the adjacent subintervals.

At first glance there may seem to be little advantage in implementing the modified basis function approach as opposed to the perturbation method of (2.2). However, as we shall see shortly, when the mesh $\mathcal{M}$ is not known, the modified basis function approach will allow us to adaptively determine the mesh. The perturbation approach of (2.2) does not conveniently allow this.

As was the case for each $v_j, j = 1, 2, \cdots, n$, the general solution to each $v_j^{\pm}$ in each subinterval is composed of a homogeneous solution and a particular solution. The homogeneous solution is parameterized by two constants. However, because of the way we have chosen to represent our perturbation solution, we have the freedom to choose these two constants in some meaningful way. Although there is a variety of ways in which one can determine them, we have employed a least squares minimization. We determine the homogeneous constants such that the L^2 norm of the given perturbation correction is minimized. This implies that each $v_j^{\pm}, j = 1, 2, \cdots, n$, will lie in the orthogonal complement of the nullspace of (2.3). We can think of this minimization procedure as a device to keep the perturbation corrections as close to the canonical zeroth order set (v_0^+ and v_0^-) as possible. If we denote the general solution to $v_j^{\pm}$ on each subinterval by

$$v_j^{\pm} = c_1^{\pm} v_0^+ + c_2^{\pm} v_0^- + v_{p,j}^{\pm}, \qquad j = 1, 2, \cdots, n, \tag{2.12}$$

for some constants $c_1^{\pm}$ and $c_2^{\pm}$, where $v_{p,j}^{\pm}$ is the particular solution, then minimization of the L^2 norm of each $v_j^{\pm}$ amounts to solving the system of equations

$$\begin{bmatrix} \int_{R_{k-1}}^{R_k} (v_0^+)^2\,dr & \int_{R_{k-1}}^{R_k} (v_0^-)(v_0^+)\,dr \\ \int_{R_{k-1}}^{R_k} (v_0^-)(v_0^+)\,dr & \int_{R_{k-1}}^{R_k} (v_0^-)^2\,dr \end{bmatrix} \begin{bmatrix} c_1^{\pm} \\ c_2^{\pm} \end{bmatrix} = \begin{bmatrix} -\int_{R_{k-1}}^{R_k} (v_{p,j}^{\pm})(v_0^+)\,dr \\ -\int_{R_{k-1}}^{R_k} (v_{p,j}^{\pm})(v_0^-)\,dr \end{bmatrix}. \tag{2.13}$$

Ordinarily when we solve (1.5) by a piecewise perturbation series, we would like the local error or the global error incurred by such a solution method to be less than some pre-set error tolerance. If, for example, the global error we incur is larger or smaller than our error tolerance, we want to appropriately increase or decrease the size of the mesh intervals. Note however, that even though we are applying the perturbation scheme of (2.2) locally—in each subinterval—each time we change the size of the mesh intervals we must repeat the determination of each perturbation correction on the whole interval $[\alpha, \beta]$. Hence we have a local perturbation series solution with a global mesh adjustment procedure. The modified basis function approach also produces a local perturbation series solution. However, since the entire perturbation series solution in each subinterval is parameterized by two constants, one can apply a variation of Gordon's [14] shooting method with local error estimates to obtain a local mesh adjustment procedure. This substantially increases the efficiency of our piecewise perturbation series method. If the error incurred in a given subinterval is too large or too small we need only repeat the calculation in that subinterval.

We now focus our attention on the method that is used in determining our perturbation corrections analytically. For a differential equation with an arbitrary inhomogeneous term, one can obtain a particular solution by the method of variation of parameters [8] or Green's function techniques [10]. However, if the coefficients of the differential equation are constants, and if the inhomogeneous term is some polynomial–exponential combination, then an analytical solution can be obtained by the method of undetermined coefficients.

Note that if $\tilde{p}(r)$ is taken to be a piecewise constant approximation to $p(r)$—for example the value of $p(r)$ at the midpoint of each subinterval—then $v_0^{\pm}$ are exponentials,

$$v_0^+ = e^{\sqrt{\tilde{p}}r}, \qquad v_0^- = e^{-\sqrt{\tilde{p}}r}. \tag{2.14}$$

Hence, if in each subinterval we represent $(p - \tilde{p})$ in a Taylor series expanded about the midpoint of the interval and terminated at some finite upper limit, we will be able to determine an analytical particular solution for $v_j^{\pm}$, $j = 1, 2, \cdots, n$. Recall that if we want to obtain a particular solution of the second order inhomogeneous ordinary differential equation

$$y'' - \lambda^2 y = \left(\sum_{i=0}^{s} b_i r^i\right) e^{\pm\lambda r}, \tag{2.15}$$

$\lambda \neq 0$, by the method of undetermined coefficients, then we look for a particular solution of the form

$$y_p(r) = \left(\sum_{i=0}^{s} a_i r^{i+1}\right) e^{\pm\lambda r}. \tag{2.16}$$

If we substitute (2.16) and its second derivative into (2.15), factor out the common exponentials and then equate like powers or r, we have

$$\begin{aligned} &\text{(a)} \quad a_s = \pm\frac{b_s}{2\lambda(s+1)}, \\ &\text{(b)} \quad a_i = \pm\frac{b_i - (i+2)(i+1)a_{i+1}}{2\lambda(i+1)}, \qquad i = 0, 1, \cdots, s-1. \end{aligned} \tag{2.17}$$

The general solution of (1.21) can then be expressed as

$$y(r) = c_1 e^{\lambda r} + c_2 e^{-\lambda r} + y_p(r), \tag{2.18}$$

for constants c_1 and c_2 and y_p of the form given by (2.16) and (2.17). As a result, we see that if in each subinterval $(p - \tilde{p})$ is expressed as a Taylor series expanded about the midpoint of the interval, then the right-hand side of (2.10b) is a polynomial–exponential combination. Hence, a general solution to v_1^+ or v_1^- will be similar in form to (2.18). The least squares minimization described in (2.12)–(2.13) can then be applied to determine the constants parameterizing the homogeneous solution of $v_1^{\pm}$. This procedure can be continued in determining the $v_j^{\pm}$, $j = 2, 3, \cdots, n$.

An important point to note is that in the determination of a particular solution of (2.10b, c) by the method of undetermined coefficients if the right hand side is of one (or a mixed) exponential type, then so is the particular solution. The "full" least squares minimization procedure defined in (2.13) has the effect that particular solutions containing more than first order perturbation corrections will always be of mixed exponential type. This complicates the calculation and the representation of the modified basis functions. An alternative to this full least squares minimization is to always force the homogeneous solution to be of the same exponential type as the particular solution. Hence for the "plus" perturbation corrections in (2.12) we can set $c_2^+ = 0$ and perform a minimization with respect to c_1^+. Similarly, for "minus" perturbation corrections we can set $c_1^- = 0$ and perform a minimization with respect to c_2^-. Thus each of our modified basis functions will be of a given exponential type—that is, only $e^{\sqrt{\bar{p}}r}$ in the plus modified basis function and only $e^{-\sqrt{\bar{p}}r}$ in the minus modified basis function. Maintaining a homogeneity of given exponential type in the plus and minus modified basis functions is appealing from a mathematical as well as from a conceptual viewpoint. Such a "partial" least squares minimization will mean that the L^2 norm of each of our perturbation corrections will not be as small as if we had employed a full least squares minimization. However, we would hope that the coefficient of the opposite exponential in the homogeneous solution would be small anyway, so that the loss would be small.

Although the partial least squares minimization causes a slight reduction in the convergence rate of our perturbation solution, the computational gains from such a procedure make its implementation well worthwhile. As Anderson remarks, it is not really the convergence rate of each of our modified basis functions separately that matters, but rather the ability of linear combinations of modified basis functions to approximate solutions to the original equation [3].

Finally, we note that there is nothing sacred about our perturbation parameterization. In problems of physical interest, the potential $p(r)$ often contains a natural expansion parameter. Hence, if we expand $u^\pm$ as well as $(p-\tilde{p})$ in a series and then follow arguments similar to those which led to (2.10), we find

$$\begin{aligned} &-v_0^{\pm\prime\prime}+\tilde{p}v_0^\pm=0,\\ &-v_1^{\pm\prime\prime}+\tilde{p}v_1^\pm=-(p-\tilde{p})_0v_0^\pm,\\ &-v_2^{\pm\prime\prime}+\tilde{p}v_2^\pm=-(p-\tilde{p})_0v_1^\pm-(p-\tilde{p})_1v_0^\pm,\\ &\quad\vdots \end{aligned} \tag{2.19}$$

where the quantities $(p-\tilde{p})_l$ can simply be the corresponding terms in a local Taylor series respresentation of $(p-\tilde{p})$ expanded about the midpoint of the interval in question. Since $(p-\tilde{p})$ is expanded in a power series, the degree of the polynomial approximation to u^+ and u^- will increase from stage to stage in proportion to the degree of the approximation to $(p-\tilde{p})$. However, in order to minimize the number of terms to be manipulated, it is advantageous to curb the growth of these degrees. The parameterization which led to (2.19) achieves this result. We bring in higher order terms in $(p-\tilde{p})$ only as we bring in higher order terms in the perturbation expansion.

3. Numerical implementation of the piecewise analytical perturbation series method. In actually solving scattering and bound state problems we require a numerical algorithm which enables us to implement the piecewise analytical perturbation series method described at the end of the last section. Recall that by choosing a mesh

$$\mathcal{M}=\{0<\alpha=R_0<R_1<\cdots<R_M=\beta<\infty\}, \tag{3.1}$$

where we set $h_k=R_k-R_{k-1}$, $k=1,2,\cdots,M$, we can reduce the scattering and bound state problems to a finite domain by imposing boundary conditions at R_0 and R_M. In practice the numerical algorithms for solving both problems will allow us to adjust the size of each mesh interval in order to meet some pre-set error criterion. We reformulate the scattering boundary value problem as an initial value problem on $[\alpha,\beta]$ and use a shooting method, specifically the method of complementary functions [20], to integrate the wavefunction and its derivative from α to β. Such a shooting method requires that we specify initial conditions for the wavefunction and its derivative at α. The assumed initial conditions at α and the resulting solution at β will ultimately be adjusted to satisfy the asymptotic form of the wavefunction $u_\infty(r)$. In the bound state problem we specify initial conditions at both α and β and integrate the wavefunction and its derivative from α to some interior point—say R_{MID}—and from β to R_{MID}. Since we are interested in problems in which the effective intermolecular potential, $p(r)+E$, has a local minimum which is negative, we typically take R_{MID} to be the position of the bottom of the well. There will be a nontrivial solution for only particular energy values and it will be characterized by the fact that the interface conditions at R_{MID} are satisfied. We use a root finding technique to adjust the energy so that the interface conditions are satisfied.

3.1. Propagation. We begin our discussion of the numerical implementation of PAPSM by considering the propagation of our algorithm. Omitting consideration of the regions $[0, R_0]$ and $[R_M, \infty]$, we will discuss the shooting method across a general interior subinterval $[R_{k-1}, R_k]$, $k=1,2,\cdots,M$. Introducing a reduced independent variable x, where

$$r=\frac{R_k+R_{k-1}}{2}+\left(\frac{R_k-R_{k-1}}{2}\right)x, \tag{3.2}$$

we can define a corresponding reduced dependent variable and coefficient such that the radial Schrödinger equation can be written as

$$-w''(x)+q(x)w(x)=0, \qquad -1<x<1. \tag{3.3}$$

If we approximate $q(x)$ by a piecewise constant polynomial approximation $\tilde{q}(x)$ obtained by evaluating $q(x)$ at the midpoint of the interval in question, then by rewriting (3.3) in the form

$$-w''+\tilde{q}w=-(q-\tilde{q})w \tag{3.4}$$

we can apply our modified basis function perturbation algorithm to generate two modified basis functions w^{+n} and w^{-n}. We denote the general approximate solution to (3.4) in the kth subinterval by

$$\bar{w}_k^n=\alpha^n w^{+n}+\beta^n w^{-n}, \tag{3.5}$$

for constants α^n and β^n. If we use the initial conditions that arise from the continuity conditions of the wavefunction and its derivative at the interface point between the $(k-1)$st and kth subinterval, we can determine α^n and β^n by solving two linear equations. Once α^n and β^n are determined, we can evaluate $\bar{w}_k^n(1)$ and $\bar{w}_k^{n\prime}(1)$. With the propagation complete across the kth subinterval, we must check to see if the mesh length, h_k, has to be adjusted to satisfy our given local error tolerance.

If a perturbation series is convergent, we would hope that the first neglected term be small compared to the expansion we are considering. In our problems, we define an error estimate, ε_k^n, associated with our perturbation solution in the kth subinterval. We set

$$\varepsilon_k^n(x)=\frac{|\bar{w}_k^{n+1}(x)-\bar{w}_k^n(x)|+L_k|\bar{w}_k^{n+1\prime}(x)-\bar{w}_k^{n\prime}(x)|}{|\bar{w}_k^n(x)|+L_k|\bar{w}_k^{n\prime}(x)|}. \tag{3.6}$$

In classically allowed regions, $(p(r)<0)$, our wavefunction will be oscillatory. Hence there is the possibility that it may vanish at one or more points in the interval. As a result, we have included derivative terms in (3.6) to avoid the possibility of ε_k^n becoming singular at these points. L_k is a local characteristic distance over which the wavefunction varies. We follow Gordon [14] and choose L_k such that for a "typical" wavefunction

$$|\bar{w}_k^n| \sim L_k |\bar{w}_k^{n\prime}|. \tag{3.7}$$

He uses the estimate

$$L_k = \left(\tilde{q} + \left|\frac{dq}{dx}\right|_{x=0}^{2/3}\right)^{-1/2}. \tag{3.8}$$

In order to use the estimate in (3.6) to adjust the size of the mesh to satisfy a pre-set error tolerance, we must know how ε_k^n varies with interval length h_k. From [31], we expect our perturbation scheme to generate global error bounds $O(h^{n+1})$ where $h = \max_k h_k$. We expect a local error bound to be one power of h higher.

In our shooting method we are interested in the error incurred by our perturbation scheme at the right-hand side of an interval. Hence we evaluate $\varepsilon_k^n(1)$. If the error is too large, we want to shrink the mesh length and repeat the calculation. If the error is too small, we want to increase the mesh length and repeat the calculation.

If we let TOL be our pre-set error tolerance, then we can adjust the mesh interval by using the formula

$$h_{k,\text{NEW}} = \left[\frac{\text{TOL}}{\varepsilon_k^n(1)}\right]^{1/(n+2)} h_{k,\text{OLD}}. \tag{3.9}$$

Ordinarily we cannot determine the exact h_k that will produce a local error equal to TOL. However, we can avoid this difficulty by requiring the local error in each subinterval to be less than some maximum value TOL1 and greater than some minimum value TOL2

$$\text{TOL2} < \text{TOL} < \text{TOL1}. \tag{3.10}$$

Once we satisfy our error criterion, we normalize the wavefunction and its derivative by dividing them by the quantity

$$N_k = |\bar{w}_k^n(1)| + |L_k \bar{w}_k^{n\prime}(1)|. \tag{3.11}$$

Upon normalization, the values $\bar{w}_k^n(1)/N_k$ and $\bar{w}_k^{n\prime}(1)/N_k$ become the initial data for the $(k+1)$st subinterval. We start the propagation over with $h_{k+1} = h_k$.

3.2. Initialization and termination. Since $p(r) \to \infty$ as $r \to 0$, in problems of interest, we know that $u(r) \to 0$ exponentially rapidly as $r \to 0$. The boundary conditions at small r for the scattering problem require that $u(r) = 0$ at $r = 0$. However, in practice, $u(r)$ becomes so tiny at some small but nonzero r, call it R_0, that one can simply set the wavefunction to zero at R_0 [5]. This amounts to assuming the potential is an infinitely repulsive wall inside R_0. Gordon [14] has employed a linear reference potential to improve the treatment of the boundary conditions at small r for the scattering problem. Luthey [23] and Anderson [3] have employed a singular potential method. Although the singular potential method provides the best treatment of the boundary conditions at small r, we have initialized our scattering problem by employing a single linear segment from R_0 to R_1 which can be extrapolated back to the origin. Most of our numerical experiments have been designed to compare the efficiency of Gordon's [14] widely used offset tangent method and our piecewise analytical perturbation series method. Gordon's method was chosen, as opposed to another piecewise analytical solution method, due to the availability of a large amount of test data. In order to help us make a fair comparison between the two methods, we have implemented Gordon's Airy function initialization routine.

In the region of small r, a linear reference potential produces two linearly independent basis functions—the regular and irregular Airy functions. The regular Airy function tends to zero as its argument becomes large and positive or equivalently as $r \to 0$. The irregular Airy function diverges as its argument becomes large and positive or equivalently as $r \to 0$. Since we want $u(r) \to 0$ as $r \to 0$, we take the wavefunction proportional to the regular Airy function in $[0, R_0]$. Specifically, if we represent our linear reference potential as the first two terms of the Taylor series representation of $q(x)$ expanded about the midpoint of the first interval, then upon introducing the change of variables

$$\xi = \operatorname{sgn} a|a|^{1/3}x + a^{-2/3}b, \tag{3.12}$$

where

$$\text{(3.13)} \quad \begin{aligned} &\text{(a)} & a &= q'(0), \\ &\text{(b)} & b &= q(0), \end{aligned}$$

(3.3a, b) can be rewritten as

$$-y''(\xi) + \xi y(\xi) = 0, \tag{3.14}$$

where $y(\xi) = w(\xi(x))$. The general solution of (3.14) is a linear combination of the regular and irregular Airy functions, Ai (ξ) and Bi (ξ), respectively. As stated above, we write

$$u(R_0) = \alpha \operatorname{Ai}(\xi(-1)), \tag{3.15}$$

for some constant α. Upon normalizing $u(R_0)$ to some convenient value, say .5, we can write [30] the two initial conditions for our shooting method as

$$\text{(3.16)} \quad \begin{aligned} u(R_0) &= .5, \\ u'(R_0) &= .5\left(\frac{2}{R_1 - R_0}\right) \operatorname{sgn} a|a|^{1/3} \frac{\operatorname{Ai}'(\xi(-1))}{\operatorname{Ai}(\xi(-1))}. \end{aligned}$$

In order to terminate our scattering algorithm we must determine the asymptotic form of $u(r)$ in $(R_M, \infty]$. We assume that the interaction potential, $u(r)$, vanishes more rapidly than r^{-2} as $r \to \infty$. Hence we have

$$p(r) \to \frac{l(l+1)}{r^2} - E \quad \text{as } r \to \infty, \tag{3.17}$$

where l is the specified orbital angular momentum quantum number. In the scattering problem E is greater than zero and we associate this with asymptotically open or physical scattering states. Equation (3.17) then implies that the radial Schrödinger equation approaches

$$-u''(r) + \left(\frac{l(l+1)}{r^2} - k^2\right) u(r) = 0, \tag{3.18}$$

where $k^2 = E$. This is the Riccati–Bessel equation whose real and linearly independent solutions can be written in terms of the regular and irregular spherical Bessel functions, $j_l(kr)$ and $y_l(kr)$ respectively [1]. We write these two linearly independent solutions as

$$\mathcal{J}_l(kr) = krj_l(kr) \quad \text{and} \quad \mathcal{Y}_l(kr) = kry_l(kr). \tag{3.19}$$

$u(r)$ then asymptotes to

$$u_\infty(r) = \mathcal{J}_l(kr) - \mathcal{R}\mathcal{Y}_l(kr), \tag{3.20}$$

where the reactance, $\mathcal{R}$, (or more precisely quantities derived therefrom) is the quantity of fundamental interest.

Our shooting method propagates an approximate solution with increasing r until we reach the Mth subinterval. We generally estimate the asymptotic region to be such that R_M is greater than some given value of r. Assuming we have satisfied our pre-set local error tolerance and we have normalized our solution we denote the value of the wavefunction and its derivative at the right-hand side of the Mth subinterval by n_1 and n_2 respectively. We can determine the constants α and β parameterizing our asymptotic solution in the $(M+1)$st subinterval by solving

$$(3.21)\quad \begin{array}{ll} \text{(a)} & n_1 = \alpha\mathcal{J}_l(R_M) + \beta\mathcal{Y}_l(R_M), \\ \text{(b)} & n_2 = \alpha\mathcal{J}_l'(R_M) + \beta\mathcal{Y}_l'(R_M). \end{array}$$

A comparison between (3.20) and (3.21a) reveals that

$$\mathcal{R} = -\frac{\beta}{\alpha}. \tag{3.22}$$

We then apply our perturbation algorithm on the $(M+1)$st subinterval. We determine the constants parameterizing our modified basis function perturbation solution in the $(M+1)$st subinterval and, by denoting the value of the wavefunction and its derivative at the right-hand side of the $(M+1)$st subinterval by $\bar{n}_1$ and $\bar{n}_2$ respectively, we can determine the constants α and β parameterizing our asymptotic solution in the $(M+2)$nd subinterval by solving a system of equations similar to (3.21). We repeat the calculation of $\mathcal{R}$ on the $(M+2)$nd subinterval and continue this process over successive intervals of uniform length h_M until the variations in $\mathcal{R}$ are less than an allowed tolerance. Once convergence has been verified, the phase shift, δ, is evaluated using the relations in (3.23) and the calculations terminated. We have

$$(3.23)\quad \begin{gathered} S = S_R + iS_I, \quad S_I = \frac{2\mathcal{R}}{1+\mathcal{R}^2}, \quad S_R = 1 - \mathcal{R}S_I, \\ \delta = \frac{1}{2}\tan^{-1}\left(\frac{S_I}{S_R}\right). \end{gathered}$$

Initialization of our bound state algorithm for small r is precisely the same as that for the scattering problem. However, since we are not only propagating our solution from R_0 to R_{MID}, but from R_M to R_{MID} as well, we must also initialize our algorithm in the region of large r.

As in the case of the scattering problem, we assume that the interaction potential, $u(r)$, vanishes more rapidly than r^{-2} as $r \to \infty$. Hence we have

$$p(r) \to \frac{l(l+1)}{r^2} - E \quad \text{as } r \to \infty. \tag{3.24}$$

The radial Schrödinger equation still approaches (3.18); however, since k^2 is now negative, the two linearly independent solutions S_1 and S_2 can be written in terms of the two modified spherical Bessel functions i_l and i_{-l}. We have

$$(3.25)\quad \begin{array}{ll} \text{(a)} & S_1 = |k| r i_l(|k| r), \\ \text{(b)} & S_2 = |k| r i_{-l}(|k| r). \end{array}$$

If we take appropriate linear combinations of (3.25a, b), we can obtain two linearly independent solutions $I_l^+(r)$ and $I_l^-(r)$. $I_l^-(r)$ decays to zero as $r \to \infty$ and $I_l^+(r)$ diverges as $r \to \infty$. Since we want $u(r) \to 0$ as $r \to \infty$, we have

$$u(r) = \beta I_l^-(r) \quad \text{in } [R_M, \infty), \tag{3.26}$$

for some constant β. If we normalize $u(R_M)$ to some convenient value, say .5, then we have

$$u(R_M) = .5, \qquad u'(R_M) = .5\,\frac{I_l^{-\prime}(R_M)}{I_l^-(R_M)}. \tag{3.27}$$

To determine the eigenvalues (bound states) we must find the energies, E, such that the interface conditions at R_{MID} are satisfied. We begin our propagation at R_0 with the initial conditions in (3.16) and propagate our solution until we reach R_{MID}. If we denote the normalized solution propagating from the left by $\bar{u}_L(r)$, then if the energy used is an eigenvalue, there exists a scalar, v_L, such that the eigenfunction $u(r) = \bar{u}_L(r)v_L$. In addition, we start the propagation at R_M with the initial conditions in (3.27) and propagate our solution until we reach R_{MID}. If we denote the normalized solution propagating from the right by $\bar{u}_R(r)$, then if the energy used is an eigenvalue, there exists a scalar, v_R, such that the eigenfunction $u(r) = \bar{u}_R(r)v_R$. At an eigenvalue and only at an eigenvalue it must be possible to join smoothly at R_{MID} the solutions from the left- and right-hand sides. Our two interface conditions become

(3.28) (a) $\bar{U}v = 0,$

where

(b) $$\bar{U} = \begin{bmatrix} \bar{u}_L(R_{\text{MID}}) & \bar{u}_R(R_{\text{MID}}) \\ \bar{u}'_L(R_{\text{MID}}) & \bar{u}'_R(R_{\text{MID}}) \end{bmatrix} \quad \text{and} \quad v = \begin{bmatrix} v_L \\ -v_R \end{bmatrix}.$$

The system of equations in (3.28a) has a nontrivial solution if the determinant of the 2×2 coefficient matrix $\bar{U}$ vanishes. The determinant of $\bar{U}$ will also be a continuous function of the energy E. Hence we can relate our eigenvalue problem to a nonlinear root finding problem where we must find the zeros of det $\bar{U}$. In practice we perform a one-dimensional search in the energy to find the region where det $\bar{U}$ changes sign. Once we have bracketed an energy range containing a zero of det $\bar{U}$, we employ a direct quadratic interpolation root finding technique [3] to find the eigenvalue.

3.3. Potential difficulties. There are two potential difficulties associated with our piecewise analytical perturbation series method. The first involves the fact that as $\bar{q}(x)$ gets small in one or more intervals, the two modified basis functions may become numerically linearly dependent. The second difficulty is associated with the fact that in regions around turning points—points where $q(x) = 0$—we have observed our perturbation series to be slowly converging or diverging. It is instructive at this time to briefly discuss these problems together with ways of mitigating their effects.

Consider the case where $\bar{q}(x)$ is small and positive. We would expect the dominant terms of the modified basis functions to be $a_0e^{\sqrt{\bar{q}}x}$ and $b_0e^{-\sqrt{\bar{q}}x}$, where a_0 and b_0 are constants. However, for small $q(x)$ these terms tend to become multiples of each other when evaluated numerically. To avoid this numerical linear dependence we consider the linear combinations

$$(3.29) \qquad \frac{w^{+n}}{2a_0}+\frac{w^{-n}}{2b_0} \quad \text{and} \quad \frac{w^{+n}}{2\sqrt{\tilde{q}}a_0}-\frac{w^{-n}}{2\sqrt{\tilde{q}}b_0},$$

where $w^{\pm n}$ are the two modified basis functions. The linear combinations in (3.29) produce two new modified basis functions whose leading terms are $\cosh(\sqrt{\tilde{q}}x)$ and $\sinh(\sqrt{\tilde{q}}x)/\sqrt{\tilde{q}}$ respectively. As $\tilde{q}(x)\to 0$, the leading terms in the perturbation series become 1 and x respectively. For the higher order terms in the modified basis functions we eliminate the need to evaluate $e^{\sqrt{\tilde{q}}x}$ and $e^{-\sqrt{\tilde{q}}x}$ by converting them to hyperbolic sines and cosines. For even smaller values of $\tilde{q}$, we have found it helpful to evaluate $\cosh(\sqrt{\tilde{q}}x)$ and $\sinh(\sqrt{\tilde{q}}x)/\sqrt{\tilde{q}}$ by taking the first few terms of their Taylor series expansion about $x=0$. This has the added feature of allowing us to remove the numerical singularity of $\sinh(\sqrt{\tilde{q}}x)/\sqrt{\tilde{q}}$ as $\tilde{q}\to 0$. A similar procedure can be applied to the case where $\tilde{q}(x)$ is small and negative; see [30]. In this case the new set of modified basis functions will be in terms of sines and cosines.

Typically we use the exponential representation for the two modified basis functions when $\tilde{q}\geqq 1.0$. We use the sinh–cosh or sin–cos representation when $.1\leqq\tilde{q}<1.0$. However, for $\tilde{q}\ll .1$ we find that, even though we are employing linear combinations of modified basis functions as in (3.29), our perturbation series converges slowly or diverges. This leads us to our second problem.

The difficulty in using the perturbation series formalism in turning point regions stems from the fact that as $\tilde{p}\to 0$, the process of evaluating the coefficients comprising the particular solutions of the equations in (2.19) using recurrence relations similar to those in (2.17) is unstable numerically. One way to mitigate the effect of turning point regions is to employ the Airy function method of Gordon [14]. We recall that no special problems occur in the vicinity of turning points for this method. As a result we have employed a zeroth order piecewise analytical solution method in these regions based upon a piecewise linear potential approximation to $p(r)$. Other ways to mitigate the effect of turning point regions are discussed in some detail in [30].

4. Numerical results and discussion. A variety of numerical experiments were performed to evaluate the accuracy and efficiency of our piecewise analytical perturbation series method. The accuracy of the method is related to the local order of the error expression in (3.6). Recall that if a quantity τ has a functional dependence on h_k, then to say that $\tau=O(h_k^p)$ implies that $\lim_{h_k\to 0} h_k^{-p}\tau=c$, for some constant c. Hence, we perform tests which determine the local order of ε_k^n for an approximation to u containing a given number of perturbation corrections.

The evaluation of the efficiency of the method is much more difficult. Implicit in such an evaluation is a comparison of our method with other proposed methods of solution. The comparison should be based upon the number of subintervals needed and the cost per subinterval. Although we cannot adequately compare our method with all proposed methods, we will make comparisons between ours and Gordon's [15] offset tangent piecewise analytical solution method. If we then incorporate the results of Riehl, Diestler, and Wagner [27] in their comparison of the Numerov method and their piecewise analytical solution method with our results, we will have a better understanding of the relative merits of PAPSM, PASM and the Numerov method.

For our test calculations we have employed the hypothetical $A-B_2$ atom–diatomic molecule system first introduced by Lester and Bernstein [22]. This model system serves as a standard test problem from which we can compare our results with those of others. In this one-dimensional system, the coefficient function $p(r)$ is given by

$$p(r) = -E + \frac{l(l+1)}{r^2} + v(r), \tag{4.1}$$

where E is the energy, l is the orbital angular momentum quantum number and $v(r)$ is an interaction potential. For our system we have taken $v(r)$ as a Lennard-Jones 12–6 potential such that

$$v(r) = \lambda^2\left(\frac{1}{r^{12}} - \frac{2}{r^6}\right), \tag{4.2}$$

where λ^2, a scalar, is set equal to 1000.0. In Fig. 1 we plot the effective intermolecular potential

$$v^*(r) = \frac{l(l+1)}{r^2} + v(r), \tag{4.3}$$

as a function of radial distance r for a variety of orbital angular momentum quantum numbers.

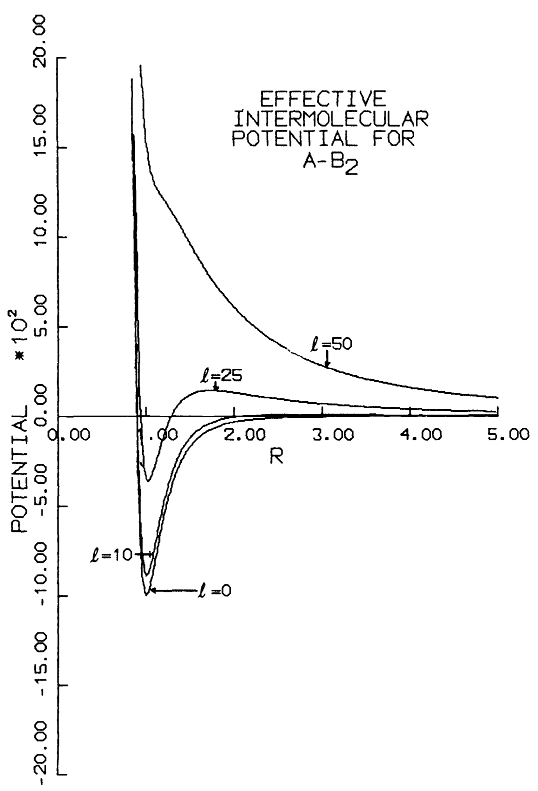


FIG. 1. *Effective intermolecular potential for model $A - B_2$ system.*

All of the numerical results in this paper were obtained using a DEC PDP-10 computer at Harvard University's Aiken Computation Laboratory. The computer codes were written in single precision FORTRAN.

4.1. Order of the method. As we mentioned previously, we expect the error estimate in (3.6) to be proportional to some power of h_k. If we evaluate $\varepsilon_k^n(x)$ at a set of equispaced points in $[R_{k-1}, R_k]$, average the results and then repeat the calculation several times for a new h_k equal to one half the old value, we can obtain average local error estimates. If we then plot the local error estimates versus corresponding mesh interval lengths, h_k, on log–log paper, we can obtain the power of h_k to which our local error estimate is proportional.

We have performed such experiments for the perturbation scheme defined in (2.19) using a variety of energies, angular momentum quantum numbers and starting points. From the results contained in [31], we expect

$$\|\varepsilon_k^n\|_\infty = \operatorname*{ess\,sup}_{R_{k-1} \leqq r \leqq R_k} |\varepsilon_k^n| = O(h_k^{n+2}). \tag{4.4}$$

All of the numerical experiments we have performed indicate that the $O(h_k^{n+2})$ local error dependence is obeyed. In Table 1 we have recorded the theoretical and average calculated orders of our local error estimate for a variety of approximate solutions containing as many as three perturbation corrections. A total of five interior equispaced points were used in obtaining an average value for $\varepsilon_k^n(x)$ for each value of h_k.

TABLE 1
Local order: $O(h_k^\alpha)$.

n	Theoretical	Calculated
0	h_k^2	$h_k^{2.00}$
1	h_k^3	$h_k^{2.81}$
2	h_k^4	$h_k^{3.93}$
3	h_k^5	$h_k^{5.13}$

We observe that as we continue to add perturbation corrections to our expansion, the local order of our method essentially increases by one power of h_k. As we shall see, such an increase in the order of the method results in a reduction in the number of subintervals needed to solve our problems.

4.2. Efficiency of the method. The numerical methods that are used to solve linear two-point boundary value problems can be classified as either direct methods or as shooting methods. Direct methods, as the name implies, are applied directly to the boundary value problem. Shooting methods on the other hand are based upon the equivalence existing between the boundary value problem and a corresponding initial value problem. As it turns out, shooting methods are a more effective class of numerical techniques than direct methods for solving the radial Schrödinger equation.

In the past, several authors [7], [13] have noted that step by step integration of the one-dimensional radial Schrödinger equation by the method of Numerov was superior to any other method proposed. This was true for $k^2 > 0$, giving a closed channel, and for $k^2 < 0$, giving an open channel. Riehl, Diestler and Wagner [27] have performed numerical experiments designed to compare the efficiency of a one-dimensional Numerov method and a one-dimensional piecewise analytical solution method with one perturbation correction. Their piecewise analytical method was based upon a piecewise constant approximation to $p(r)$. We have performed a number of experiments which compare the efficiency of our perturbation method with the widely used off-set tangent method of Gordon [15]. By combining our results with those of

Riehl et al., we will be able to get a better understanding of the relative merits of our perturbation series method, the Numerov method and the piecewise analytical solution method of Gordon.

Our numerical experiments are divided into two test cases—scattering and bound state problems.

In order to begin the scattering test calculations for a given value of E and l, we need values for R_0 (the starting point) and R_M (the beginning of the asymptotic region). From previous similar calculations we can obtain rough estimates for both R_0 and R_M. By performing a series of calculations where R_0 and R_M are varied until changes in the reactance are less than some allowed tolerance, we can obtain the values of R_0 and R_M to be used in a test calculation. See [30]. For our first three test problems we have employed initial estimates of R_0 and R_M used by Luthey [23] in her A–B_2 model calculations.

In Table 2 we list the relevant parameters for our first three scattering calculations. We recall that in each subinterval we allow $\varepsilon_k^n(1)$ to be no larger than TOL1 and no smaller than TOL2. In addition, once we reach the asymptotic region, we terminate our calculation when variations in the reactance are less than RTOL.

TABLE 2
$A-B_2$ parameters.

λ^2	1000.0
R_0	.78
R_M	4.0
TOL1	1.0×10^{-4}
TOL2	7.0×10^{-5}
RTOL	1.0×10^{-4}

For the first three problems we have considered, we have set $l=0$. For the perturbation series method with $n=0,1,2,3$ and for Gordon's off-set tangent algorithm, we have recorded the data by tabulating the phase shifts, in units of π, the number of subintervals required to reach the asymptotic region, the average time it took to propagate the solutions from one subinterval to another and finally the total time of the calculation.

We recall that in the neighborhood of turning points we have resorted to a piecewise linear polynomial approximation to $p(r)$. Hence for the modified basis function approach, we have broken the recorded subintervals into exponential intervals (E) and Airy intervals (A). In addition, the off-set tangent method has difficulty when the argument of the Airy function becomes large and negative. This occurs in the extreme classical regions where the potential is nearly flat. In these regions we have used a piecewise constant polynomial approximation to $p(r)$. We again record both the number of exponential and Airy subintervals. The choice of when to implement the Airy function turning point approximation as well as the extreme classical exponential approximation is discussed in some detail in [30]. We merely note that when the ratio of $q(0)/q'(0)$ falls below 4 or 5 we use the Airy function approximation in turning point regions in our method and when the argument of the Airy function becomes smaller than -1000, we use the exponential approximation in Gordon's method.

Scattering problems.

Problem 1. $E=1000$, $l=0$. In Table 3 we record the number of subintervals and the phase shifts in units of π for the piecewise analytical perturbation series method with full and partial least squares minimization conditions and the off-set tangent method.

In Table 4 we record the average representative interval propagation times together with the total times required to integrate the solution from R_0 to R_M for the perturbation methods and Gordon's algorithm.

TABLE 3
Test data: $E = 1000$, $l = 0$.

	n	δ/π	$\#E$	$\#A$	Total $\#$
full least squares	0	.203	42	3	45
	1	.202	21	3	24
	2	.203	11	4	15
	3	.204	8	4	12
partial least squares	0	.203	43	3	46
	1	.203	21	3	24
	2	.205	11	4	15
	3	.204	10	4	14
	offset	.204	1	24	25

TABLE 4
Average calculation times.

	n	Interval (in milliseconds)	Total (in seconds)
full least squares	0	550	25.1
	1	730	17.3
	2	910	12.6
	3	1150	11.8
partial least squares	0	470	22.2
	1	550	13.5
	2	630	9.5
	3	730	9.9
	offset	650	16.1

We observe that the shortest total time occurred for $n = 2$ with partial least squares minimization conditions. We also note that the total number of subintervals needed to solve the test problem was fairly insensitive as to whether we implemented the full least squares minimization conditions or the partial least squares minimization conditions. In fact, for all the test problems considered, for a given n, the partial least squares solution was always faster than the full least squares solution. As a result, in the calculations that follow, we will only employ the modified basis function approach with partial least squares minimization conditions.

In Table 3 we observe that we do not obtain a substantial reduction in the number of subintervals needed to solve the test problem when going from $n = 2$ to $n = 3$ for both the full and partial least squares minimization methods. This behavior can best be explained by recalling that the calculations were done on a PDP-10 in single precision arithmetic. After a number of calculations have been performed on such a machine we can expect at most 5 or 6 significant figures of accuracy. However, since we expect the higher order terms in the perturbation series expansion to be small compared to the expansion as a whole, there is the possibility that round-off error can affect the calculations when say n is equal to 3. This behavior was observed early in the development of the piecewise analytical perturbation series method. To verify the round-off error hypothesis, we performed numerous test calculations on an IBM 370–168 computer in double precision arithmetic. We found that we were able to include as many as seven perturbation corrections in our approximate solution before the effects of round-off began to dominate the calculations.

Problem 2. $E = 730$, $l = 0$. We record the relevant test data and calculation times in Tables 5 and 6.

TABLE 5
Test data: $E = 730$, $l = 0$.

n	δ/π	$\#E$	$\#A$	Total $\#$
0	.335	45	3	48
1	.331	22	3	25
2	.334	10	4	14
3	.327	10	4	14
offset	.335	1	25	26

TABLE 6
Total time (*in seconds*).

n	Total time
0	23.1
1	14.1
2	8.9
3	9.9
offset	16.7

Problem 3. $E = 550$, $l = 0$. We record the relevant test data and calculation times in Tables 7 and 8.

TABLE 7
Test data: $E = 550$, $l = 0$.

n	δ/π	$\#E$	$\#A$	Total $\#$
0	−.156	49	3	52
1	−.143	21	4	25
2	−.143	11	4	15
3	−.140	12	3	15
offset	−.145	1	26	27

TABLE 8
Total time (in seconds).

n	Total time
0	25.0
1	14.2
2	9.5
3	10.7
offset	17.4

In all three problems considered we see that the $n=2$ case is by and large the fastest method. The slowest method is the case where $n=0$. This is simply a piecewise analytical solution method with a piecewise constant polynomial approximation to $p(r)$.

In a number of problems one wishes to determine the phase shift for a given l value, the same interaction potential and a spectrum of energies. The first time a calculation is done for a given energy and orbital angular momentum quantum number, we evaluate the $(n+1)$st perturbation correction in each subinterval to get an estimate of the local error incurred by the solution. We then apply the formula in (3.9) for some given value of TOL in order to adjust the mesh so that we satisfy the pre-set error tolerance. More computational effort is required to evaluate this extra perturbation correction and to determine whether the local error tolerance satisfies (3.10) than if we had evaluated the approximate perturbation series solution through only the nth perturbation correction with the mesh intervals already determined. However, as we shall see, for subsequent calculations at different collision energies but with the same interaction potential and the same orbital angular momentum quantum number, we can repeat the calculation using the mesh intervals determined by the initial calculation. This eliminates the need to evaluate the $(n+1)$st perturbation correction in each subinterval and it also eliminates the mesh adjustment calculation.

The choice of the proper energy with which to determine the mesh in such energy spectrum calculations is nevertheless a subtle point. Gordon [15] and Luthey [23] set the mesh based upon the highest energy being used. In contrast, we have found that it generally requires more computation time to solve a given problem with fixed n, the same initial point R_0, the same l value, but lower energy. This is partially explained because as the energy decreases we expose a larger section of the steep potential to the left of the turning point. We expect this to be the region where we obtain the smallest mesh intervals. In addition, based upon the results of our test problems, we have found that the convergence property of the perturbation series is related to the energy of the problem we are trying to solve or—more specifically—it is related to the difference (v^*-E) between the effective intermolecular potential and the energy. Hence, if we applied the perturbation series algorithm (fixed n) on a given region in r, for an effective intermolecular potential $v^*(r)$, but for two different energies, we could expect that it would take a smaller number of subintervals to propagate the solution over this region for the higher energy than it would for the lower energy. Ideally we should set R_0 based upon the highest energy and then set the mesh based upon the lowest energy we are considering. This eliminates the error that would be introduced if we used larger mesh intervals than are consistent with a given local error tolerance—especially for problems in which the lowest energy differs substantially from the highest.

In Tables 9–11 we have recorded the data obtained by running our computer code for a variety of energies and angular momentum values. As the energy changes, we have adjusted the initial and terminal points, R_0 and R_M. The corresponding values of R_0 and R_M are recorded as well.

TABLE 9
Test data: $E = 1000, 730, l = 0, 2, 6$.

			$l=0$		$l=2$		$l=6$	
	E	n	δ/π	#	δ/π	#	δ/π	#
$R_0 = .78, R_M = 4.0$	1000	0	.203	46	.160	44	−.029	43
		1	.203	24	.169	23	−.027	23
		2	.205	15	.172	14	−.026	14
		3	.204	14	.172	14	−.025	14
		offset	.204	25	.167	25	−.026	24
$R_0 = .78, R_M = 4.5$	730	0	−.355	48	−.372	48	.449	48
		1	−.331	25	−.368	25	.410	25
		2	−.334	14	−.370	15	.410	15
		3	−.328	14	−.364	14	.412	15
		offset	−.334	26	−.369	26	.439	25

TABLE 10
Test data: $E = 550, 100, l = 0, 2, 6$.

			$l=0$		$l=2$		$l=6$	
	E	n	δ/π	#	δ/π	#	δ/π	#
$R_0 = .79, R_M = 5.0$	550	0	−.156	52	−.186	48	−.424	47
		1	−.143	25	−.183	23	−.429	24
		2	−.143	15	−.188	14	−.436	14
		3	−.140	15	−.181	13	−.429	13
		offset	−.145	27	−.186	25	−.425	24
$R_0 = .80, R_M = 6.0$	100	0	.022	67	−.051	66	.478	63
		1	.020	32	−.056	32	.476	31
		2	.021	18	−.055	18	.477	18
		3	.024	16	−.053	16	.478	17
		offset	.021	34	−.051	34	.477	33

Observe that there is a significant variation in computation time for a given l value and changing energy. Also, there are almost 50% increases in computation time for several methods between $E = 1000$ and $E = 100$. This can partially be attributed to the fact that as the energy decreases substantially, R_M moves far enough to the right to affect the number of subintervals needed in the calculation. However, of more importance, is the point we have already made regarding the convergence property of our perturbation series. We would expect a larger number of subintervals to be used in integrating the solution over a given r region for the $E = 100$ problem, and a given value of l, than for the $E = 1000$ problem.

Eigenvalue problems. From the previous section we recall that in the eigenvalue problem, we must determine the values of E such that $\det \bar{U} = 0$. Generally as we

TABLE 11
Total time (in seconds).

E	n	$l=0$ Time	$l=2$ Time	$l=6$ Time
	0	22.2	21.2	20.8
	1	13.5	13.0	13.0
1000	2	9.5	8.9	8.9
	3	9.9	9.9	9.9
	offset	16.1	16.1	15.2
	0	23.1	23.1	23.1
	1	14.1	14.1	14.1
730	2	8.9	9.5	9.5
	3	9.9	9.9	10.6
	offset	16.7	16.7	16.1
	0	25.0	23.1	22.6
	1	14.2	13.0	13.5
550	2	9.5	8.9	8.9
	3	10.7	9.2	9.2
	offset	17.4	16.1	15.4
	0	32.0	31.6	30.2
	1	17.9	18.0	17.4
100	2	11.4	11.4	9.1
	3	11.4	11.4	12.1
	offset	21.7	21.7	20.9

start at the bottom of the well and move upward in energy, the determinant of $\bar{U}$ will change sign several times indicating the presence of a number of eigenvalues.

There are a couple of ways we can proceed. We can first determine a representative trial mesh for some energy near the middle of the well and we can then begin our energy search at the bottom (top) of the well and move upward (downward) in energy. This will allow us to get a rough estimate of the position of the eigenvalues. In the vicinity of each eigenvalue we can then use a more refined mesh and initiate our root finding technique. An alternative, and the method we have implemented, involves sectioning the potential into several energy regions. In each of these regions we can determine a mesh and perform an energy search to see where the determinant of $\bar{U}$ changes sign. Once an eigenvalue has been bracketed, we can implement the root finding routine.

We will again perform our calculations on the $A-B_2$ model system. However, in contrast to the scattering problem, we must always contend with two turning points in our eigenvalue calculations. In each of these regions we will again employ a piecewise linear potential approximation to $p(r)$.

To start our calculations, we divide our potential well into several sections and we determine a mesh for an energy in each of these sections. Each time we determine a mesh, we adjust the values of R_0 and R_M in a manner similar to the procedure outlined above. However, in the bound state calculations, we determine the final values of R_0 and R_M when variations in the determinant of $\bar{U}$ are less than some allowed tolerance. Once the mesh is set, we perform a search for the eigenvalues in the neighborhood of each of these mesh energies. If we locate a region where det $\bar{U}$ changes sign, the root finding procedure is initialized.

We note that each time we do an energy search and each time the root finding routine performs an iteration, we use the previously determined mesh. We will soon see that this greatly reduces the computation time needed to determine one or more eigenvalues.

We determined the first mesh for $E = -900.0$. We found that $R_0 = .83$ and $R_M = 1.6$. In Table 12 we have recorded the eigenvalue and the number of intervals used for each of the methods we have tested.

TABLE 12
Test data: $E = -900$, $l = 0$.

n	λ_1	$\# E$	$\# A$	Total #
0	−979.3	28	8	36
1	−979.1	16	9	25
2	−979.3	10	10	20
offset	−979.3	0	24	24

We observe that the total number of subintervals decreases for the modified basis function methods as we include higher perturbation corrections in the approximate solution. This behavior is similar to what we observed in the scattering problems. However, we note that we require a larger number of Airy intervals in the bound state calculations than we did in the scattering calculations. This is due to the fact that we must now contend with two turning points.

As we mentioned previously, the great advantage of the piecewise analytical perturbation series method and the piecewise analytical solution method is the ability to quickly repeat a calculation for a given value of l but different energy. In determining each eigenvalue, we will heavily rely upon this feature. Hence, we must determine the time required to propagate the solution from one subinterval to another with the mesh already set. In Table 13 we record average representative times for a repeated interval calculation for our modified basis function methods and Gordon's off-set tangent method.

TABLE 13
Interval repeat times
(in milliseconds).

n	Time
0	33
1	87
2	117
3	188
offset	113

The total calculation time is then composed of the time it took to determine the original mesh, the time it took to perform the energy search and the time it took to do the root finding iterations.

A successful energy search implies that we have found two energies E_1 and E_2 such that $\det \bar{U}(E_1) \cdot \det \bar{U}(E_2) \leqq 0$. Hence we require two propagations with the mesh set. In addition, our root finding routine requires a number of propagations with the mesh set. Essentially, each propagation corresponds to an iteration. The total number of propagations with the mesh set usually varied between 4 and 8.

If we use the data in Tables 4, 12 and 13 and if, for the sake of convenience, we take the number of propagations with the mesh set equal to 6, then we can get an idea of the total computation time required to determine each eigenvalue.

In Table 14 we record the times for the first eigenvalue.

TABLE 14
λ_1 *total time* (*in seconds*).

n	Original	Repeated	Total
0	18.4	14.6	33.0
1	14.7	19.3	34.0
2	12.8	18.4	31.2
offset	15.6	21.7	37.3

We note that although the case $n = 0$ was by far the slowest method in the original calculation, it has the fastest repeat calculation time. As we will see in Fig. 2, the eigenvalue, λ_1, is very near the bottom of the potential well. This implies that the two turning points are quite close together. As it turns out our perturbation series calculation in this region with $n = 3$ is almost totally dominated by Airy intervals. As a result, we have not included any data for the perturbation series method with $n = 3$ for λ_1.

In Tables 15–22 we have recorded the interval numbers and the total times it took to determine the remaining eigenvalues of the $A - B_2$ system for each of the methods we have been considering. The energies listed correspond to the energies for which a mesh was determined. In each case we have recorded the values of R_0 and R_M that were used in the calculations.

TABLE 15
Test data: $E = -500$, $R_0 = .79$, $R_M = 1.8$, $l = 0$.

n	λ_2	$\#E$	$\#A$	Total $\#$
0	−472.1	33	10	43
1	−472.0	17	12	29
2	−472.5	12	11	23
3	−472.5	9	12	21
offset	−472.6	0	29	29

TABLE 16
λ_2 *total time* (*in seconds*).

n	Original	Repeated	Total
0	22.0	17.8	39.8
1	17.2	22.7	39.9
2	14.7	21.2	35.9
3	14.4	24.4	38.8
offset	18.9	26.2	45.1

TABLE 17
Test data: $E = -300$, $R_0 = .78$, $R_M = 1.95$, $l = 0$.

n	λ_3	#E	#A	Total #
0	−287.7	41	10	51
1	−287.7	20	12	32
2	−288.3	12	13	25
3	−288.3	10	14	24
offset	−288.3	0	32	32

TABLE 18
λ_3 *total time* (*in seconds*).

n	Original	Repeated	Total
0	25.8	19.9	45.7
1	18.8	24.8	43.6
2	16.0	23.0	39.0
3	16.4	27.7	44.1
offset	20.8	28.9	49.7

TABLE 19
Test data: $E = -150$, $R_0 = .78$, $R_M = 2.3$, $l = 0$.

n	λ_4	#E	#A	Total #
0	−136.5	48	10	58
1	−136.5	21	14	35
2	−136.9	14	14	28
3	−136.9	13	14	27
offset	−136.9	0	35	35

TABLE 20
λ_4 *total time* (*in seconds*).

n	Original	Repeated	Total
0	29.1	21.7	50.8
1	20.7	27.3	48.0
2	17.9	25.8	43.7
3	18.6	32.2	50.8
offset	22.8	31.6	54.4

TABLE 21
Test data: $E = -50$, $R_0 = .78$, $R_M = 2.80$, $l = 0$.

n	λ_5/λ_6	#E	#A	Total #
0	−78.54/−29.40	58	10	68
1	−78.53/−29.43	25	13	38
2	−78.70/−29.66	16	15	31
3	−78.34/−29.49	19	12	31
offset	−78.56/−29.57	0	32	39

TABLE 22
λ_5, λ_6 *total time* (*in seconds*).

n	Original	Repeated	Total
0	33.8	24.4	58.2
1	22.2	29.2	51.4
2	19.8	28.5	48.3
3	21.7	39.4	61.1
offset	25.4	35.3	60.7

A number of observations are in order. As in the scattering problems, the case $n = 2$ is always the fastest. Although the single energy calculations with $n = 0$ were totally uncompetitive with the other four methods, the fact that its repeat calculation time is so fast makes it competitive and in some cases better than all but the $n = 2$ method. In fact as the number of iterations required in the root finding routine increases, the $n = 0$ method can actually become the method of choice.

In Fig. 2 we plot the position of the six eigenvalues for our $A - B_2$ system.

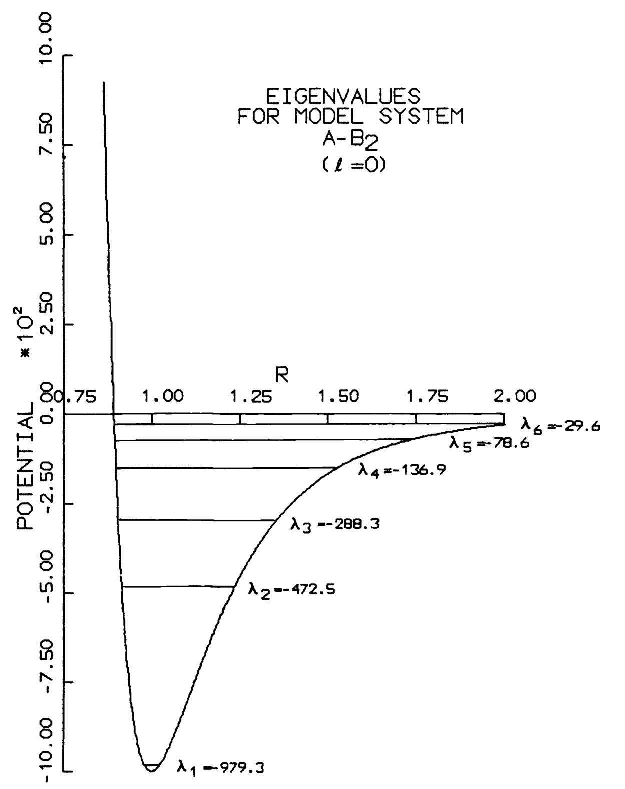


FIG. 2. *Eigenvalues for model $A - B_2$ system* ($l = 0$).

4.3. Discussion. In all of the test problems considered—both scattering and bound state—we have seen that our piecewise analytical perturbation series solution method with $n = 2$ and partial least squares minimization conditions is more efficient than Gordon's off-set tangent method and it is more efficient than the other modified basis function methods we have considered. We have not, however, made a comparison between our solution method and the one-dimensional Numerov method. We recognize the fact, however, that in order to get a better understanding of the relative merits of the piecewise analytical perturbation series solution method, the piecewise analytical solution method and Numerov's method, a series of numerical experiments should be performed which test the three solution methods on a variety of problems. We can, however, draw a number of conclusions about the three solution methods by combining our results with those of Riehl, Diestler and Wagner [27].

Riehl, Diestler and Wagner have compared the one-dimensional Numerov method with a one-dimensional piecewise analytical solution method which includes one perturbation correction. Their method was based upon a piecewise constant polynomial approximation to $p(r)$. They found that for values of l near zero the piecewise analytical solution method they considered was faster than the Numerov method. However, for large l (10 or 20) the Numerov method was almost always faster than the piecewise analytical perturbation series method they considered. This behavior can best be explained by remembering the convergence property of our perturbation series. We recall that our perturbation series converges more rapidly the larger the value of $v^* - E$. We note that for a given value of E, the quantity $v^* - E$ decreases as the angular momentum increases. Hence with E constant we would expect the perturbation method to require a larger number of subintervals to propagate the solution over a region in r for large l than for small l. The more subintervals required, the longer the calculation takes. The Numerov method, however, computes the wavefunction directly. As l increases, so does the effective intermolecular potential. This results in a decrease in the local wavenumber of the wavefunction. Hence the wavefunction oscillates less rapidly for larger l. Thus a coarser grid can be used to represent the wavefunction. The larger the intervals, the quicker the calculation.

Although it is difficult to accurately compare the efficiency of our piecewise analytical perturbation series method with the method used by Riehl, Diestler and Wagner, we can note the following points. The method they have used to produce their approximate solution gives a solution similar in form to the perturbation scheme in (2.2) with $(p - \tilde{p})(r)$ expanded through the quadratic term. We have experimented with a variety of perturbation schemes and have found the method employed in (2.9) and (2.19) with partial least squares minimization conditions to be the most efficient method for a given n. In addition, we have observed in our test cases that the piecewise analytical solution method with two perturbation corrections included in the zeroth order solution is a faster solution method than a zeroth order solution with one perturbation correction.

Based upon the test results of Riehl, Diestler and Wagner, the experiments we have done with various perturbation schemes and our test results, we anticipate our method to be more efficient than the Numerov method for a single scattering calculation at low angular momentum. In addition, we anticipate our method to be more competitive than existing piecewise analytical solution methods with the Numerov method at large values. Finally, our piecewise analytical perturbation series method is the method of choice for energy spectrum scattering calculations.

No study similar to the one of Riehl, Diestler and Wagner [27] has been conducted for the eigenvalue problem. Ixaru et al. [2] have examined the stability of a first order perturbative piecewise analytical solution method against roundoff errors and they have compared the efficiency of their method with the Numerov method for eigenvalue problems. However, their results were based upon equispaced meshes and they did not optimize their algorithm with respect to computation time by implementing a variable step method. We note that the eigenvalue calculation is similar to the energy spectrum calculations in the scattering problem. A number of repeat calculations are required when using the root-finding routine. We again anticipate our piecewise analytical perturbation series method with two perturbation corrections and partial least squares minimization conditions to be the method of choice in such calculations.

As a final remark, we recall that the motivation for considering a piecewise analytical perturbation series solution method based upon a piecewise constant polynomial approximation to $p(r)$ was the fact that since exponentials were less expensive to evaluate than Airy functions and since we could include a number of analytical perturbation corrections in our zeroth order solution, we hoped that computational gains could be obtained over the widely used offset tangent method of Gordon. Our numerical experiments have indeed confirmed these hypotheses. However, a major drawback to the piecewise constant method is the fact that it can have problems in the neighborhood of turning points. The Airy function method, however, does not suffer from this difficulty. It is natural to ask whether a piecewise analytical perturbation series solution method based upon piecewise linear polynomial approximations to $p(r)$ would be a more efficient method of solution for scattering and bound state problems. Such a method would have the best of both worlds in the sense that it would have the increased accuracy arising from the addition of perturbation corrections in its zeroth order solution and it would not have difficulty in the region around turning points. As it turns out, our perturbation series arguments can be generalized to include piecewise linear polynomial approximations to $p(r)$. As to whether such a method would be computationally more efficient than our method, a number of numerical experiments would have to be performed.

5. Remarks. We have developed a piecewise analytical perturbation series method (PAPSM) for solving the radial Schrödinger equation scattering and bound state problems. PAPSM is a shooting method based on a perturbation series solution of the radial Schrödinger equation when the potential function $p(r)$ is replaced by a piecewise constant polynomial $\tilde{p}(r)$. We saw that the efficiency of the method was related to the ability to adaptively adjust the mesh so that the local error incurred was less than some pre-set error tolerance. The mesh selection procedure required that we knew how our local error estimate varied with the mesh length h_k. By making use of the rigorous error bounds derived in another paper we were able to infer the dependence of our local error estimate on h_k. We performed a number of experiments designed to compare the accuracy and efficiency of the method with Gordon's piecewise analytical solution method. We found that by including several perturbation corrections in our approximate zeroth order solution, we obtained a numerical method that was 40–50% faster than Gordon's method for the scattering problems considered and 16–22% faster than Gordon's method for the bound state problems considered.

Acknowledgment. The author wishes to thank Dr. D. G. M. Anderson, Dr. G. Baker and Dr. R. Gordon for their helpful ideas and suggestions concerning this material.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, eds., *Handbook of Mathematical Functions*, National Bureau of Standards U.S. Government Printing Office, Washington, DC, 1970.

[2] GH. ADAM, L. GR. IXARU AND A. CORCIOVEI, *A first order perturbative numerical method for the solution of the radial Schrödinger equation*, J. Comp. Phys., 22 (1976), pp. 1–33.

[3] D. G. M. ANDERSON, private communications.

[4] P. B. Bailey, M. K. Gordon and L. F. Shampine, *Automatic solution of the Sturm–Liouville problem*, ACM Trans. Math Software, 4 (1978), pp. 193–208.

[5] R. B. Bernstein, *Quantum mechanical (phase shift) analysis of differential elastic scattering of molecular beams*, J. Chem. Phys., 33 (1960), pp. 795–804.

[6] G. Birkhoff, C. de Boor, B. Swartz and B. Wendroff, *Rayleigh–Ritz approximation by piecewise cubic polynomials*, SIAM J. Numer. Anal., 3 (1966), pp. 188–203.

[7] J. M. Blatt, *Practical points concerning the solution of the Schrödinger equation*, J. Comp. Phys., 1 (1967), pp. 382–386.

[8] W. E. Boyce and R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, John Wiley, New York, 1977.

[9] J. Canosa and R. G. De Oliveira, *A new method for the solution of the Schrödinger equation*, J. Comp. Phys., 5 (1970), pp. 188–207.

[10] G. F. Carrier and C. E. Pearson, *Ordinary Differential Equations*, Blaisdell, Waltham, MA, 1968.

[11] J. W. Cooley, *An improved eigenvalue corrector formula for solving the Schrödinger equation for central fields*, Math. Comp., 15 (1961), pp. 363–374.

[12] A. M. Dunker and R. G. Gordon, *Solution for bound state wavefunctions and matrix elements by the piecewise analytic method*, J. Chem. Phys., 64 (1976), pp. 4984–4994.

[13] L. Fox and D. F. Mayers, *Computing Methods for Scientists and Engineers*, Clarendon Press, Oxford, 1968.

[14] R. G. Gordon, *New method for constructing wavefunctions for bound states and scattering*, J. Chem. Phys., 51 (1969), pp. 14–25.

[15] ———, *Quantum scattering using piecewise analytic solutions*, Methods in Computational Physics, vol. 10, B. Alder, S. Fernbach, M. Rottenberg, eds., Academic Press, New York, 1971, pp. 81–109.

[16] F. E. Harris, *Expansion approach to scattering*, Phys. Rev. Letters, 19 (1967), p. 173.

[17] L. Gr. Ixaru, *An algebraic solution for the Schrödinger equation*, Internal Report IC/6/69, International Centre of Theoretical Physics, Trieste, 1969.

[18] ———, *The error analysis of the algebraic method for solving the Schrödinger equation*, J. Comp. Phys., 9 (1972), pp. 159–163.

[19] O. Johnson, IBM Scientific Center (Houston), Report No. 320–2343, Houston, 1968.

[20] H. B. Keller, *Numerical Methods for Two Point Boundary Value Problems*, Blaisdell, Waltham, MA, 1968.

[21] S. K. Knudson and B. Kirtman, *Variation-perturbation treatment of scattering problems. I. Potential scattering by a central field*, Phys. Rev., A3 (1971), pp. 972–978.

[22] W. A. Lester and R. B. Bernstein, *Structural features of the S-matrix for the rotational Excitation of homonuclear diatomic molecules by atom impact: Close coupled versus approximate computations*, Chem. Phys. Letters, 1 (1967), pp. 207–210.

[23] Z. A. Luthey, *Piecewise analytical solutions method for the radial Schrödinger equation*, Ph.D. thesis, Applied Mathematics, Harvard University, Cambridge, MA, 1974.

[24] A. Messiah, *Quantum Mechanics*, John Wiley, New York, 1958.

[25] N. F. Mott ahd H. S. W. Massey, *The Theory of Atomic Collisions*, Clarendon Press, Oxford, 1949.

[26] R. K. Nesbet, *Analysis of the Harris variational method in scattering theory*, Phys. Rev., 175 (1968), pp. 134–142.

[27] J. P. Riehl, D. J. Diestler and A. F. Wagner, *Comparison of perturbative and direct-numerical-integration techniques for the calculation of phase shifts for elastic scattering*, J. Comp. Phys., 15 (1974), pp. 212–225.

[28] A. Rosenthal and R. G. Gordon, *Piecewise analytic wavefunctions for bound states and scattering*, J. Chem. Phys., 64 (1979), pp. 1621–1629.

[29] L. I. Schiff, *Quantum Mechanics*, McGraw Hill, New York, 1968.

[30] M. D. Smooke, *Piecewise analytical perturbation series solutions of the radial Schrödinger equation*, Ph.D. thesis, Applied Mathematics, Harvard University, Cambridge, MA, 1978.

[31] ———, *Error estimates for piecewise perturbation series solutions of the radial Schrödinger equation. I. One-dimensional case*, Rep. SAND80-8611, Sandia Laboratories, Livermore, CA, 1980.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

SOME EXTENSIONS OF AN ALGORITHM FOR SPARSE LINEAR LEAST SQUARES PROBLEMS*

MICHAEL T. HEATH†

Abstract. Several algorithms are developed which extend the method of George and Heath for sparse linear least squares problems to include rank-deficient problems, linear equality constrained problems, and updating of solutions. An application of these methods to the solution of sparse square nonsymmetric linear systems is also presented.

Key words. sparse least squares, rank deficiency, numerical rank determination, updating, constraints

1. Introduction.

In [9] an algorithm is presented for solving the unconstrained linear least squares problem

$$\min_x \|Ax-b\|_2, \tag{1.1}$$

where A is an $m\times n$ matrix, $m \geqq n$, rank $(A)=n$, and x and b are vectors of appropriate dimension. In addition, for sparse problems the effectiveness of the method depends on the sparsity of A^TA. The latter matrix, unfortunately, is not necessarily as sparse as A, as can readily be seen by considering a matrix A having at least one relatively full row. The purpose of this paper is to relax some of these restrictions by extending the algorithm of [9] to include least squares problems having matrices of arbitrary shape and rank, problems having linear equality constraints, and problems having a few "nasty" rows which cause excessive fill in the structure of A^TA.

The classical approach to solving the linear least squares problem is via the system of normal equations

$$A^TAx=A^Tb. \tag{1.2}$$

The $n\times n$ symmetric positive definite matrix $B=A^TA$ is factored using Cholesky's method into R^TR, where R is upper triangular, and then x is computed by solving the two triangular systems $R^Ty=A^Tb$ and $Rx=y$. This algorithm has several attractive features for large sparse problems. The Cholesky factorization does not require pivoting for stability so that the ordering for B (i.e., column ordering for A) can be chosen based on sparsity considerations alone. Moreover, there exists well-developed software for obtaining a good ordering in advance of any numerical computation, thereby allowing use of a static data structure. Another advantage is that the row ordering of A is irrelevant so that the rows of A can be processed sequentially from an auxiliary input file in arbitrary order, and A need never be represented in fast storage in its entirety at any one time. Unfortunately the normal-equations method may be numerically unstable. This is due to the potential loss of information in explicitly forming A^TA and A^Tb, and to the fact that the condition number of B is the square of that of A. Moreover, the normal-equations method is not easily extended to handle the more general circumstances, such as possible rank deficiency, addressed in this paper.

* Received by the editors April 21, 1981, and in revised form November 9, 1981. This research was supported by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy, under contract W-7405-eng-26 with the Union Carbide Corporation.

† Mathematics and Statistics Research Department, Computer Sciences Division, Union Carbide Corporation, Nuclear Division, Oak Ridge, Tennessee 37830.

A well-known stable alternative to the normal equations is provided by orthogonal factorization. An orthogonal matrix Q is computed which reduces $[A, b]$ to the form

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix}, \qquad Qb = \begin{bmatrix} c \\ d \end{bmatrix}, \tag{1.3}$$

where R is $n \times n$ and upper triangular. Since the Euclidean norm is invariant under orthogonal transformation, the solution to (1.1) may be obtained by solving the triangular system $Rx = c$. The matrix Q usually results from Gram–Schmidt orthogonalization or from a sequence of Householder or Givens transformations. Both the Gram–Schmidt and Householder algorithms process the unreduced part of the matrix A by columns and can cause severe intermediate fill for large sparse problems. The use of Givens rotations is much more attractive in that the matrix is processed by rows, gradually building up R, and intermediate fill is confined to the working row. This approach, implemented with a good column ordering and an efficient data structure, is the basis for the algorithm described in § 2.

Throughout this paper standard techniques of numerical linear algebra, especially linear least squares, are used without explicit references. In such cases a full discussion of the methods employed, with citations of original sources, will be found in [17].

2. The basic algorithm. The algorithm is developed in detail in [9]. Its motivation is to combine the flexibility, convenience and low storage requirements of the normal equations with the stability of orthogonal factorization. The steps of the algorithm are as follows:

ALGORITHM 1
1. Determine the structure (not the numerical values) of $B = A^TA$.
2. Find an ordering for B (column ordering for A) which has a sparse Cholesky factor R.
3. Symbolically factorize the reordered B, generating a row-oriented data structure for R.
4. Compute R and c by processing the rows of $[A, b]$ one by one using Givens rotations.
5. Solve $Rx = c$, then permute the components of x back into the original column ordering of A.

Steps 1 through 3 of Algorithm 1 are the same as would be used in a good implementation of the normal equations method. These steps may be carried out very efficiently using existing sparse matrix software, such as SPARSPAK [11]. It is important to emphasize that the data structure for R is generated in advance of any numerical computation, and therefore dynamic storage allocation to accommodate fill during the numerical computation is unnecessary. See [9] for a detailed description of an appropriate data structure. The order in which the rows of A are processed in step 4 does not affect the structure of R. Therefore, the rows may be accessed from an external file one at a time in arbitrary order. In particular, a row ordering scheme may be used to reduce the amount of computation associated with intermediate fill in the working row or to enhance stability when dealing with problems having widely varying weights or row norms. Thus, Algorithm 1 requires the same storage and exploits sparsity to the same degree as the normal equations, allows convenient use of auxiliary storage, and in addition is numerically stable.

The details of step 4 of the algorithm are of some interest and will be needed for reference later. Let a^T be a given row of A to be processed next, with the components of a^T suitably permuted to reflect the new column ordering for A. Let j be the subscript of the first nonzero component of a^T. It is shown in [9] that there is space in row j of the data structure of R to accommodate a^T. If row j of the data structure is still vacant, as will likely be the case early in the row-by-row processing, then a^T may simply be placed into row j of the data structure. If, on the other hand, row j of the data structure is already occupied by previously stored numerical values, then row j may be used to annihilate the first nonzero of a^T with a Givens rotation. It is further shown in [9] that the resulting "shorter" row can also be accommodated in the data structure, even though some fill may have occurred as a result of the transformation. Thus, the process may be repeated until either an unoccupied row is found in which to place the working row or all its nonzeros have been annihilated.

3. Rank deficiency. In this section the requirement that $m \geqq n$ is dropped, and we allow rank $(A) = k \leqq \min\{m, n\}$. When $k < n$ there is no longer a unique solution to (1.1). However, among the infinite set of vectors x which minimize the least squares residual, there is a unique solution x of minimum norm. This minimum-norm solution is the result ordinarily desired in the rank-deficient case and is the solution which would be given by the Moore–Penrose pseudoinverse, although the latter is not a useful computational tool. It is worthwhile observing that for many practical purposes a basic solution (i.e., a least squares solution having at most k nonzero components) is equally useful and rather easier to compute (as will be seen below). These solutions are introduced in [20] and quantitatively compared in [14].

In practice the rank of A is not usually known in advance, and therefore it must be estimated as part of the solution procedure. Numerically, the choice of rank may not be clear cut and may be somewhat dependent on the particular solution algorithm. The situation is further complicated by the fact that the rank may not be well determined in that a small perturbation to the matrix may cause the rank to change. Nevertheless, heuristic rules which are reasonable, if not entirely rigorous, usually do an adequate job of numerical rank determination in most contexts.

Perhaps the most reliable method for computing the minimum-norm solution to a rank-deficient, rectangular linear system when the rank is unknown is by means of the singular value decomposition

$$A = U\Sigma V^T, \tag{3.1}$$

where U and V are orthogonal matrices of order m and n, respectively, and Σ is an $m \times n$ nonnegative diagonal matrix. The diagonal entries of Σ are the singular values of A. In theory, exactly $k = \text{rank}(A)$ of the singular values are positive. In practice, some of the computed singular values may be very small but nonzero. In this case the rank is estimated by declaring those singular values whose magnitudes fall below some tolerance to be negligible. The choice of an appropriate tolerance depends on the accuracy with which the data are known and on the machine precision. The choice may be obvious if there is a sharp break or gap between the "large" and "small" singular values, but this may well not be the case. Once the rank is selected the minimum-norm solution is given by

$$x = V\Sigma^+ U^T b,$$

where Σ^+ is an $n \times m$ diagonal matrix whose nonzero diagonal entries are the reciprocals of the corresponding nonnegligible singular values of A.

Unfortunately, this decomposition is not very useful for large sparse least squares problems. Not only are the matrices U and V generally full, but the orthogonal transformations successively applied to A in computing the decomposition cause an unacceptable amount of intermediate fill before the diagonal form is ultimately reached.

A more promising approach is offered by the orthogonal factorization (1.3). Even in the rank-deficient case this factorization exists and can be stably computed by a sequence of orthogonal transformations. However, the triangular factor R is then no longer nonsingular, so that it cannot be used directly to solve the least squares problem. In the Householder and Gram–Schmidt algorithms A is reduced to triangular form by annihilating the subdiagonal elements of successive columns. What is ordinarily done when rank deficiency is expected is to interchange columns at each step so that the unreduced column of largest norm is brought into the next position to be reduced. After k steps this process yields a factorization of the form

$$QAP = \begin{bmatrix} R & S \\ 0 & T \end{bmatrix}, \tag{3.2}$$

where R is $k \times k$ and upper triangular and P is a permutation matrix which performs the column interchanges. In theory, if rank $(A) = k$, then R is nonsingular and $T = 0$. In practice, with finite precision arithmetic, the computed elements of T become small but remain nonzero. Thus, as before, the rank is estimated by terminating the factorization process at some point and declaring the remaining unreduced portion T to be negligible. Numerous criteria have been proposed for making this choice, usually based on some measure of the size of T or on the condition of the resulting triangular matrix R (see, for example, [1], [13], [16], [17], [18], [21]).

At this point a basic solution may be obtained from the factorization (3.2) by solving the triangular system $Ry = c$, where the vector c consists of the first k components of Qb, and setting

$$x = P\begin{bmatrix} y \\ 0 \end{bmatrix}.$$

To compute the minimum-norm solution, however, requires more work. Continuing from the factorization (3.2), the block S is annihilated by a sequence of orthogonal transformations from the right, resulting in the complete orthogonal factorization

$$QAV = \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix}, \tag{3.3}$$

where R is again $k \times k$, upper triangular and nonsingular and V is an orthogonal matrix of order n. (Note that P has been absorbed into V and that the R in (3.3) is not the same as R in (3.2).) The minimum-norm solution is now obtained by solving the triangular system $Ry = c$ and setting

$$x = V\begin{bmatrix} y \\ 0 \end{bmatrix}.$$

Although this approach works well for small problems, it has a number of drawbacks when used in conjunction with Algorithm 1 for large sparse problems. Obviously, the processing of A by columns is inconsistent with the row-oriented processing of Algorithm 1. Moreover, the column ordering in Algorithm 1 is fixed in advance of any numerical computation so as to produce a triangular factor with low fill. Thus, the column interchanges required in obtaining (3.2) would not be permissible. Finally, the additional orthogonal transformations needed to annihilate S in order to obtain (3.3) would cause fill in R which might be excessive, and in any case would not be anticipated by the fixed data structure for the triangular factor. We circumvent these difficulties by first showing that form (3.2) can be obtained without column pivoting during the factorization process, and then showing that the minimum-norm solution can be computed directly from (3.2) without the additional processing needed to obtain (3.3).

If Algorithm 1 is applied to a matrix A of rank $k<n$ using exact arithmetic, the result is an $n \times n$ triangular factor of rank k which has $n-k$ zeros on its diagonal. Thus, R might be thought of as schematically depicted in Fig. 1(a). However, recalling the details of step 4 of Algorithm 1, the working row is always placed into the data structure so that its diagonal entry is nonzero, and diagonal entries can only grow as a result of further processing with Givens rotations. Thus, if a diagonal entry of R is zero, then that row must have been untouched by the Givens processing, and therefore all its entries must be zero, as depicted in Fig. 1(b). Now a matrix of the latter form can be permuted into the form (3.2) by row interchanges (Fig. 1(c)) and column interchanges (Fig. 1(d)). As we shall see below, however, access to R and S in their original locations is all that is required for our purposes, so that such row and column interchanges need not actually be performed. For simplicity we continue to use the notation of (3.2) although in practice this form need not be obtained explicitly.

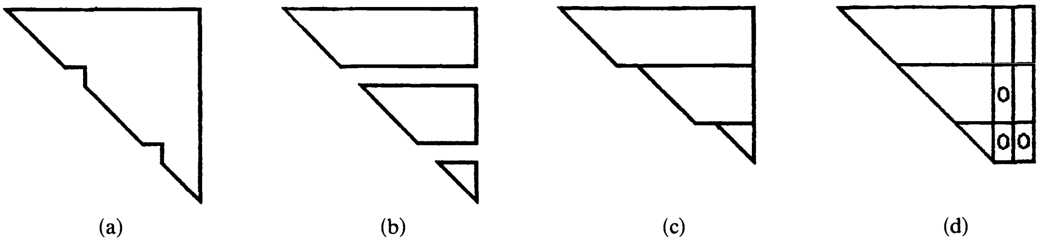


FIG. 1

As usual, the situation is more complicated in finite precision arithmetic. One might expect numerical rank deficiency in A to be revealed by the presence of $n-k$ small diagonal elements in the triangular factor, and this is usually, though not always, the case (see example of Wilkinson cited below). It is no longer true, however, that a diagonal entry being small implies that the remainder of the row is also negligible. There are two ways of coping with this difficulty. One way is to test against some small tolerance, rather than exact zero, when checking symbolic nonzeros in the working row to see if they are in fact numerically nonzero. If this is done then the algorithm proceeds just as described in the preceding paragraph and a triangular factor of the form of Fig. 1(b) results. A drawback of this method is that the tolerance must be set before processing is begun, so that it must be an absolute tolerance which cannot take the scale of the problem into account unless the user can supply this information in advance.

A second alternative is to process all of the rows of A using an exact zero test in step 4, then examine the diagonal of the resulting triangular factor for small entries. Any row whose diagonal element falls below some tolerance may be considered to have resulted from premature termination of normal processing. Processing is therefore resumed on such a row, including the corresponding right-hand side component, until the row is annihilated in the usual manner, and a row of zeros is left in its original location in the data structure. Some care must be exercised with regard to the order in which the rows having small diagonal elements are processed. The reason for this is that the further processing of a row cannot affect any row above its original location, but may fill in a previously small diagonal element below. Thus, these rows should be processed in order of increasing row subscript as they are encountered along the diagonal, and the rank deficiency is equal to the number of zero rows after all row processing is complete. Although it is slightly more complicated than the first method, this approach conveniently allows a relative tolerance to be used in the test for small diagonal elements, since their magnitudes may be compared, for example, to the largest diagonal element.

Either of these methods results in triangular factor of the form of Fig. 1(b) and a corresponding estimate for the rank. Although it is undoubtedly somewhat less robust in determining the rank than algorithms using explicit column pivoting during the factorization process, the author has found this approach to work very well in practice, even for small dense problems. For example, on the 36 test cases generated by PROG2 in [17] this scheme agrees with the rank determined by Householder transformations with column pivoting (subroutine HFTI) using the same absolute tolerances for each algorithm. On the other hand, it is possible, though unlikely to occur in practice, for a triangular matrix to be severely ill conditioned, yet have no small diagonal elements. The example devised by Wilkinson [19] having all diagonal elements equal to 1 and all elements above the diagonal equal to -1 illustrates this point. In such a case the numerical rank deficiency would not be detected by the algorithm described thus far. However, it could be detected by using the triangular factor to solve the linear systems required to estimate the condition number [4], so that the user could at least be warned of such pathology.

We turn now to the problem of computing the minimum-norm solution directly from a factorization of the form (3.2). We first observe that although there is no unique least squares solution in the rank-deficient case, there is still a unique point in the range space of A closest to the right-hand side vector b. The indeterminacy results from the representation of this unique point as a particular solution, such as the basic solution

$$\bar{y} = \begin{bmatrix} y \\ 0 \end{bmatrix},$$

where y is the solution to the $k \times k$ system $Ry = c$, plus an arbitrary solution to the homogeneous system. Since any least squares solution can be represented as a sum of $\bar{y}$ and a null space component, the minimum-norm solution can be determined by finding the point in the null space closest to $\bar{y}$, which is another least squares problem. In particular, the minimum residual vector for this latter problem is the minimum-norm solution to the original least squares problem. Thus, we may regard the solution procedure as a sequence of two projections: first the vector b is projected onto the range space of A, then the point so determined is projected onto the null space of A. If we let the $k \times (n-k)$ matrix K be the solution to the linear system $RK = S$, then the columns of the $n \times (n-k)$ matrix

$$\begin{bmatrix} K \\ -I \end{bmatrix}$$

form a basis for the null space of A. Thus, the minimum-norm solution to the original least squares problem is given by the minimum residual vector for the least squares problem

$$\begin{bmatrix} K \\ -I \end{bmatrix} z \cong \begin{bmatrix} y \\ 0 \end{bmatrix}. \tag{3.4}$$

Problem (3.4) can be solved by means of the orthogonal factorization

$$U\begin{bmatrix} K \\ -I \end{bmatrix} = \begin{bmatrix} L^T \\ 0 \end{bmatrix}, \qquad U\begin{bmatrix} y \\ 0 \end{bmatrix} = \begin{bmatrix} s \\ t \end{bmatrix}, \tag{3.5}$$

where U is an orthogonal matrix of order n and L is a lower triangular matrix of order $n-k$, but the solution z need not be computed explicitly since only the residual vector is needed and this is given by

$$\begin{bmatrix} y-Kz \\ z \end{bmatrix} = U^T\begin{bmatrix} s-L^Tz \\ t \end{bmatrix} = U^T\begin{bmatrix} 0 \\ t \end{bmatrix}.$$

Providing the rank deficiency $n-k$ is small, as is usually the case, the orthogonal factorization (3.5) can be computed by any method, such as Householder transformations, suitable for small dense problems.

Observe that the systems $Ry=c$ and $RK=S$ are easily solved using the sparse triangular factor already computed. The vector c consists of the k components of the transformed right-hand side corresponding to the nonzero rows of the triangular factor. The matrix S consists of the columns of the triangular factor whose diagonal entries have been declared zero, and these are easily extracted from the data structure one by one as needed. In solving these triangular systems it is unnecessary to extract R from the data structure or write a special back substitution routine to skip over the zero rows, since the same effect may be obtained by simply setting the "zero" diagonal elements equal to 1 and noting that the corresponding components of the right-hand sides, both for c and for columns of S, will already be zero. In this way the same back substitution routine used for the full-rank case will give correct results with no division by zero.

We thus arrive at the following algorithm for large sparse rank-deficient linear least squares problems:

ALGORITHM 2

1. Perform steps 1 through 4 of Algorithm 1, carrying out additional Givens processing, if necessary, to annihilate any rows having diagonal elements falling below given tolerance. The result is a permuted form of (3.2) with T negligible.
2. Solve $Ry=c$.
3. Solve $RK=S$.
4. Compute the orthogonal factorization (3.5).
5. $x = U^T\begin{bmatrix} 0 \\ t \end{bmatrix}$.

This approach to handling rank deficiency is similar to that in [7] and [14], although these references are not concerned with preservation of sparsity. In contrast to our geometric derivation, the algorithm can also be derived analytically or algebraically. In the former category, an expression can be written for the components of a general least squares solution in terms of R and S and then its norm minimized by differentiation. In the latter category, the formula for the minimum-norm solution to an underdetermined linear system of full row rank [6] can be applied to the matrix $[R, S]$.

4. Updating. As we have noted, if an otherwise sparse matrix A has a few rows with a relatively large number of nonzeros, the resulting fill in A^TA (and hence in its Cholesky factor) would severely limit the usefulness of Algorithm 1. In such cases it is desirable to withhold these "nasty" rows from initial Givens processing, then update the resulting solution to incorporate the effects of the omitted rows. Such updating procedures are common in least squares applications when new observations are added to a previously solved problem. However, in the present context it is important to note that only the solution is updated and not the factorization, since otherwise our purpose would be defeated. Thus, in this section we consider the least squares problem

$$\begin{bmatrix} A \\ E \end{bmatrix} x \cong \begin{bmatrix} b \\ f \end{bmatrix}, \tag{4.1}$$

where A is $m \times n$, E is $p \times n$, and vectors b and f are dimensioned accordingly. We assume that A is sparse and has rank n, and that p is relatively small compared to n.

We begin by applying Algorithm 1 to $[A, b]$, so that we have R and c as in (1.3). Let y be the solution to the linear system $Ry = c$ and write the solution to problem (4.1) as $x = y + z$, so that we seek to determine z. Let $r = f - Ey$. Then since

$$Q(b - Ax) = \begin{bmatrix} c \\ d \end{bmatrix} - \begin{bmatrix} R \\ 0 \end{bmatrix} x = \begin{bmatrix} R(y-x) \\ d \end{bmatrix} = \begin{bmatrix} -Rz \\ d \end{bmatrix}$$

and $f - Ex = r - Ez$, we see that z is given by the solution to the least squares problem

$$\begin{bmatrix} R \\ E \end{bmatrix} z \cong \begin{bmatrix} 0 \\ r \end{bmatrix}. \tag{4.2}$$

Let the $n \times p$ matrix K be the solution to the linear system $R^TK = E^T$. Under the transformation of variables $u = Rz$ and $v = r - Ez = r - K^Tu$, problem (4.2) becomes that of finding the minimum-norm solution to the underdetermined linear system

$$[K^T \quad I] \begin{bmatrix} u \\ v \end{bmatrix} = r. \tag{4.3}$$

Problem (4.3) can be solved by means of the orthogonal factorization

$$U \begin{bmatrix} K \\ I \end{bmatrix} = \begin{bmatrix} L^T \\ 0 \end{bmatrix}, \tag{4.4}$$

where U is an orthogonal matrix of order $n + p$ and L is a lower triangular matrix of order p. We use another transformation of variables

$$\begin{bmatrix} s \\ t \end{bmatrix} = U \begin{bmatrix} u \\ v \end{bmatrix},$$

so that (4.3) becomes

$$[L \quad 0] \begin{bmatrix} s \\ t \end{bmatrix} = r.$$

In this way s is determined by the linear system $Ls = r$, leaving t free to be chosen so as to minimize the norm. The choice $t = 0$ yields the desired result. We are thus led to the following algorithm for solving problem (4.1):

```
ALGORITHM 3
1. Perform steps 1 through 4 of Algorithm 1 on $[A, b]$.
2. Solve $Ry = c$.
3. Solve $R^T K = E^T$.
4. Compute the orthogonal factorization (4.4).
5. $r = f - Ey$.
6. Solve $Ls = r$.
7. $\begin{bmatrix} u \\ v \end{bmatrix} = U^T \begin{bmatrix} s \\ 0 \end{bmatrix}$.
8. Solve $Rz = u$.
9. $x = y + z$.
```

The sparse triangular factor computed in step 1 is used to solve all subsequent large linear systems required in this algorithm. Since the number p of updating rows is presumably small, the orthogonal factorization (4.4) required in step 4 may be carried out by Householder transformations or any other method suitable for small dense problems, resulting in a small triangular system to be solved in step 6.

Although this algorithm was motivated by the problem of withholding rows from the initial orthogonal factorization, it is, of course, also applicable in any instance when updating is required to incorporate new observations into a problem previously solved using Algorithm 1. In the latter case a second alternative is to use Givens rotations to process the new rows directly into the triangular factor, provided that the new rows add no new structure to A^TA and that the original transformed right-hand side vector c has been preserved.

5. Constraints. It is sometimes the case, due to theoretically known relationships among the variables or to a few unusually accurate measurements, that it is desirable that some of the equations in a linear system be satisfied exactly while the remaining equations are satisfied only in the least squares sense. Such a problem may be written in the form

$$\min_x \|Ax - b\|_2 \quad \text{subject to } Gx = h, \tag{5.1}$$

where G is a $q \times n$ matrix and h is a q-vector. We assume that A is of full column rank n and the constraint equations are consistent. There are a number of methods for solving such a linear equality constrained least squares problem:

1. using a basis of the null space of the constraint matrix G;
2. by direct elimination of variables using the constraint equations;
3. by weighting;
4. by the method of Lagrange multipliers.

The first three of these methods are described in Chapters 20, 21, and 22, respectively, of [17]; the fourth is described in [12]. The first two methods, when implemented using orthogonalization, generally involve transformations on the least squares matrix A which may cause unacceptable fill in the large sparse case. Better preservation of sparsity might be obtained by using Gaussian elimination, but then the pivoting generally required to maintain stability would bring on just the kind of data access problems Algorithm 1 is designed to avoid. The third method, weighting, is the simplest to implement since it requires only the solution of an unconstrained least squares problem of the form

$$\begin{bmatrix} wG \\ A \end{bmatrix} x \cong \begin{bmatrix} wh \\ b \end{bmatrix}, \tag{5.2}$$

where w is a scalar weighting parameter chosen so that the constraint equations are satisfied as accurately as desired. Provided G is also sparse, Algorithm 1 is applicable to problem (5.2), although, as with any problem having disparate row norms, the ordering of the rows may have to be chosen so as to maximize stability and accuracy. In this context, processing all the heavily weighted constraint equations first is usually adequate if the only disparity is due to the weight w [17]. It is not unusual, however, especially with constraints imposed for theoretical reasons, for G to consist of a small number of dense rows. For example, it may be required that the sum of all the variables be equal to 1 or some other prescribed constant, in which case G contains a completely full row. The method of Lagrange multipliers is very attractive when G is dense but q is small, and for this reason we develop a simplified variant in which the multipliers are not explicitly computed.

The method of Lagrange multipliers applied to problem (5.1) consists of introducing a q-vector λ of multiplier variables and solving the expanded linear system of order $n+q$

$$\begin{bmatrix} A^TA & G^T \\ G & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} A^Tb \\ h \end{bmatrix}. \tag{5.3}$$

Straightforward block elimination leads to an approach using normal equations to solve (5.3) for λ, then x, by means of the following sequence of steps:

1. Solve $A^TAy = A^Tb$ for y.
2. Solve $A^TAJ = G^T$ for the $n \times q$ matrix J.
3. $M = GJ$.
4. $r = h - Gy$.
5. Solve $M\lambda = r$. (Note that M is symmetric.)
6. $x = y + J\lambda$.

As usual, a numerically superior approach is provided by orthogonalization, and explicit computation of λ can thereby be avoided. Let R and c be as in (1.3) as computed by Algorithm 1. We first note that y in step 1 is simply the solution to the unconstrained least squares problem and is given by the solution to the triangular system $Ry = c$. Next, if we define the $n \times q$ matrix K as the solution to the linear system $R^TK = G^T$, then we have

$$M = GJ = G(A^TA)^{-1}G^T = G(R^TR)^{-1}G^T = K^TK.$$

Thus, we can avoid explicitly forming the symmetric matrix M by computing its Cholesky factor directly via the orthogonal decomposition

$$UK = \begin{bmatrix} L^T \\ 0 \end{bmatrix}, \tag{5.4}$$

where U is an orthogonal matrix of order n and L is a lower triangular matrix of order q. Once again, the small size q permits the use of Householder transformations in computing (5.4) since sparsity is not an issue there. We note that if G, hence K, is rank deficient (as will be the case, for instance, if the constraint equations are inconsistent), this can be detected at this stage by checking the condition of L.

In this way the linear system for λ in step 5 could be replaced by the system $LL^T\lambda = r$. We note, however, that λ is immediately multiplied by J in step 6 and that

$$J\lambda = (A^TA)^{-1}G^T\lambda = (R^TR)^{-1}R^TK\lambda = R^{-1}U^T\begin{bmatrix} L^T \\ 0 \end{bmatrix}\lambda = R^{-1}U^T\begin{bmatrix} s \\ 0 \end{bmatrix},$$

where we have defined $s = L^T\lambda$. Thus, we see that it is unnecessary to determine λ, but only to solve the system $Ls = r$. We therefore arrive at the following algorithm for solving problem (5.1):

```
ALGORITHM 4
1. Perform steps 1 through 4 of Algorithm 1 on $[A, b]$.
2. Solve $Ry = c$.
3. Solve $R^TK = G^T$.
4. Compute the orthogonal factorization (5.4).
5. $r = h - Gy$.
6. Solve $Ls = r$.
7. $u = U^T\begin{bmatrix} s \\ 0 \end{bmatrix}$.
8. Solve $Rz = u$.
9. $x = y + z$.
```

The great similarity between Algorithms 3 and 4 reflects the close relationship between these two updating problems. The only substantive difference is the matrix to be factorized in step 4. In particular, Algorithm 4 can be derived from Algorithm 3 by taking $E = wG$ and $f = wh$, as in (5.2), and letting $w \to \infty$. Alternatively, the transformation of variables technique used in deriving Algorithm 3 can also be used to obtain Algorithm 4, or, more generally, an algorithm for least squares problems having both kinds of update equations:

$$\begin{bmatrix} A \\ E \end{bmatrix} x \cong \begin{bmatrix} b \\ f \end{bmatrix} \quad \text{subject to } Gx = h, \tag{5.5}$$

where all matrices and vectors are as in (4.1) and (5.1). The resulting algorithm for problem (5.5) is as follows:

```
ALGORITHM 5
1. Perform steps 1 through 4 of Algorithm 1 on $[A, b]$.
2. Solve $Ry = c$.
3. Solve $R^TJ = E^T$ and $R^TK = G^T$.
4. Compute the orthogonal factorization $U\begin{bmatrix} J & K \\ I & 0 \end{bmatrix} = \begin{bmatrix} L^T \\ 0 \end{bmatrix}$.
5. $r = \begin{bmatrix} f - Ey \\ h - Gy \end{bmatrix}$.
6. Solve $Ls = r$.
7. $\begin{bmatrix} u \\ v \end{bmatrix} = U^T\begin{bmatrix} s \\ 0 \end{bmatrix}$.
8. Solve $Rz = u$.
9. $x = y + z$.
```

6. Square systems. Most popular algorithms for solving sparse nonsymmetric square ($m = n$) linear systems (e.g., the Harwell software package MA28 [8]) use row and column pivoting during the factorization process in order to preserve sparsity and ensure numerical stability, and therefore they employ a relatively high-overhead dynamic data structure. Since Algorithm 1 uses an efficient static data structure with the column ordering fixed in advance, it is natural to conjecture that Algorithm 1 might be a viable alternative for solving sparse square systems. One potential difficulty is the necessity of withholding any dense rows from the factorization in order to prevent excessive fill in the triangular factor. In our previous updating schemes we have assumed that the sparse part of the matrix is already of full column rank without the omitted rows. If any rows are omitted from a square system, however, the resulting system is underdetermined and therefore has rank less than n. The general problem of updating a rank-deficient system when the added rows may change the rank by an unknown amount is quite complicated and can be numerically delicate (see, e.g., [5]). If $n-k$ rows are omitted from a square nonsingular system, however, the rank of the remaining part must be exactly k and the update must restore full rank. This fortunate circumstance allows us to cope with this special case in a convenient way. We simply replace the $n-k$ omitted rows by $n-k$ judiciously chosen sparse rows which give us a square nonsingular system to which Algorithm 1 is applicable. The resulting solution is then modified to obtain the solution to the original problem. This updating could be done using the Woodbury formula [15] or, equivalently, the capacitance matrix method [3]. In the present context, however, we use a more direct and numerically superior approach.

Although the dense rows may appear anywhere in the system, for simplicity of notation we place them last. Thus, we wish to solve the linear system

$$\begin{bmatrix} A \\ E \end{bmatrix} x = \begin{bmatrix} b \\ f \end{bmatrix}, \tag{6.1}$$

where A is $k \times n$ and sparse, and E is $(n-k)\times n$ and dense. We assume that the entire system (6.1) is nonsingular and that $n-k$ is relatively small compared to n. We denote the $(n-k)\times n$ sparse matrix which replaces E in the initial orthogonal factorization by G.

The best possible choice of replacement rows in terms of sparsity preservation would be to select the $n-k$ rows of the identity matrix which yield a square nonsingular system, since this would add nothing to the structure of A^TA. The problem of which rows of the identity matrix to use is solved by the rank determination procedure in step 1 of Algorithm 2 applied to the k sparse rows of the system. With a suitable tolerance step 1 will result in a matrix having $n-k$ zero rows. The appropriate rows of the identity matrix may then be added to the system by simply placing a 1 in the diagonal entry of each of the $n-k$ zero rows. (This will in fact have already been done if the trick suggested in § 3 for avoiding a special back substitution routine is used. We also note that it may be desirable to use a scalar multiple of the identity matrix to improve conditioning if the scale of the problem differs significantly from unity.) With the matrix G consisting of the indicated rows of the identity matrix, step 1 of Algorithm 2 provides a factorization of the form

$$\begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} A \\ G \end{bmatrix} = \begin{bmatrix} R & S \\ 0 & I \end{bmatrix}, \quad \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} b \\ 0 \end{bmatrix} = \begin{bmatrix} c \\ 0 \end{bmatrix}, \tag{6.2}$$

where we have written the matrices in permuted and partitioned form for notational simplicity. Applying this transformation to system (6.1) yields

$$\begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} A \\ E \end{bmatrix} = \begin{bmatrix} R & S \\ E_1 & E_2 \end{bmatrix}, \quad \begin{bmatrix} Q & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} b \\ f \end{bmatrix} = \begin{bmatrix} c \\ f \end{bmatrix},$$

where $E = [E_1, E_2]$ and E_2 is square. Thus, we wish to solve the equivalent system

$$\begin{bmatrix} R & S \\ E_1 & E_2 \end{bmatrix} x = \begin{bmatrix} c \\ f \end{bmatrix}, \tag{6.3}$$

utilizing the available sparse factorization (6.2). Carrying out block elimination on (6.3) leads to the system

$$\begin{bmatrix} R & S \\ 0 & K_2^T \end{bmatrix} x = \begin{bmatrix} c \\ r \end{bmatrix}, \tag{6.4}$$

where the $(n-k) \times n$ matrix $K^T = [K_1^T, K_2^T]$ is given by the solution to the system

$$\begin{bmatrix} R^T & 0 \\ S^T & I \end{bmatrix} \begin{bmatrix} K_1 \\ K_2 \end{bmatrix} = \begin{bmatrix} E_1^T \\ E_2^T \end{bmatrix},$$

and

$$r = f - K_1^T c = f - E_1 R^{-1} c = f - E_1 y = f - E \begin{bmatrix} y \\ 0 \end{bmatrix}.$$

If we now let u be the solution to the square system $K_2^T u = r$ (computed by any suitable method, such as elimination or orthogonalization), we see that the solution to the system

$$\begin{bmatrix} R & S \\ 0 & I \end{bmatrix} x = \begin{bmatrix} c \\ u \end{bmatrix}$$

is also the solution to (6.4).

Reverting to our prior notation (with R representing the entire triangular factor) and organizing the computation to emphasize its similarity to Algorithm 3, we get the following algorithm for solving problem (6.1):

ALGORITHM 6

1. Perform step 1 of Algorithm 2 on $[A, b]$, then replace any zero diagonal entries in the resulting triangular factor with ones (effectively adding $[G, 0]$ to the system).
2. Solve $Ry = c$.
3. Solve $R^T K = E^T$ for the $n \times (n-k)$ matrix K, and let the square matrix K_2 consist of the rows of K corresponding to G.
4. Compute the orthogonal factorization $UK_2 = L^T$.
5. $r = f - Ey$.
6. Solve $Ls = r$.
7. $u = U^T s$.
8. Solve $Rz = \begin{bmatrix} 0 \\ u \end{bmatrix}$.
9. $x = y + z$.

Early computational experience with Algorithms 1 and 6 on a modest set of square sparse test problems indicates that this method does hold significant promise and is worthy of further development. Our approach seems to be reasonably competitive, usually requiring less storage but more execution time than MA28. Our algorithms perform especially well on problems having a very regular sparsity pattern, such as the matrices corresponding to geodetic networks or finite element grids, because of the availability of good heuristics for presorting the rows into an order which reduces the arithmetic operation count for the orthogonal factorization [9]. Due at least in part to the lack of more broadly effective row ordering schemes, our method has so far done less well on problems having more random sparsity patterns. Even in such cases, however, some of the other features enjoyed by our method, such as the convenience with which auxiliary storage can be used [10], may nevertheless make it an attractive alternative.

7. Conclusions. In this paper we have developed several algorithms for handling sparse linear least squares problems not covered by the basic algorithm of [9]. These algorithms retain, however, the fundamental spirit of the basic algorithm: the column ordering is fixed in advance to minimize fill and the rows may be processed in essentially arbitrary order. Each of the extended algorithms requires the solution of several linear systems of two types: large systems which are solved using the sparse triangular factor computed by the basic algorithm, and small systems which are solved using methods appropriate to the small dense case. For the latter systems we have emphasized the use of numerically stable orthogonalization techniques. All of the algorithms presented have been implemented in computer software and have been found to work well in practice. The results of extensive numerical testing and comparisons with other methods will be presented elsewhere.

There remain numerous questions relating to these algorithms and their possible further extensions. The difficult problem of general updating for rank-deficient problems has already been mentioned. (For a treatment of this problem in the context of a different basic algorithm, see [2].) The details of the tolerances and tests needed for numerical rank determination are not universally agreed upon. Further investigation is needed into the effect of the row ordering in Algorithm 1 with regard to efficiency and stability. A better understanding of this problem should lead to row ordering heuristics which are effective for broader classes of problems. It is obvious that dense rows must be withheld from the initial orthogonal factorization because of the fill they would cause. It may also be possible that significantly less fill would result from the omission of other, less obvious rows. The identification of such rows, however, appears to be a difficult problem. Other potential generalizations of our technique include the ability to handle inequality constraints, nondiagonal weight matrices (i.e., correlated observations), and total least squares (i.e., all variables subject to error).

Acknowledgments. The author is pleased to acknowledge helpful discussions of various aspects of the problems treated in this paper with Å. Björck, R. E. Funderlic, A. George, G. H. Golub, R. J. Plemmons, D. S. Scott, and R. C. Ward. One of the referees suggested substantial improvements to Algorithm 6 which led to its present form.

REFERENCES

[1] N. Anderson and I. Karasalo, *On computing bounds for the least singular value of a triangular matrix*, BIT, 15 (1975), pp. 1–4.
[2] Å. Björck and I. S. Duff, *A direct method for the solution of sparse linear least squares problems*, Linear Algebra Appl., 34 (1980), pp. 43–67.
[3] B. L. Buzbee, F. W. Dorr, J. A. George and G. H. Golub, *The direct solution of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.
[4] 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.
[5] R. E. Cline and R. E. Funderlic, *The rank of a difference of matrices and associated generalized inverses*, Linear Algebra Appl., 24 (1979), pp. 185–215.

[6] R. E. Cline and R. J. Plemmons, *l_2-solutions to underdetermined linear systems*, SIAM Rev., 18 (1976), pp. 92–106.

[7] P. Deuflhard and W. Sautter, *On rank-deficient pseudoinverses*, Linear Algebra Appl., 29 (1980), 91–111.

[8] I. S. Duff, MA28—*A set of Fortran subroutines for sparse unsymmetric linear equations*, Rep. AERE-R8730, Harwell, England, June 1977.

[9] A. George and M. T. Heath, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra Appl., 34 (1980), pp. 69–83.

[10] A. George, M. T. Heath and R. J. Plemmons, *Solution of large-scale sparse least squares problems using auxiliary storage*, this Journal, 2 (1981), pp. 416–429.

[11] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[12] G. H. Golub, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[13] G. H. Golub, V. Klema and G. W. Stewart, *Rank degeneracy and least squares problems*, Rep. STAN-CS-76-559, Computer Science Dept., Stanford Univ., Stanford CA, August 1976.

[14] G. H. Golub and V. Pereyra, *Differentiation of pseudoinverses, separable nonlinear least square problems and other tales*, in Generalized Inverses and Applications, M. Z. Nashed, ed., Academic Press, New York, 1976, pp. 303–324.

[15] H. V. Henderson and S. R. Searle, *On deriving the inverse of a sum of matrices*, SIAM Rev., 23 (1981), pp. 53–60.

[16] I. Karasalo, *A criterion for truncation of the QR-decomposition algorithm for the singular linear least squares problem*, BIT, 14 (1974), pp. 156–166.

[17] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[18] T. A. Manteuffel, *Numerical rank determination in linear least squares problems*, Rep. SAND79-8243, Sandia Laboratories, Livermore, CA, August 1979. (See also Reps. 80-0655 and 80-1260.)

[19] G. Peters and J. H. Wilkinson, *The least squares problem and pseudo-inverses*, Comput. J., 13 (1970), pp. 309–316.

[20] J. B. Rosen, *Minimum and basic solutions to singular linear systems*, SIAM J. Appl. Math., 12 (1964), pp. 156–162.

[21] A. van der Sluis and G. W. Veltkamp, *Restoring rank and consistency by orthogonal projection*, Linear Algebra Appl., 28 (1979), pp. 257–278.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

ON THE COMPUTATION OF OPTIMAL DESIGNS FOR CERTAIN TIME SERIES MODELS WITH APPLICATIONS TO OPTIMAL QUANTILE SELECTION FOR LOCATION OR SCALE PARAMETER ESTIMATION*

RANDALL L. EUBANK,† PATRICIA L. SMITH‡ AND PHILIP W. SMITH§

Abstract. Using the results of Chow (Ph.D. dissertation, Texas A & M Univ., 1978) on the optimal placement of knots in the approximation of functions by piecewise polynomials, we present an algorithm for the computation of optimal designs for certain time series models considered by Eubank, Smith and Smith (Ann. Statist., 9 (1981), pp. 486–493), (Tech. Rep. 150, Southern Methodist Univ., 1981). The ideas underlying this algorithm form a unified approach to the computation of optimal spacings for the sample quantiles used in the asymptotically best linear unbiased estimator of a location or scale parameter.

Key words. time series, optimal designs, splines, estimation, location parameter, scale parameter

1. Introduction.

Consider the linear regression model in which a stochastic process, Y, is observed having the form

$$Y(t) = \beta f(t) + X(t), \qquad t \in [0, 1], \tag{1.1}$$

where β is an unknown parameter, f is a known regression function and $X(\cdot)$ is a zero mean process with known covariance kernel, R. The X process is assumed to admit $k-1$ quadratic mean derivatives at each point $t \in [0, 1]$.

When the Y process is observed over all of $[0, 1]$, the reproducing kernel Hilbert space (RKHS) techniques developed by Parzen (1961a), (1961b) may be used to construct a linear unbiased estimator of the parameter β. We will denote this estimator by $\hat{\beta}$. For finite sampling schemes the regression design problem has been considered by Sacks and Ylvisaker (1966), (1968), (1970), Wahba (1971), (1974), and Eubank, Smith and Smith (1981a), (1981b). In the context of model (1.1), a regression design is a set of noncoincident points in $[0, 1]$. The problem of design selection is, therefore, one of choosing from among the members of the class of all $n+2$ point designs

$$D_n := \{(t_0, t_1, \cdots, t_{n+1}): 0 = t_0 < t_1 < \cdots < t_{n+1} = 1\},$$

where := means "is defined as."

It is assumed throughout this paper that it is possible to sample not only the Y process but its derivatives as well. Given $T \in D_n$, one can then consider the estimation of β by an estimator based on the observation set

$$Y_{k,T} = \{Y^{(i)}(t): t \in T, i = 0, \cdots, k-1\}.$$

In particular, generalized least squares may be utilized to obtain the best linear unbiased estimator (BLUE) of β using the observations $Y_{k,T}$. This estimator will be denoted by $\hat{\beta}_{k,T}$.

An optimal $(n+2)$-point design for model (1.1) is a $T^* \in D$ which satisfies

$$V(\hat{\beta}_{k,T^*}) = \inf_{T \in D_n} V(\hat{\beta}_{k,T}).$$

* Received by the editors January 15, 1981, and in final form December 7, 1981.

† Department of Statistics, Southern Methodist University, Dallas, Texas 75275. The research of this author was supported in part by the Office of Naval Research under contract N00014-75-C-0439.

‡ Department of Mathematical Sciences, Old Dominion University, Norfolk, Virginia 23508.

§ Department of Mathematical Sciences, Old Dominion University, Norfolk, Virginia 23508. The research of this author was supported in part by the U.S. Army Research Office under grant DAHC04-75-0816.

Problems pertaining to the existence of optimal designs can be handled as in Sacks and Ylvisaker (1966). Sufficient conditions for the uniqueness of optimal designs have been given by Wahba (1971) and Eubank, Smith and Smith (1981a), (1981b) for certain types of covariance kernels.

Unfortunately, it will not always be possible to sample the derivatives of the Y process. However, results regarding $\hat{\beta}_{k,T}$ are still useful in this event since, as noted by Wahba (1971),

$$\inf_{T\in D_{nk}} V(\hat{\beta}_T) \leqq \inf_{T\in D_n} V(\hat{\beta}_{k,T}) \leqq \inf_{T\in D_n} V(\hat{\beta}_T)$$

where $\hat{\beta}_T$ is the BLUE of β obtained without the use of derivative information, i.e., the generalized least squares estimator of β obtained from model (1.1) using the observation set $Y_T = \{Y(t): t \in T\}$. It should also be noted that, for the process considered here, the work of Barrow and Smith (1978) has the consequence that optimal designs for $\hat{\beta}_{k,T}$ are asymptotically optimal for $\hat{\beta}_T$. In addition, when $k = 2$ the optimal designs for $\hat{\beta}_{k,T}$ are, under certain conditions on f, precisely the optimal designs for $\hat{\beta}_T$ (cf. Theorem 2.3 of Eubank, Smith and Smith (1981a) and Theorem 2.2 of Eubank, Smith and Smith (1981b)). Of course, in the important case of $k = 1$ considered by Sacks and Ylvisaker (1966), $\hat{\beta}_{1,T} = \hat{\beta}_T$ and our work is also applicable to the regression design problem in this instance.

In this paper we continue the work of Eubank, Smith and Smith (1981a), (1981b) by constructing an algorithm for the computation of optimal designs for the case that R is the covariance kernel corresponding to a $(k-1)$-fold multiple integral of a Brownian bridge or Brownian motion process or certain generalizations of these processes. The case $k = 1$ corresponds to the Brownian bridge and Brownian motion covariance kernels and is of particular importance. In fact, a model of the form (1.1) with $X(\cdot)$ a Brownian bridge process has been shown by Parzen (1979) to arise in the estimation of a location or scale parameter by linear combinations of order statistics. It will be seen (§ 4) that our algorithm can be used, in conjunction with the work of Eubank (1981), to obtain a unified framework for optimal spacing selection for the quantiles utilized in the asymptotically best linear unbiased estimator of a location or scale parameter.

In § 2 we give some preliminary results regarding certain relationships between the selection of designs for model (1.1) and the approximation of functions by piecewise polynomials. By use of these relationships, it is possible to obtain an algorithm for optimal design computation through the modification of work by Chow (1978) on piecewise polynomial approximation with variable knots. The optimal design algorithm is presented in § 3 along with several illustrations of its use. Its application to location or scale parameter estimation is discussed in § 4. Section 5 contains a short summary.

2. Optimal designs and piecewise polynomial approximation. The covariance kernel, R, for the process (1.1) is the reproducing kernel for a reproducing kernel Hilbert space (RKHS) which will be denoted as $H(R)$ (cf. Parzen (1961a), (1961b)). The problem of optimal design selection for the estimator $\hat{\beta}_{k,T}$ may be formulated as a minimum norm approximation problem in $H(R)$ in the following manner. Let $\|\cdot\|_R$ denote the norm in $H(R)$ and define

$$S_{k,T} := \operatorname{span}\{R^{(0,j)}(\cdot, t): t \in T, j = 0, 1, \cdots, k-1\}$$

where

$$R^{(i,j)}(s, t) := \frac{\partial^{i+j}}{\partial s^i \, \partial t^j} R(s, t).$$

The orthogonal projector (with respect to $\|\cdot\|_R$), $\mathcal{P}_{k,T}$, which maps $H(R)$ onto $S_{k,T}$ has been shown by Sacks and Ylvisaker (1970) to satisfy

$$V(\hat{\beta}_{k,T}) = \|\mathcal{P}_{k,T}f\|_R^{-2}. \tag{2.1}$$

As $\mathcal{P}_{k,T}$ is an orthogonal projection, we have

$$\|\mathcal{P}_{k,T}f\|_R^2 = \|f\|_R^2 - \|f - \mathcal{P}_{k,T}f\|_R^2. \tag{2.2}$$

From (2.1) and (2.2) it follows that T^* is an optimal design if and only if

$$\|f - \mathcal{P}_{k,T^*}f\|_R = \inf_{T \in D_n} \|f - \mathcal{P}_{k,T}f\|_R. \tag{2.3}$$

Thus we see that the optimal design problem is equivalent to the nonlinear best approximation problem: Find $T^* \in D_n$ such that $s^* := \mathcal{P}_{k,T^*}f$ satisfies

$$\|f - s^*\|_R = \inf_{T \in D_n} \inf_{s \in S_{k,T}} \|f - s\|_R.$$

In order to study this problem more closely we now restrict our attention to a specific class of X processes and their corresponding covariance kernels. Let

$$K(s,t) := \int_0^1 \frac{(s-u)_+^{k-1}(t-u)_+^{k-1}}{(k-1)!^2}\,du, \qquad 0 \leqq s, \quad t \leqq 1, \tag{2.4}$$

where $(x)_+^r = x^r$ for $x \geqq 0$ and is 0 otherwise, and let $U(\cdot)$ denote the corresponding normal process, i.e., a $(k-1)$-fold multiple integral of Brownian motion. Define a new process, W, by

$$W(t) = \begin{cases} U(t) - E[U(t) | U^{(j)}(1), j = k-q, \cdots, k-1], & 0 < q \leqq k, \\ U(t), & q = 0. \end{cases} \tag{2.5}$$

It will be assumed in subsequent discussions that R is the covariance kernel defined by

$$R(s,t) := \operatorname{Cov}(W(s), W(t)). \tag{2.6}$$

When $q = 1$, R is the covariance kernel corresponding to a $(k-1)$-fold multiple integral of a Brownian bridge process whereas the case of $q = 0$ reverts to the covariance kernel (2.4) for a multiple integral of Brownian motion. The case of $q = k$ was considered by Eubank, Smith and Smith (1981a).

For processes with covariance kernels of the form (2.6), it is known that (cf. Eubank, Smith and Smith (1981a), (1981b)),

i) $H(R)$ is a Hilbert function space consisting of functions which satisfy for $f \in H(R)$,

$$f^{(j)} \text{ absolutely continuous}, \quad j = 0, \cdots, k-1, \quad \text{with } f^{(k)} \in L^2[0,1],$$

and boundary conditions

$$\begin{aligned} f^{(j)}(0) &= 0, \qquad j = 0, \cdots, k-1, \\ f^{(j)}(1) &= 0, \qquad j = k-q, \cdots, k-1, \quad \text{for } 0 < q \leqq k, \end{aligned}$$

or just $f^{(j)}(0) = 0$, $j = 0, \cdots, k-1$, for $q = 0$. The norm for $f \in H(R)$ is

$$\|f\|_R = \|f^{(k)}\|_{L^2} = \left\{ \int_0^1 (f^{(k)}(x))^2\,dx \right\}^{1/2}. \tag{2.7}$$

ii) When $q=k$, observations at 0 or 1 provide no information regarding the parameter β (this follows from property (i) since, in this case, $f^{(j)}(0)=f^{(j)}(1)=0$, $j=0,\cdots,k-1$). Therefore, only design points in the interior of $[0,1]$ are utilized when estimating β. This convention has the consequence that the designs to be selected here agree with those considered by Eubank, Smith and Smith (1981a). We note in passing that similar remarks about observation selection hold for other values of q (cf. Sacks and Ylvisaker (1966) for the case of $k=1$, $q=0$).

iii) $R(s,t)$, as a function of s for fixed t, is a spline of order $2k$ in continuity class C^{2k-2} with a knot at t.

iv) For $T=(t_0,t_1,\cdots,t_{n+1})\in D_n$, the best $L^2[0,1]$ approximation to $f^{(k)}$ from $P^k(T)$, the set of piecewise polynomials of order k with breakpoints at $t_1,\cdots,t_n$, is $(\mathcal{P}_{k,T}f)^{(k)}$.

Let $\mathcal{Q}_{k,T}$ denote the $L^2[0,1]$ orthogonal projector for $P^k(T)$. Then, equation (2.7) and result (iv) have the consequence that

$$\|f-\mathcal{P}_{k,T}f\|_R=\|f^{(k)}-(\mathcal{P}_{k,T}f)^{(k)}\|_{L^2}=\|f^{(k)}-\mathcal{Q}_{k,T}f^{(k)}\|_{L^2}$$

and, hence,

$$\inf_{T\in D_n}\|f-\mathcal{P}_{k,T}f\|_R=\inf_{T\in D_n}\|f^{(k)}-\mathcal{Q}_{k,T}f^{(k)}\|_{L^2}.$$

Therefore, in view of (2.3), the optimal design problem for the types of Y processes considered here coincides with finding the breakpoints of the best $L^2[0,1]$ piecewise polynomial approximation to $f^{(k)}$.

Approximation by piecewise polynomials with free breakpoints has been studied by Barrow et al. (1978), Barrow and Smith (1978) and Chow (1978). Their results, restated in the design setting, yield this partial characterization of optimal designs for covariance kernels of the form (2.6).

THEOREM 1. *Let* $T^*=(t_0^*,t_1^*,\cdots,t_{n+1}^*)\in D_n$ *an an optimal design. If* $f\in H(R)\cap C^{2k}[0,1]$ *and* $f^{(2k)}>0$ *on* $(0,1)$ *then, for* $i=1,\cdots,n$,

$$(2.8)\qquad (\mathcal{P}_{k,T^*}f)^{(k)}(t_i^*-)=\begin{cases}(\mathcal{P}_{k,T^*}f)^{(k)}(t_i^*+), & k\ \textit{even},\\ 2f^{(k)}(t_i^*)-(\mathcal{P}_{k,T^*}f)^{(k)}(t_i^*+), & k\ \textit{odd}.\end{cases}$$

Sometimes the necessary condition (2.8) is also sufficient to guarantee an optimal design. We state such a result from Eubank, Smith and Smith (1981a), (1981b).

THEOREM 2. *Let* $f\in H(R)\cap C^{2k}[0,1]$ *with* $f^{(2k)}>0$ *on* $[0,1]$ *and* $\log f^{(2k)}$ *concave on* $(0,1)$. *Then* $\hat\beta_{k,T}$ *has a unique optimal design for each* n.

In general, uniqueness is quite difficult to prove. At present, very few other positive results concerning uniqueness are available.

3. An algorithm for computing optimal designs. Theorem 1 suggests that we should find a design $T=(t_0,\cdots,t_{n+1})$ for which $F_i(T)=0$, $i=1,\cdots,n$, where

$$(3.1)\qquad F_i(T)=\begin{cases}(\mathcal{P}_{k,T}f)^{(k)}(t_i+)-(\mathcal{P}_{k,T}f)^{(k)}(t_i-), & k\ \text{even},\\ 2f^{(k)}(t_i)-(\mathcal{P}_{k,T}f)^{(k)}(t_i-)-(\mathcal{P}_{k,T}f)^{(k)}(t_i+), & k\ \text{odd}.\end{cases}$$

Thus, setting $F(T)=(F_1(T),\cdots,F_n(T))^t$, we see that we are looking for a zero of the vector valued function F. Such zeros will be candidates for optimal designs. Chow

(1978) has shown that for $i=1,\cdots,n$,

$$\begin{aligned}F_i(T)=&\frac{(t_i-t_{i-1})^k}{(k-1)!}\int_0^1(1-\tau)^{k-1}\tau^k f^{(2k)}[t_{i-1}+\tau(t_i-t_{i-1})]\,d\tau\\&-\frac{(t_{i+1}-t_i)^k}{(k-1)!}\int_0^1(1-\tau)^k\tau^{k-1}f^{(2k)}[t_i+\tau(t_{i+1}-t_i)]\,d\tau,\end{aligned}\tag{3.2}$$

where we recall $t_0:=0$ and $t_{n+1}:=1$. Consequently, the Jacobian matrix of F at T, $A(T):=[(\partial F_i/\partial t_j)(T)]$, is tridiagonal with nonzero elements given by

$$\frac{\partial F_i}{\partial t_{i-1}}=-\frac{k(t_i-t_{i-1})^{k-1}}{(k-1)!}\int_0^1(1-\tau)^k\tau^{k-1}f^{(2k)}[t_{i-1}+\tau(t_i-t_{i-1})]\,d\tau,\qquad 2\leqq i\leqq n,\tag{3.3}$$

$$\begin{aligned}\frac{\partial F_i}{\partial t_i}=&\frac{(t_i-t_{i-1})^{k-1}}{(k-1)!}\int_0^1(1-\tau)^{k-2}\tau^k(k\tau-1)f^{(2k)}[t_{i-1}+\tau(t_i-t_{i-1})]\,d\tau\\&+\frac{(t_{i+1}-t_i)^{k-1}}{(k-1)!}\int_0^1(1-\tau)^k\tau^{k-2}(k(1-\tau)-1)f^{(2k)}[t_i+\tau(t_{i+1}-t_i)]\,d\tau,\\&\qquad 1\leqq i\leqq n,\end{aligned}\tag{3.4}$$

$$\frac{\partial F_i}{\partial t_{i+1}}=-\frac{k(t_{i+1}-t_i)^{k-1}}{(k-1)!}\int_0^1(1-\tau)^{k-1}\tau^k f^{(2k)}[t_i+\tau(t_{i+1}-t_i)],\qquad 1\leqq i\leqq n-1.\tag{3.5}$$

When f is $2k$ times continuously differentiable and $f^{(2k)}>0$ we can use Newton's method to find a $T^*\in D_n$ which is a zero of F. Such a T^* will be an optimal design candidate and may be constructed using the algorithm presented below. If, in addition, f satisfied the conditions of Theorem 2, then the T^* located by the algorithm will be the optimal design.

ALGORITHM
Step 1. Select an initial $T=(t_0,\cdots,t_{n+1})$.
Step 2. Check to insure that $T\in D_n$.
Step 3. Compute $F(T)$ and $A(T)$.
Step 4. Compute $b=A(T)^{-1}F(T)$.
Step 5. Stop if b is small or the maximum number of iterations has been met.
Step 6. Set $t_i=t_i-b_i$, $i=1,\cdots,n$, and return to Step 2.

As was indicated above, the algorithm (when it converges) finds a design T^* which satisfies a necessary condition for design optimality. In order to enhance our chances of finding a "good" design, care should be taken in Step 1. An initial design choice which usually yields good results is the nth element of an *asymptotically optimal design sequence* (cf. Sacks and Ylvisaker (1966)). Such a sequence can be constructed using the density

$$h(x)=|f^{(2k)}(x)|^{2/2k+1}\Big/\int_0^1|f^{(2k)}(s)|^{2/2k+1}ds\tag{3.6}$$

in the following manner. Let H denote the distribution function corresponding to h with associated inverse (or quantile) function H^{-1}. Then it can be shown (cf. Eubank, Smith and Smith (1981a), (1981b) and/or Sacks and Ylvisaker (1966), (1970)) that the nth element of an asymptotically optimal design sequence for $\hat\beta_{k,T}$ consists of the points

(3.7) $$t_i = H^{-1}\left(\frac{i}{n+1}\right), \qquad i = 0, \cdots, n+1.$$

The design sequence $\{T_n\}_{n=1}^{\infty}$ obtained by solving (3.7) for successive values of n is asymptotically optimal in the sense that

(3.8) $$\lim_{n\to\infty} \frac{V(\hat{\beta}_{k,T_n}) - V(\hat{\beta})}{\inf_{T\in D_n} V(\hat{\beta}_{k,T}) - V(\hat{\beta})} = 1.$$

Although this relationship between optimal and asymptotically optimal designs pertains to large n, it is often the case (as will be discussed in the examples) that the asymptotics carry over to small n at least to the extent that the values of asymptotically optimal designs provide a good indication of the locations of the optimal design points.

In some cases H^{-1} has a closed form making asymptotically optimal designs easy to compute. However, even when this is not the case, H^{-1} can be readily evaluated through numeric tabulation of H and subsequent interpolation.

If after one or more iterations the check in Step 2 fails, this indicates that the algorithm has moved out of the feasible region, D_n. Such an occurrence is usually indicative of a poor choice for an initial design. In this event one alternative, of course, is to simply reinitialize with another design and try again. Alternatively, one might reduce the size of the step taken in Step 6, i.e., take

(3.9) $$t_i = t_i - \alpha b_i$$

for some $0<\alpha<1$. More generally, a modified version of the algorithm could be utilized, where for instance, after the ith iteration the new design points are taken as

(3.10) $$t_j = t_j - \alpha\delta_i b_j, \qquad j = 1, \cdots, n,$$

for some $0<\alpha<1$ where $0<\delta_i\leqq 1$ is the largest value such that the resulting design remains in D_n. We have not tested this last modification since it would only reduce convergence time and since, for all the problems we have considered, convergence has always occurred (in terms of a relative change in the design points between successive iterations of less than 10^{-11}) after only 4 to 7 iterations using an asymptotically optimal initial design.

The integrals computed in Step 3 of the algorithm will usually require evaluation by numerical methods. This can be readily accomplished through the use of a Gaussian quadrature rule.

We conclude this section by presenting several examples which illustrate the use of the algorithm and Theorems 1 and 2 in the computation of optimal designs. For simplicity the X process implicit in each of the following examples is taken to have covariance kernel (2.4). When exhibiting a particular design we present only those values which are in the interior of [0, 1].

Example 1. Consider first the case of

(3.11) $$f(t) = \frac{1}{6!}t^6.$$

This regression function furnishes an example of a function which satisfies the conditions of Theorems 1 and 2 and, hence, in this case for $k = 1, 2, 3$, we are assured of unique optimal designs for all values of n. In addition, through the use of this function it will be possible to examine instances when the use of uniformly spaced design points, either for estimation or for initialization of the algorithm, is a sound strategy, as well as instances when it is not. We now consider the construction and properties of various designs for this regression model.

In computing optimal designs for this model, asymptotically optimal designs were utilized as starting values. For a regression function having the form (3.11) the asymptotically optimal designs are simple to compute since the H^{-1} function is given by

$$H^{-1}(x)=\begin{cases} x^{3/11} & \text{when } k=1,\\ x^{5/9} & \text{when } k=2,\\ x & \text{when } k=3.\end{cases}$$

It is important to note that for $k=3$, the asymptotically optimal designs consist of uniformly spaced design points. These are, in fact, seen to be the optimal designs. Therefore, in the case of $f(t)=(1/6!)t^6$ with $k=3$ sampling the Y process at uniform intervals is not only sound but an optimal strategy. This is not the case, however, for $k=1$ and 2.

The optimal designs of size $n=1, 3, 5, 10, 20$ for this regression model were computed for $k=1, 2, 3$. The variances of $\hat{\beta}_{k,T}$ corresponding to these designs are presented in Table 1 along with, for comparison, the variances obtained through the use of asymptotically optimal and uniform designs. These values may be compared, with regret, to the values of $V(\hat{\beta})=\|f^{(k)}\|_{L^2}^{-2}$ provided at the bottom of the table.

Examination of Table 1 reveals that, as one might suspect, the substantive gains from the use of optimal (as opposed to asymptotically optimal) designs occur for small

TABLE 1
Variance of $\hat{\beta}_{k,T}$ for various designs when $f(t)=(1/6!)t^6$

k:	1			2			3
n	Optimal designs	Asymptotically optimal designs	Uniform designs	Optimal designs	Asymptotically optimal designs	Uniform designs	Optimal designs
1	184555.05063	185382.42720	267426.49874	5224.46233	5225.98932	52838.98710	252.00984
3	163646.59657	163765.35862	184532.68263	5186.20258	5186.27201	5190.99885	252.00015
5	160614.32569	160651.69363	169799.77924	5184.41623	5184.42633	5185.41386	252.00001
10	159033.14199	159039.58083	161751.23364	5184.03539	5184.03592	5184.12677	252*
20	158570.29681	158571.25684	159315.97500	5184.00260	5184.00262	5184.00958	252†
		$V(\hat{\beta})=158440$			$V(\hat{\beta})=5184$		$V(\hat{\beta})=252$

* Agrees with $V(\hat{\beta})$ to 6 decimals. † Agrees with $V(\hat{\beta})$ to 8 decimals.

n and/or k. Asymptotically optimal designs perform quite well in this case even for relatively small n over all values of k. In contrast, for $k=1$, uniformly spaced design points tend to perform poorly, relative to optimal or asymptotically optimal designs, especially for small n. The use of uniform designs would seem acceptable for large n when $k=2$ and, of course, the uniform, asymptotically optimal and optimal designs all agree when $k=3$.

Through the use of asymptotically optimal designs to initialize the algorithm it was possible to obtain convergence to the optimal designs, in every instance, in 5 or fewer iterations. However, uniformly spaced design points tried as starting values resulted in a failure of the check in Step 2 of the algorithm for $n \geqq 10$ when $k = 2$ and even for n as small as 3 when $k = 1$. In the instance of $k = 2$ it was found that a step of size $\alpha = .29$ in (3.9) could give convergence (to the extent of 5 digit accuracy) after 41 iterations. (Values of $\alpha > .3$ were all apparently too large to provide similar results.) No such value of α could be found when $k = 1$. Keeping in mind the criterion (2.8) that is utilized for locating an "optimal" design point, this latter fact comes as no surprise when one compares, for instance, the uniform 3-point design (.25, .5, .75) with the optimal design (.65828, .81674, .92042) for this case.

Example 2. As an example of a regression function which is not a polynomial, we now suppose f has the form

$$f(t) = \frac{8}{105} t^{7/2},$$

where the factor 8/105 is introduced to simplify subsequent numerical presentations. Both Theorems 1 and 2 are applicable to this function when $k = 1$ but not when $k = 2$ or 3 as, in these latter cases, $f^{(2k)}$ is not in $C[0, 1]$. Consequently, this will provide an illustration of the performance of the algorithm under conditions other than those of Theorem 1 or the ideal conditions of Theorem 2.

As in the previous example, the H^{-1} function has a closed form. In this case,

$$H^{-1}(x) = \begin{cases} x^{1/2} & \text{when } k = 1, \\ x^{5/4} & \text{when } k = 2, \\ x^{7/2} & \text{when } k = 3. \end{cases}$$

The variances of $\hat{\beta}_{k,T}$ corresponding to optimal, asymptotically optimal and uniformly spaced designs of size $n = 1, 3, 5, 10, 20$ are presented in Table 2 for $k = 1$, 2, 3. As in the previous example, optimal designs were computed by use of asymptotically optimal starting values. In all instances convergence occurred after at most 7 iterations. Examinations of the values in Table 1 lead, again, to the conclusion that the use of optimal designs (rather than asymptotically optimal designs) will be of the most value when n is small. It should be remembered, however, that in contrast to

TABLE 2
Variance of $\hat{\beta}_{k,T}$ for various designs when $f(t) = (8/105)t^{7/2}$

k:	1			2			3		
n	Optimal designs	Asymptotically optimal designs	Uniform designs	Optimal designs	Asymptotically optimal designs	Uniform designs	Optimal designs	Asymptotically optimal designs	Uniform designs
1	94.98829	95.12777	102.67984	9.00467	9.00471	9.00532	2.00009	2.00018	2.00039
3	86.63789	86.65974	88.60695	9.00031	9.00031	9.00041	2*	2.00001	2.00010
5	85.34369	85.35069	86.22809	9.00006	9.00006	9.00009	2*	2*	2.00004
10	84.65509	84.65631	84.92172	9.00001	9.00001	9.00001	2†	2†	2.00001
20	84.45077	84.45096	84.52462	9*	9*	9*	2†	2†	2*
	$V(\hat{\beta}) = 84.375$			$V(\hat{\beta}) = 9$			$V(\hat{\beta}) = 2$		

* Agrees with $V(\hat{\beta})$ to 6 decimals. † Agrees with $V(\hat{\beta})$ to 7 or more decimals.

the results presented in Table 1, those given in Table 2 were obtained, for $k=2$, 3 when Theorem 1 is not applicable. This suggests that the algorithm may still perform well under moderate departures from the assumptions upon which it is based.

Example 3. Finally, consider the case of f having the form

$$f(t)=(t-\tfrac{1}{2})^8-(\tfrac{1}{2})^8+8(\tfrac{1}{2})^7t. \tag{3.12}$$

This regression function provides an example of optimal design duplicity, as well as an illustration of the sensitivity some functions exhibit with regard to the selection of an initial design.

Using $n=1$ with $k=2$, one finds that $V(\hat{\beta}_{2,T})$ has a local maximum (rather than minimum) at $T^0=\{.5\}$ with $V(\hat{\beta}_{2,T^0})=.74681$. There are two optimal designs $T^1=\{.23079\}$ and $T^2=\{.76921\}$ where $V(\hat{\beta}_{2,T^1})=V(\hat{\beta}_{2,T^2})=.72007$. Choices of starting values such as .6 or .4 lead to convergence to T^0 whereas, for instance, the choices .2 and .8 results in convergence to T^1 and T^2 respectively. It should be noted that T^0 is also the uniform and asymptotically optimal design for this case. We therefore have an instance when the use of either uniform or asymptotically optimal (in lieu of optimal) designs is not only a poor, but is in fact the worst, strategy.

The function (3.12) has also been considered in Bock (1976) where the graph of $\|f-\mathcal{P}_{2,T}f\|_R$ versus t_1 is seen to have a "W-shape." Although there is little difference, in this case, between the variance at the local maximum and at the two minima, it is clear that functions may be constructed for which this difference is arbitrarily large.

In the examples, we have considered only regression functions for which the H^{-1} function has a closed form. This has been for the sake of illustration and has the consequence that the asymptotically optimal designs perform better, for estimation purposes, than might otherwise be the case. Unfortunately, situations where H^{-1} is of a closed form are rare in practice. The reader is referred to Eubank (1979) for several examples of the evaluation of H^{-1} by numerical methods.

All the computations in this and subsequent sections were performed on either the IBM 360 computer at Arizona State University or the CDC 6600 at Southern Methodist University.

4. Application to location or scale parameter estimation. Suppose a random sample, $Z_1, \cdots, Z_N$ is obtained from a distribution of the form $F(z)=F_0(z/\beta)$, where F_0 is a known distributional form and β is an unknown scale parameter. F_0 is assumed to be absolutely continuous with associated probability density function f_0. Let $Q_0(t):=F_0^{-1}(t)$ and define the density-quantile function as $d_0(t):=f_0(Q_0(t))$, $0\leqq t\leqq 1$. The sample quantile function is defined by $\tilde{Q}(t)=Z_{(j)}$, $(j-1)/N<t\leqq j/N$, $j=1,\cdots\cdot, N$, where $Z_{(j)}$ denotes the jth sample order statistic.

Parzen (1979) has shown that, for N sufficiently large, a model for scale parameter estimation is

$$d_0(t)\tilde{Q}(t)=\beta d_0(t)Q_0(t)+\sigma X(t), \qquad t\in[0,1], \tag{4.1}$$

where $\sigma=\beta/\sqrt{N}$ and $X(\cdot)$ is a Brownian bridge process. Eubank (1981) has shown that the problem of optimal design selection for model (4.1) is identical to the problem of selecting an optimal spacing for the sample quantiles utilized in constructing the asymptotically best linear unbiased estimator (ABLUE) of β (cf. Sarhan and Greenberg (1962) for discussions and examples of the more classical approach to the optimal spacing problem).

Given a design $T\in D_n$ (classically referred to as a *spacing* in the context of this problem) the ABLUE of β, $\hat{\beta}_T$, is, in fact, the corresponding generalized least squares estimator of β formed from model (4.1). Since, for a Brownian bridge process, $k = q$, it follows from property (ii) in § 2 that only those observations which correspond to the design points, $t_1, \cdots, t_n$, in the interior of $[0, 1]$ are used for estimation. Thus, $\hat{\beta}_T$ is of the form $\sum_{i=1}^{n} c(t_i)d_0(t_i)\tilde{Q}(t_i)$ where explicit expressions for the $c(t_i)$ can be found in Sarhan and Greenberg (1962). The *optimal spacing problem* consists of finding a spacing (design) for which the variance of $\hat{\beta}_T$ is a minimum or, equivalently,

$$\text{ARE}(\hat{\beta}_T) = V(\hat{\beta})/V(\hat{\beta}_T), \tag{4.2}$$

the relative efficiency of $\hat{\beta}_T$ with respect to $\hat{\beta}$, is a maximum. Equation (4.2) can be shown to provide the asymptotic (as $N \to \infty$) relative efficiency of the ABLUE with respect to the maximum likelihood estimator of β (cf. Eubank (1981)), which indicates the reason underlying the use of the ARE notation.

Upon examination of (4.1) there may appear to be a disparity between this model and the regression model (1.1) (and, consequently, between the optimal spacing and optimal design problems) due to the factor σ which appears in (4.1). The presence of this term implies that the variance of $\hat{\beta}_T$ is not $\|\mathcal{P}_{1,T}d_0\|_R^{-2}$ as in (2.1) but rather $\sigma^2\|\mathcal{P}_{1,T}d_0\|_R^{-2}$. However, σ^2, although unknown, is independent of T. Consequently, to minimize the variance of $\hat{\beta}_T$ it suffices to minimize $\|\mathcal{P}_{1,T}d_0\|_R^{-2}$ and the optimal spacing problem is therefore equivalent to the optimal design problem discussed in the previous two sections.

At present the literature on optimal spacings is composed of numerous articles cataloging the optimal spacings for various distribution types (cf. Eubank (1981) for a list of references). Thus the classical approach to the optimal spacing problem has been to consider each distribution separately. As the Brownian bridge process corresponds to the special case of k and q both equal to 1, it now follows that the algorithm presented in the previous section may also be used for the computation of optimal spacings. Two important consequences of this fact are:

(i) Model (4.1) in conjunction with Theorems 1 and 2 and the algorithm of § 3 provide the first simple, unified framework for the computation of optimal spacings.

(ii) Through reference to the optimal spacing literature, comparisons may be made between designs (spacings) obtained from the algorithm and those computed by other authors using the classical approach which involves a search using global optimization for each distribution.

It is important to note that, due to the particular characteristics of a distribution, it is sometimes possible to show uniqueness for optimal spacings when Theorem 2 is not applicable. Such results may be helpful in providing an indication of how our algorithm will perform under nonideal conditions. We illustrate this and the other comments with an example.

Let F_0 be the distribution function for the Pareto distribution, i.e.,

$$F_0(x) = 1 - (1+x)^{-\nu}, \qquad x, \nu > 0.$$

In this case

$$d_0(t)Q_0(t) = \nu[(1-t) - (1-t)^{(\nu+1)/\nu}] \tag{4.3}$$

and

$$[d_0(t)Q_0(t)]'' = -\left(1+\frac{1}{\nu}\right)(1-t)^{(1-\nu)/\nu}. \tag{4.4}$$

Theorem 2 is therefore applicable and insures a unique optimal design when $\nu \leqq 1$. The unique optimal spacings for $\nu = .5$ obtained from the algorithm with $n = 1, 3, 7$

TABLE 3
Optimal spacings (designs) for the Pareto, $\nu = .5, 2, k = 1$

n:	1		3		7	
	$\nu = .5$	$\nu = 2$	$\nu = .5$	$\nu = 2$	$\nu = .5$	$\nu = 2$
t_1	.35961	.61809	.16295	.34049	.07805	.17865
t_2			.35048	.63042	.16078	.34519
t_3			.58405	.85886	.24934	.49868
t_4					.34549	.63789
t_5					.45215	.76118
t_6					.57488	.86617
t_7					.72776	.94889
ARE$(\hat{\beta}_T)$	.77461	.72136	.94657	.92597	.98700	.98088

are presented in Table 3 and agree with those obtained by Kulldorf and Vännman (1973) using global optimization methods. Also given in Table 3 are the results obtained from the algorithm when $\nu = 2$. Even though neither Theorem 1 nor 2 applies, it is still true (cf. Kulldorf and Vännman (1973)) that the optimal spacings for the Pareto are unique in this case as well. The fact that the spacings computed by the algorithm agree with the optimal spacings for $\nu = 2$ given by Kulldorf and Vännman (1973) is an important illustration of the fact that unique optimal designs exist for a wider class of functions than those satisfying the hypotheses of Theorems 1 and 2 and, in such instances, may be computed with this algorithm.

If, instead of scale parameter estimation, location parameter estimation is of interest, the distribution function has the form $F(z) = F_0(z - \beta)$. A model similar to (4.1) holds in this case as well. To obtain an algorithm for optimal spacing computation in this instance, it is only necessary to interchange the roles of d_0 and $d_0 \cdot Q_0$ in the previous discussion.

The example presented in this section illustrates how the algorithm presented in § 3 may be used for optimal spacing computation, and, in addition, provides an indication of how it performs under departures from the "ideal conditions" of Theorem 1 or 2. For these reasons it has been useful to consider a situation where the optimal spacings had been obtained by other methods and were, therefore, available for comparison purposes. However, the value of this algorithm to the practitioner will lie in its use for the computation of the optimal spacings in situations which have not been considered in the literature and for which existing results are not available. It is our belief, based on comparison with the classical results, that this algorithm will be a valuable tool for this purpose even under moderate departures from the conditions of Theorem 1 or 2. We also conjecture that, since the algorithm is based on the local behavior of the d (or $d \cdot Q$) function near an optimal spacing element, optimal spacings can be obtained more rapidly and efficiently through the use of this method rather than an *ad hoc* global optimization technique. Unfortunately, the computational aspects of optimal spacing construction are typically not reported in the literature on the subject, and consequently it is difficult to obtain comparisons which support this contention.

5. Summary and discussion. In this paper an algorithm has been presented for the computation of optimal designs for certain time series models. This algorithm locates a design which satisfies a necessary condition for optimality provided $f^{(2k)}$ is continuous and of one sign on [0, 1]. If, in addition, $\log f^{(2k)}$ is concave on (0, 1) the use of this algorithm should provide the optimal design. The algorithm has also been shown to be useful in the selection of order statistics for location or scale parameter estimation. The advantage of this approach to spacing selection over classical techniques is that it provides a unified approach to optimal spacing selection which obviates the need for global optimization.

Experience with this algorithm indicates that it works rather well even when the conditions of Theorem 1 are only approximately satisfied (e.g., Example 2 of § 3 and the case of $\nu = 2$ in § 4). However, it may be more sensitive in such cases to the choice of initial designs. While uniformly spaced starting values are easily input and may produce the optimal design, they can also give poor or misleading results. Generally, better results may be obtained by initializing with an asymptotically optimal design, and, consequently, this method is recommended even though one must begin by evaluating the function H^{-1} as in (3.7).

REFERENCES

D. L. Barrow, C. K. Chui, P. W. Smith and J. D. Ward (1978), *Unicity of best mean approximation by second order splines with variable knots*, Math. Comp., 32, pp. 1131–1143.

D. L. Barrow and P. W. Smith (1978), *Asymptotic properties of best $L^2[0, 1]$ approximation by splines with variable knots*, Quart. Appl. Math., 36, pp. 293–304.

T. L. Bock (1976), *Comments on local minima arising from variable knot linear L2 splines*, Tech. Rep. 217, Dept. of Electrical Engineering and Computer Science, Princeton Univ., Princeton, NJ.

J. Chow (1978), *Uniqueness of best $L^2[0, 1]$ approximation by piecewise polynomials with variable breakpoints*, Ph.D., dissertation, Dept. of Mathematics, Texas A & M Univ., College Station.

R. L. Eubank (1979), *A density-quantile function approach to choosing order statistics for the estimation of location and scale parameters*, Tech. Rep. A10, Institute of Statistics, Texas A & M Univ., College Station.

———, (1981), *A density-quantile function approach to optimal spacing selection*, Ann. Statist., 9, pp. 494–500.

R. L. Eubank, P. L. Smith and P. W. Smith (1981a), *Uniqueness and eventual uniqueness of optimal designs in some time series models*, Ann. Statist., 9, pp. 486–493.

———, (1981b), *Uniqueness and eventual uniqueness of optimal designs in some time series models, II*, Tech. Rep. 150, Southern Methodist Univ., Dallas, TX.

G. Kulldorf and K. Vännman (1973), *Estimation of the location and scale parameter of the Pareto distribution by linear functions of order statistics*, J. Amer. Statist. Assoc., 68, pp. 218–227.

E. Parzen (1961a), *An approach to time series analysis*, Ann. Math. Statist., 32, pp. 951–989.

———, (1961b), *Regression analysis for continuous parameter time series*, in Proc. 4th Berkeley Symposium on Mathematics, Statistics and Probability, vol. I, Univ. of California Press, Berkeley, pp. 469–489.

———, (1979), *Nonparametric statistical data modeling*, J. Amer. Statist. Assoc., 74, pp. 105–131.

J. Sacks and D. Ylvisaker (1966), *Designs for regression problems with correlated errors*, Ann. Math. Statist., 37, pp. 66–89.

———, (1968), *Designs for regression problems with correlated errors: many parameters*, Ann. Math. Statist., 39, pp. 46–69.

———, (1970), *Designs for regression problems with correlated errors III*, Ann. Math. Statist., 41, pp. 2057–2074.

A. E. Sarhan and B. G. Greenberg (1962), *Contributions to Order Statistics*, John Wiley, New York.

G. Wahba (1971), *On the regression design problem of Sacks and Ylvisaker*, Ann. Math. Statist., 42, pp. 1035–1053.

———, (1974), *Regression design for some equivalence classes of kernels*, Ann. Statist., 2, pp. 925–934.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 2, June 1982

OUTFLOW BOUNDARY CONDITIONS FOR FLUID DYNAMICS*

ALVIN BAYLISS† AND ELI TURKEL‡

Abstract. A radiation boundary condition is derived for the Euler equation linearized about a constant state with a mean flow. Since nonlinearities and viscosity are not important in the far field, this boundary condition is also useful for high Reynolds number Navier–Stokes flow. The use of the radiation boundary condition allows both an acceleration to a steady state and a constriction in the size of the computational domain. This results in savings in both computer storage and running times. Results are presented for both the Navier–Stokes and Euler equations. A variety of schemes have been used in conjunction with the boundary condition. These include explicit and implicit finite difference schemes and spectral methods. The effectiveness of the radiation condition is evident in all these cases.

Key words. acceleration of convergence, exterior regions, fluid dynamics, radiation boundary conditions, subsonic outflow boundary.

1. Introduction. The numerical computation of steady-state fluid flows in exterior regions is frequently accomplished by integrating the time-dependent equations until a steady state is achieved. Viscosity effects are generally important only in localized regions, e.g., boundary layers. Thus the equations of fluid dynamics are basically hyperbolic in most regions of space and so admit wave-like solutions. It follows that a steady state can be achieved only by the propagation of energy outside the region of interest. In many problems this is accomplished by the radiation of energy to infinity.

In order to numerically solve a problem that is posed in an exterior region, it is usual to simulate the problem in a finite computational region. One alternative is to map the exterior region into a finite domain. This can create substantial errors, however, since one can not resolve the waves near infinity [6]. Instead, one usually computes in the physical domain and introduces artificial boundaries so that the computational domain is finite. When the flow is subsonic normal to the artificial surface, boundary conditions must be imposed along the artificial surface [12]. These boundary conditions should simulate the propagation of waves out of the computational domain. When this is not done, spurious reflected waves can be generated at the artificial boundaries. The resultant influx of energy into the computational region can delay convergence to a steady state and also degrade the accuracy of the steady-state solution. To preserve the accuracy, it is often necessary to place the artificial surface far from the region of interest. This results in a significant increase in both computer storage and running time.

Rudy and Strikwerda [14] developed a nonreflecting boundary condition by analyzing a simplified one-dimensional model. There is a free parameter which was chosen, for the one-dimensional case, to optimize the rate of convergence. For the two-dimensional problems, the parameter was chosen by computational experimentation. In some cases the use of their boundary condition led to dramatic accelerations in the rate of convergence to a steady state [14], [15].

* Received by the editors August 27, 1980, and in revised form March 3, 1981. This research was supported jointly by the U.S. Department of Energy under grant DOE-EY-76-C-02-3077 and the U.S. Air Force under contract 76-2881, and also in part by the National Aeronautics and Space Administration under contracts NAS1-14101 and NAS1-15810 while the authors were in residence at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, Virginia 23665.

† Courant Institute of Mathematical Sciences, New York University, New York, New York 10012.

‡ University of Tel-Aviv, Tel Aviv, Israel.

In this study, a boundary condition is developed which utilizes the structure of the outgoing waves in either two or three space dimensions. This condition is a generalization of one introduced in [2] for the wave equation. This boundary condition results from matching the solution to a functional form which is an approximation to an outgoing wave. It is easily shown that in the far field the fluid dynamics equations reduce to a convective wave equation. Hence, one can construct boundary conditions for the Euler equations which approximate outgoing waves. Since the viscosity terms in the Navier–Stokes equations are usually negligible in the far field, these boundary conditions are equally valid for these equations. The use of these boundary conditions can permit a substantial reduction in the size of the computational domain as well as an increase in the rate of convergence to a steady state. Using the results of [2] a family of boundary conditions can be developed, although only the first member of the family is studied in this paper. This condition is derived in § 2, while computational results are presented in § 3.

2. Derivation of radiation boundary conditions. In high-Reynolds number flows, the effects of viscosity are usually restricted to boundary layers in the vicinity of bodies. In the far field, the viscosity effects are small compared with the truncation error of the numerical scheme. Hence, it is legitimate to derive our far-field boundary conditions based on the linearized inviscid equations of motion. The linearization is about the steady state and is based on the assumption that the flow approaches a steady state.

Gustafsson and Kreiss [7] have shown that the problem in an exterior domain can be restricted to a bounded domain only when the dependent variables approach a constant state in the far field. For nonlinear problems, it is possible for shocks to form outside the domain of integration and to affect the solution even when the flow is smooth in the far field [9]. We assume that this does not occur. In particular, we assume that the gradients of the velocity and pressure are small in the far field.

We therefore consider the Euler equations linearized about a constant state. After a rotation of the coordinate system we can assume that this steady state is given by $u = u_\infty$, $v = 0$, $p = p_\infty$, $\rho = \rho_\infty$ where u and v are the x- and y-velocity components, p is the pressure and ρ is the density. We stress that the linearized inviscid equations are used only in deriving the far-field boundary conditions. The computational results in the next section are based on solutions of the nonlinear Navier–Stokes equations.

In the far field we have

$$\hat{u}_t + u_\infty \hat{u}_x + \frac{\hat{p}_x}{\rho_\infty} = 0, \tag{1a}$$

$$\hat{v}_t + u_\infty \hat{v}_x + \frac{\hat{p}_y}{\rho_\infty} = 0, \tag{1b}$$

$$\hat{p}_t + u_\infty \hat{p}_x + \rho_\infty c_\infty^2 (\hat{u}_x + \hat{v}_y) = 0. \tag{1c}$$

Here, $\hat{p} = p - p_\infty$, $\hat{u} = u - u_\infty$, and $\hat{v} = v$ are the perturbed flow variables. Manipulation of (1) shows that $\hat{p}$ satisfies a convective wave equation

$$\hat{p}_{tt} + 2u_\infty \hat{p}_{xt} - (c_\infty^2 - u_\infty^2)\hat{p}_{xx} - c_\infty^2 \hat{p}_{yy} = 0. \tag{2}$$

It is clear that the behavior of (2) is similar to that of the wave equation, provided that the normal outflow speed is subsonic, i.e., $u_\infty < c_\infty$. When the outflow is supersonic, no boundary conditions can be specified. Instead, all variables should be calculated by some numerical procedure, e.g., extrapolation. The accuracy of this procedure is not crucial since the errors do no propagate into the region. We therefore concentrate on the subsonic case.

Let $M_\infty = u_\infty / c_\infty$ denote the steady-state far-field Mach number. For the subsonic case, $M_\infty < 1$, one boundary condition must be specified at the outflow boundary. This situation occurs frequently in transonic calculations. In many codes either $p = p_\infty$ is specified or else the problem is underspecified with all the variables extrapolated (see, e.g., [16] and [15]). Computations presented in [14] demonstrate that the boundary condition $p = p_\infty$ can lead to reflections which delay the approach to a steady state. Underspecification can lead to the wrong steady-state solution (see [7]), and so checking that the solution is smooth is not sufficient to justify the use of extrapolation for all the variables [16]. The use of a radiation boundary condition accelerates the convergence to a steady state while maintaining a well-posed problem.

In order to simplify the procedure, we introduce a change of variables:

$$(3) \qquad \xi = (1 - M_\infty^2)^{-1/2} x, \qquad \tau = c_\infty (1 - M_\infty^2)^{1/2} t + M_\infty \xi.$$

Using these new independent variables in (2), $\hat{p}$ satisfies the wave equation

$$(4) \qquad \hat{p}_{\tau\tau} - (\hat{p}_{\xi\xi} + \hat{p}_{yy}) = 0.$$

We also introduce polar coordinates

$$(5) \qquad d^2 = \xi^2 + y^2, \qquad \tan\theta = \frac{y}{\xi}$$

to study the behavior of the solutions to (4).

Lax and Phillips [10] have shown that for large time and large d, p has the asymptotic form

$$(6) \qquad \hat{p} \simeq \frac{f(\tau - d, \theta)}{d^{1/2}}.$$

In fact, (6) is an asymptotic representation for outgoing wave solutions to (4). We now wish to develop boundary conditions which match the solution to the functional form (6). It is easily seen that the condition

$$(7) \qquad \hat{p}_\tau + \hat{p}_d + \frac{\hat{p}}{2d} = 0$$

is exact for all functions which identically have the form (6).

The boundary condition (7) is the first member of a family of radiation boundary conditions developed in [1] and [2]. There it is shown that (7) yields a well-posed problem and the solution is more accurate as d approaches infinity. The results of [10] justify the use of (7) as $t \to \infty$. Hence, the use of (7) can be expected to accelerate the approach to a steady state. In addition, it can be expected that (7) will allow the artificial surface to be brought further in without loss of accuracy as compared with the boundary condition $p = p_\infty$.

Introducing the physical coordinates (t, x, y), and the total pressure p, we can rewrite as

$$(8) \qquad \frac{1}{c_\infty (1 - M_\infty^2)^{1/2}} \left[1 - \frac{x}{d} \frac{M_\infty}{(1 - M_\infty^2)^{1/2}} \right] p_t + \frac{x}{d} p_y + \frac{p - p_\infty}{2d} = 0,$$

where

$$d^2 = \frac{x^2}{1-M_\infty^2} + y^2.$$

We can then use the Euler equations (1) to eliminate the spatial derivatives of p. The resultant equation is

$$\text{(9)} \qquad \frac{1}{(c_\infty^2 - u_\infty^2)^{1/2}} p_t - \frac{\rho_\infty c_\infty^2}{c_\infty^2 - u_\infty^2} \frac{x}{d} [u_t - u_\infty v_y] - \rho_\infty \frac{y}{d} [v_t + u_\infty v_x] + \frac{p - p_\infty}{2d} = 0.$$

If the computational domain is a long thin rectangle, $0 \leqq x \leqq L$, $0 \leqq y \leqq b$, with $b/L \ll 1$, then (9) reduces to

$$\text{(10)} \qquad p_t - \rho_\infty c_\infty u_t + \alpha(p - p_\infty) = 0, \qquad \alpha = \frac{1}{2L},$$

which is the optimal condition of Rudy and Strikwerda [14].

For axisymmetric cylindrical coordinates (t, z, r), (7) is replaced by

$$\text{(7')} \qquad \hat{p}_\tau + \hat{p}_d + \frac{\hat{p}}{d} = 0,$$

with $d^2 = z^2/(1-M_\infty^2) + r^2$. Similarly, in (8) and (9) (x, y) is replaced by (z, r), and the inhomogeneous term is $(p - p_\infty)/d$ rather than $(p - p_\infty)/2d$.

Note that at the steady state (9) does not enforce $p = p_\infty$ at the outflow boundary. However, p is equal to p_∞ only at infinity and not along any finite boundary. Instead, (9) is an improvement based on an asymptotic expansion in the reciprocal of the distance [2]. Since the gradient of v is small in the far field, p is approximately equal to p_∞. In fact, the use of zeroth-order extrapolation for the velocities, a common numerical boundary condition, approximates the condition $v_x = u_x = 0$ and enforces $p = p_\infty$ in the steady state.

In the implementation of (9), we have generally neglected the spatial derivatives of v. Furthermore, the steady states ρ_∞, u_∞ and c_∞ are usually not known. Hence, these values are replaced by the solution at the previous time step. The finite difference form of this modified radiation condition is

$$\text{(11)} \qquad \begin{aligned} &p_{i,j}^{n+1} - p_{i,j}^n - \frac{\rho_{i,j}^n (c_{i,j}^n)^2}{Q_{i,j}^n} \frac{x_{i,j}}{d_{i,j}} (u_{i,j}^{n+1} - u_{i,j}^n) \\ &\quad - \rho_{i,j}^n Q_{i,j}^n \frac{y_{i,j}}{d_{i,j}} (v_{i,j}^{n+1} - v_{i,j}^n) + \frac{Q_{i,j}^n \Delta t}{d_{i,j}} (p_{i,j}^{n+1} - p_\infty) = 0, \end{aligned}$$

where $Q_{i,j}^n = [(c_{i,j}^n)^2 - (u_{i,j}^n)^2]^{1/2}$. We calculate $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$ by zeroth order extrapolation, and $p_{i,j}^{n+1}$ by (11). The solution of (9) in conjunction with a one-dimensional implicit algorithm is discussed in the next section.

Until now, we have discussed the boundary conditions applicable at the downstream boundary. However, boundaries tangent to the mean flow also arise in practice. These boundaries are characteristic boundaries based on a linearization about the steady state. It is found that the extrapolation of all the variables can introduce oscillations. These oscillations delay the convergence to a steady state and can also degrade the accuracy of the steady state when the boundary is close to the region of interest. Since one does not want the artificial top boundary to be too far away in order to save on computer storage and running time, it is advisable not to extrapolate all the variables.

The use of the boundary condition (9) or (11) has resulted in dramatic accelerations to the steady state and has also permitted substantial constrictions in the computational domain. In the algorithm at the top boundary, it is advisable to replace p_∞ by p^n since p is not close enough to p_∞ when the top boundary is brought far in.

An alternative for the top boundary is to express (2) in terms of the convective derivative

$$\frac{D}{Dt}=\frac{\partial}{\partial t}+u_\infty\frac{\partial}{\partial x}.$$

Then (2) becomes

$$\frac{D^2\hat{p}}{Dt^2}-c_\infty^2(\hat{p}_{xx}+\hat{p}_{yy})=0. \tag{12}$$

This leads to the radiation boundary condition

$$p_t-\rho_\infty\frac{c_\infty}{d}(xu_t+yv_t)+\frac{p-p_\infty}{2d}=0. \tag{13a}$$

The condition

$$p_t-\rho_\infty c_\infty v_t=0, \tag{13b}$$

based on the one-dimensional characteristic theory, also provides improvement similar to (11) for the top boundary. The advantage of (13) is that it is equally valid for supersonic flow. We recall that even for supersonic flow, one boundary condition is required along the top boundary since the flow is subsonic with respect to the normal (i.e., v) velocity. Nevertheless, many people have used extrapolation for all the variables at the top boundary. For the variable v, zeroth order extrapolation is equivalent to

$$v_x=0. \tag{13c}$$

The results of Table 1 show that (13c) can delay the achievement of a steady state and degrade the accuracy of the resultant solution, especially when the top boundary is close to the region of interest.

3. Computational results. To validate the effectiveness of the boundary condition (9) or (11), we have tested it for a variety of problems and schemes. The first set of tests is based on the full nonlinear, time-dependent and compressible Navier–Stokes equations with the viscosity given by Sutherland's law. The equations are solved using the MacCormack two-step algorithm [11].

We first consider a rectangular domain with a uniform flow with a Mach number equal to 0.8. In the center of the domain, the initial velocity, pressure and density are changed to a different constant state (see [14]). Analytically, this perturbation should propagate downstream and exit through the downstream boundary. When the outflow boundary condition was chosen as $p=p_\infty$, the numerical solution had not converged after 20,000 iterations. Detailed output indicates that long-wavelength perturbations oscillate between the inflow and outflow boundaries, preventing the achievement of a steady state [14]. This perturbation decays slowly as a result of the dissipation in the scheme and in the equations. As a second boundary condition, we chose (10) with an optimal α, chosen by experimentation. This condition was proposed in [14] and accelerated the iteration procedure so that a steady state was achieved in 3600 iterations. With the radiation condition (11), a steady state was achieved in 1850 iterations. Thus, an acceleration by a factor of two was achieved compared with (10) without the need for a free parameter. Compared with the boundary condition $p = p_\infty$, (9) improved the convergence by a factor greater than 10.

As a more realistic problem, we consider the Navier–Stokes equations for subsonic flow over a flat plate. In this case, we apply (11) at both the characteristic top surface and the outflow boundary. We also consider the effects of varying the position of the top boundary as well as acceleration of the convergence. The same set of vertical grid points are used in all the runs, so reducing the height of the top boundary reduces the number of mesh points and hence the computational effort. The height of the top boundary is measured in boundary layer thicknesses.

The results of Table 1 show that (9) is comparable to the condition in [14] (i.e., (10)) when extrapolation is performed along the upper boundary. The use of (9) or

TABLE 1

Two-dimensional MacCormack scheme for flow over a flat plate using the compressible Navier–Stokes equations. Usually the origin is at (1.0). **Indicates inaccurate steady state.*

Position of top	b.c. at top	Outflow b.c.		No. iterations
1.0	(13c)	(10)	$\alpha = 0.3$	12800
1.0	(13c)	(9)		12500
1.0	(13c)	(9)	origin = (0, 0)	14000
1.0	(13a)	(9)		8800
0.6	(13b)	(10)	$\alpha = 0.3$	12850
0.6	(13a)	(10)	$\alpha = 0.3$	13350
0.6	(13c)	(9)		13950*
0.6	(13b)	(9)		8800
0.6	(13a)	(9)		9100
0.4	(13a)	(9)		9400
0.4	(13b)	(9)		9500

(13) at the top, together with (9) at the outflow, significantly accelerates the convergence to a steady state. Thus, when (13) is used at the upper surface, the imposition of (9) at the downstream boundary is an improvement over (10) or $p = p_\infty$. When the upper boundary is close to the flat plate, extrapolation at the upper boundary significantly degrades the accuracy of the steady state and delays the achievement of the steady state but (9) or (13) yields acceptable results.

As an additional test, we consider subsonic flow about a NACA-0012 airfoil at a 0° angle of attack. A nonorthogonal grid is used and the Euler equations are integrated using a finite volume technique [13]. Different time steps are used at each mesh point so that the difference scheme is not consistent with the time-dependent equations. The free-stream Mach number is 0.4. The origin for the radiation condition (9) is chosen at the lower left corner. In Table 2 we present the results for different exit boundary conditions. The methods denoted by * are boundary conditions without the lower-order terms involving $p - p_\infty$. The number of time steps for max $(\partial\rho/\partial t)$ to be less than 10^{-4} is given. We also show $1/N\ (\sum_{i,j} (h - h_0)^2)^{1/2}$ where h is the specific enthalpy. Since $h = h_0$ in the steady state, this is a measure of the accuracy of the steady state solution. We see from Table 2 that all the radiation-like boundary conditions are considerably better than $p = p_\infty$ specified at the outflow boundary. In particular, they allow the outflow boundary to be placed immediately beyond the airfoil. Underspecification by extrapolating all the variables deteriorates the accuracy of the steady state in addition to retarding the achievement of the steady state.

TABLE 2
*Finite-volume technique for Euler equations about NACA-0012 airfoil. * Indicates that lower order terms were not included.*

Downstream b.c.	$\sum (h-h_0)^2 \times 10^{-4}$	No. steps
$p=p_\infty$	6.68	>9000
(9)	5.96	600
(9*)	5.97	600
(10)	5.91	601
(10*)	5.90	599
Extrapolation	30.03	2111

When the steady state contains a shock, the use of the radiation condition does not seem to significantly accelerate the achievement of a steady state. In this case, the correct formulation of the inflow data is more important than the outflow boundary condition.

As a further example, we consider the one-dimensional nozzle equations,

$$\begin{aligned}&(A\rho)_t+(A\rho u)_x=0,\\ &(A\rho u)_t+[A(\rho u^2+p)]_x=A_xp,\\ &(AE)_t+[Au(E+p)]_x=0.\end{aligned} \tag{14}$$

where $A(x)$ is the cross sectional area of the nozzle. One method used to solve these equations is a linearized implicit Euler method as suggested by Briley and McDonald [4] and Beam and Warming [3]. The inflow is subsonic with both u and E specified. We consider only flows that are subsonic at the outflow with the exit pressure given. A shock forms when the exit pressure is within the proper range.

The boundary condition $p=p_\infty$ is implemented within the code by a linearization technique similar to that employed for the difference scheme. Let $\Delta v=v^{n+1}-v^n$. Then the condition $p^{n+1}=p_\infty$ is implemented by

$$\frac{u^2}{2}\Delta(A\rho)-u\Delta(A\rho u)+\Delta(AE)=\frac{A(p_\infty-p^n)}{\gamma-1}. \tag{15}$$

Similarly, the radiation condition

$$\frac{\partial p}{\partial t}-\rho c\frac{\partial u}{\partial t}+\alpha(p-p_\infty)=0 \tag{16}$$

is implemented as

$$\begin{aligned}&\left[\frac{(1+\alpha\Delta t)(u^n)^2}{2}+\frac{c^nu^n}{\gamma-1}\right]\Delta(A\rho)-\left[(1+\alpha\Delta t)u^n+\frac{c^n}{\gamma-1}\right]\Delta(A\rho u)\\ &\qquad+(1+\alpha\Delta t)\Delta(AE)=\frac{\alpha\Delta tA(p_\infty-p^n)}{\gamma-1}.\end{aligned} \tag{17}$$

We note that as $\alpha\to\infty$, (17) reduces to (15). We further note that the condition $p=p_\infty$ is a physically relevant boundary condition. Hence, the use of (16) is not consistent with the time-dependent equations. Only in the steady state do we recover the accuracy of the scheme.

In Table 3 we present the results for a selection of shock-free computations. Since the analytic solution is known for both smooth and shocked flows, exact errors can be calculated. The results of Table 3 show that the radiation condition (16) accelerates the achievement of a steady state even with the use of implicit methods and large time steps. The characteristic condition, $\alpha = 0$, produces no reflections in one dimension, and so is an appropriate radiation condition. The correct steady state is achieved due to the proper choice of initial conditions. The choice $\alpha = 0.278$ was suggested in [14]. Similarly, gains in efficiency are obtained when a Chebyshev method (see [5]) is used to obtain the solution (see Table 3b).

When the analytic steady-state solution contains a shock, the use of the radiation condition does not accelerate the achievement of a steady state. The value of the back pressure determines the location of the shock. Since we begin with a smooth flow, the use of (16) can delay the correct formation of the shock (see Table 4).

TABLE 3a

MacCormack scheme for nozzle equations (14). *Time steps chosen so that* $\max(\Delta t/\Delta x)(|u|+c) \leqq 0.9$ *with* 33 *mesh points. Exit Mach number is* 0.58. *No artificial viscosity is needed.*

Exit 15 b.c.	L^2 steady-state error $\times 10^{-4}$	No. steps
$p = p_\infty$	6.32	1670
(16) $\alpha = 0.0$	6.81	482
(16) $\alpha = 0.278$	23.14	477
(16) $\alpha = 1.0$	42.62	510
(16) $\alpha = 10.0$	9.28	1139

TABLE 3b

Chebyshev collocation scheme for nozzle equations (14). *Time steps chosen so that* $\max N\Delta t(|u|+c) \leqq 0.1$ *with* $N = 33$. *Higher modes are filtered for stability.*

Exit b.c.	L^2 steady-state error $\times 10^{-4}$	No. steps
$p = p_\infty$	2.31	19,100
(16) $\alpha = 0.0$	5.19	4,334
(16) $\alpha = 0.278$	24.23	4,271
(16) $\alpha = 1.0$	47.04	4,044
(16) $\alpha = 10.0$	19.57	10,214

TABLE 3c

Implicit linearized backward Euler scheme for nozzle equations (14). *The radiation condition is implemented as in* (17). *Artificial viscosity is added for stability. The time steps are chosen so that* $\max(\Delta t/\Delta x)(|u|+c) = 10.0$.

Exit b.c.	L^2 steady-state error $\times 10^{-4}$	No. steps
$p = p_\infty$	11.44	153
(16) $\alpha = 0.0$	8.11	52
(16) $\alpha = 0.278$	9.33	118
(16) $\alpha = 1.0$	8.40	182
(16) $\alpha = 10.0$	11.16	158

TABLE 4
MacCormack scheme for nozzle equations (14) *when steady solution contains an imbedded shock. Time step chosen so that* max $(\Delta t/\Delta x)(|u|+c) = 0.9$ *with* 33 *mesh points.* L^2 *error excludes shock region.*

Exit b.c.	L^2 steady-state error $\times 10^{-4}$	No. steps
$p = p_\infty$	20.29	499
(16) $\alpha = 0.0$	18.14	473
(16) $\alpha = 0.278$	27.03	718
(16) $\alpha = 1.0$	16.78	571
(16) $\alpha = 10.0$	20.52	640

4. Conclusion. Time-dependent codes are frequently used to achieve a steady state. For large Reynolds number situations, the steady state is achieved by allowing the energy to propagate to infinity. The boundary condition (9) simulates the radiation of energy to infinity, and so allows a more rapid achievement of a steady state. In addition, the application of (9) at characteristic boundaries is found to permit a substantial constriction in the size of the computational domain.

The efficiency of the boundary condition has been shown for a variety of situations. Further examples, for time-dependent problems, are given in [2] and [17]. A one-dimensional test shows that the boundary conditions are also useful for implicit codes with large time steps and for global methods such as a Chebyshev collocation algorithm, as well as for the standard explicit methods. This is true even though the boundary condition $p = p_\infty$ is physically relevant. Nevertheless, the use of a radiation condition allows the attainment of an accurate steady state in addition to accelerating the approach to a steady state. Imbedded shocks within the flow seem to reduce the efficiency of the radiation boundary condition.

REFERENCES

[1] A. BAYLISS AND E. TURKEL, *Computation of acoustic waves in a jet*, ICASE Rep. 78–22, Langley, VA, 1978.

[2] ———, *Radiation boundary conditions for wave-like equations* Comm. Pure Appl. Math., 33 (1980), pp. 707–726.

[3] R. W. BEAM AND R. F. WARMING, *An implicit finite difference algorithm for hyperbolic systems in conservation-law form*, J. Comput. Phys., 22 (1976), pp. 87–110.

[4] W. R. BRILEY AND H. MCDONALD, *Solution of the multi-dimensional compressible Navier–Stokes equations by a generalized implicit method*, Ibid., 24 (1977), pp. 372–397.

[5] D. GOTTLIEB AND S. ORSZAG, *Numerical Analysis of Spectral Methods: Theory and Applications*, CBMS Regional Conf. Ser. in Appl. Math., 26, Society for Industrial and Applied Mathematics, Philadelphia, 1977.

[6] C. E. GROSCH AND S. ORSZAG, *Numerical solution of problems in unbounded regions: Coordinate transformations*, J. Comput. Phys., 25 (1977), pp. 273–295.

[7] B. GUSTAFSSON AND H.-O. KREISS, *Boundary conditions for time dependent problems with an artificial boundary*, Ibid., 30 (1979), pp. 333–351.

[8] B. GUSTAFSSON AND A. SUNDSTRÖM, *Incompletely parabolic problems in fluid dynamics*, SIAM J. Appl. Math., 35 (1978), pp. 343–359.

[9] G. W. HEDSTROM, *Nonreflecting boundary conditions for nonlinear hyperbolic systems*, J. Comput. Phys., 30 (1979), pp. 222–237.

[10] P. D. LAX AND R. S. PHILLIPS, *Decaying modes for the wave equation in the exterior of an obstacle*, Comm. Pure Appl. Math., 22 (1969), pp. 737–787.

[11] R. W. MACCORMACK, *Numerical solution of the interaction of a shock wave with a laminar boundary layer*, in Lecture Notes in Physics, vol. 8, Springer-Verlag, New York, 1971, pp. 151–163.

[12] J. OLIGER AND A. SUNDSTROM, *Theoretical and practical aspects of some initial boundary value problems in fluid dynamics*, SIAM J. Appl. Math., 35 (1978), pp. 419–446.

[13] A. RIZZI, *Transonic solutions of the Euler equations by the finite volume method*, Symposium of Intl. Union Theoret. and Appl. Mech., K. Oswetitisch and D. Rues, eds., Springer-Verlag, Berlin, 1976.

[14] D. RUDY AND J. C. STRIKWERDA, *A non-reflecting outflow boundary condition for subsonic Navier–Stokes calculations*, J. Comput. Phys., 36 (1980), pp. 55–70.

[15] ———, *Boundary conditions for subsonic compressible Navier–Stokes calculations*, ICASE Rep. 79–18; Comput. & Fluids, 9 (1981), pp. 327–338.

[16] P. D. THOMAS, *Numerical method for predicting flow characteristics and performance of nonaxisymmetric nozzles-theory*, NASA Contractor Rep. 3147, NASA Langley Res. Ctr., Langley, VA, 1979.

[17] E. TURKEL, *Numerical methods for large-scale time-dependent partial differential equations*, in Computational Fluid Dynamics, W. Kollmann, ed., Hemisphere, Washington, London, 1980, pp. 127–262.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

THE CELL DISCRETIZATION ALGORITHM FOR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS*

JOHN GREENSTADT†

Abstract. The cell discretization algorithm has been developed for the solution of partial differential equations. Its application to boundary value problems involving self-adjoint elliptic equations is described, including the treatment of eigenvalue problems. Some discussion of its relationship to the finite element method is also included. Finally, various representative problems are solved numerically, by means of a Fortran program which implements the algorithm. The solutions give some indication of the behavior of the method for Dirichlet, Neumann and mixed boundary conditions. Some problems from the literature are also solved, so that comparisons can be made with other methods.

Key words. partial differential equations, cell discretization, finite element methods

1. Introduction. The cell discretization algorithm (abbreviated CD) is a procedure for reducing problems involving partial differential equations to discrete form, so that they can then be solved on digital computers. Work on this method was started in the mid-1950s, and the primary motivation was to develop a method sufficiently general that it could be applied in a straightforward way to problems involving irregular geometries. Although the initial efforts to formulate such a general method used the finite-difference approach, it was soon realized that this approach is too limited in too many ways, so that an approach based on subdomains as the principal discrete elements was adopted. At about this time, the well-known finite element method (FEM) was being developed, which uses the same basic idea of decomposing the domain in which the problem is posed into subdomains, each with its own approximate representation of the solution function.

The CD approach was formulated in variational terms from the start, but the initial functional was the least squares integral (the method of Trefftz) rather than the Dirichlet integral, on which the (generally preferable) Rayleigh–Ritz method is based. Moreover, the connections between the intracell representations, which must be made at the cell interfaces, were imposed in the form of what are now known as penalty terms (in this case, the integrals over the interfaces of the squares of the mismatches between adjacent representations). As is currently well known, penalty functions were first used by Courant in 1943 [3], but he did not there apply them to interfaces. The formulation of CD based on interface penalty functions was published in 1959 [6].

The present formulation treats the interface relationships quite differently, in that the penalty terms were abandoned (in 1967, see [7]), and the interface connections were made in the form of explicit, exact constraints, so that the discretization method reduces to the solution of a constrained variational problem, instead of an unconstrained one involving penalty functions. There are certain computational and procedural advantages to this which will be made clear as the algorithm is described. The discrete equations resulting from this procedure originally involved Lagrange multipliers, which made their solution by *iterative* means exceedingly complicated. Descriptions of this approach were given in an IBM report issued in 1967 [8], and in a talk at a Dundee Conference in 1971 [10].

To remove these complications, the discrete variables are pretransformed in such a way that the interface constraints become identities, and do not appear explicitly. The resulting minimization problem, from which the discrete equations are derived,

* Received by the editors June 2, 1980, and in revised form November 24, 1981.

† IBM Corporation, Palo Alto Scientific Center, Palo Alto, California 94304.

is again *unconstrained* (but without penalty terms), and involves a positive-definite quadratic form. It is well known that the resulting discrete equations can be solved by the Gauss–Seidel method, among others. A description of this stage of cell discretization appeared in an IBM report in 1972 [11]. In this report, the solutions (by means of a Fortran program) to various model problems were shown, demonstrating numerically (but not proving mathematically) that the method worked, and that it deserved further investigation. In the time since 1972, CD has been applied to the nuclear reactor multigroup problem, and has yielded very good numerical results for some extremely large problems, in both 2 and 3 dimensions. Furthermore, after substantial effort was made to make the program run more efficiently, the solution times became comparable with those for the standard reactor codes. (This work will be described in a separate paper.)

In addition, various technical problems associated with the method were identified and solved during this time, such as the proper treatment of Neumann and mixed boundary and interface conditions, the improvement of the "relaxation" algorithm (namely, by the replacement of the Gauss–Seidel method by the generalized conjugate gradient method [2]), the use of extrapolation in eigenvalue problems, etc. On the theoretical side, nothing has been proved with regard to convergence, partly because the methods used for the FEM do not carry over to CD. The use of a posteriori interface constraints, coupled with a freer choice of cellwise approximations, suggests that the "minimizing sequence" idea must be generalized to a "minimaxing sequence", for which the analysis is more complicated. The methods used to analyze CD are probably closer to those devised for the Weinstein method of intermediate problems [19], rather than to those suitable for the analysis of the FEM. These matters are discussed further in § 4.

The current formulation of CD is contained in a 1980 IBM report [12], of which this paper is a condensation. Thus, we shall have to refer the reader to that report for many of the details of the algorithm. However, we shall show in this paper that cell discretization is not a subset of the finite element method and, indeed, does not much overlap with it.

Although we describe here the application of CD to the elliptic case only, the method is also applicable (at least in principle) without much modification, to the parabolic and hyperbolic cases as well. An outline of these considerations may be found in an IBM report issued in 1967 [9]. However, we have had no numerical experience with time-dependent problems.

2. Discretization procedure. We shall consider the self-adjoint elliptic partial differential equation (PDE):

$$-\sum_{\substack{i=1\\j=1}}^{n} \frac{\partial}{\partial x_i}\left(a_{ij}(x)\frac{\partial\psi(x)}{\partial x_j}\right)+b_1(x)\psi(x)=b_2(x). \tag{2.1}$$

This equation is to be solved in a domain Ω, which is bounded, open, connected, etc. in $R^n (n \geqq 2)$, and whose boundary (rectifiable, without cusps, etc.) is Γ. We also assume that $\{a_{ij}\}$, b_1 and b_2 are integrable over various subdomains, surface segments, etc.

We first partition Ω into K subdomains $\{\Omega_k; k=1,\cdots,K\}$ (or *cells*), some of which will be *contiguous* with each other, i.e., if Ω_k and Ω_m are two contiguous cells, then the intersection of their closures, $\bar{\Omega}_k \cap \bar{\Omega}_m$, has a nonvanishing surface measure. Thus, e.g., cells whose closures overlap only in a point are *not* contiguous. (Points play no role in CD.) We shall denote the intersection of $\bar{\Omega}_k$ and $\bar{\Omega}_m$ by $\bar{\Gamma}_{km}$, their *interface*. The interior of $\bar{\Gamma}_{km}$ will be denoted by Γ_{km}.

For convenience, the exterior of $\bar{\Omega}$ in R^n will be denoted by Ω_0, so that the "interface" segments $\{\Gamma_{k0}\}$ are really the boundary segments of Ω. In Fig. 1 is shown a simple two-dimensional illustration of all these features. Note, e.g., that Ω_2 and Ω_6

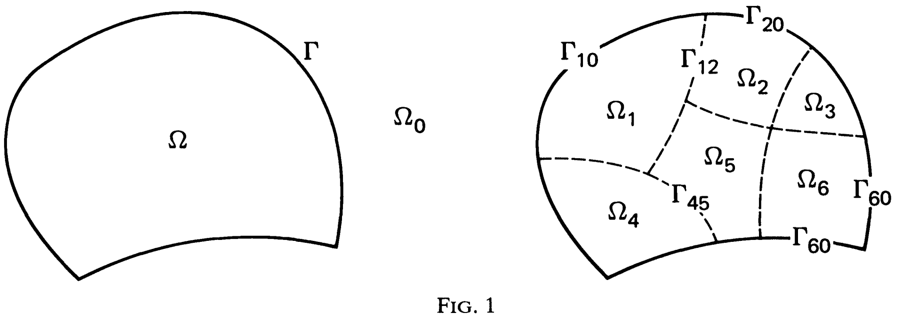


FIG. 1

are *not* contiguous neighbors, nor are Ω_3 and Ω_5. Ω_3 and Ω_4 are not neighbors at all. Note that the cell structure in Ω is quite general in that, for example, the cells are not required to be triangular, quadrilateral, etc., nor are the vertices of contiguous neighbors required to coincide. This flexibility makes CD potentially very adaptable to irregular geometries.

For each value of k, we assign to Ω_k a separate approximation $\psi_k(x)$ to the dependent variable $\psi(x)$. This approximation will depend on a finite set of unknown parameters $\{\theta_{k\mu}; \mu = 1, \cdots, M_k\}$, but its functional form must be specified in advance. Note that each M_k may be different. Approximations of this type are also quite general, and are analogous to those used in the Ritz method. They need not be polynomials, and examples will be exhibited in which trigonometric functions were used. For linear problems, such as the one being considered, the most sensible choice for ψ_k is

$$\psi_k(x; \theta_k) \equiv \sum_{\mu=1}^{M_k} \theta_{k\mu}\phi_{k\mu}(x). \tag{2.2}$$

The *basis* functions $\{\phi_{k\mu}(x)\}$, as we shall call them, are assumed to be linearly independent in Ω_k and must also be assumed to satisfy certain additional "completeness" requirements on the boundary of Ω_k, which we shall explain in § 4.

To describe a cell boundary Γ_k in more detail, we must introduce some notational conveniences. Let the set of contiguous neighbors of Ω_k be labeled by $\{m_1, m_2, \cdots, m_{J_k}\}$. We shall frequently denote this set of labels by the symbol $m[k]$, and can thus define Γ_k to be

$$\Gamma_k = \bigcup_{m[k]} \Gamma_{km[k]}, \tag{2.3}$$

i.e., the union of all the interfaces which Ω_k shares with its contiguous neighbors. Another notational convention will be to replace the symbol x by the symbol s when x is understood to label points on an interface.

We next select, for each interface Ω_{km}, a set of linearly independent *weight* functions $\{w_{km}^{\gamma}(s); \gamma = 1, \cdots, L_{km}\}$ which need only be defined on Γ_{km} itself. The use of these weight functions is the most distinctive feature of CD, and we shall have much more to say about the nature of the $\{w_{km}^{\gamma}\}$.

We use these weight functions to apply what we shall call *moment collocation* at the interfaces $\{\Gamma_{km}\}$. Let us denote by $\Delta_{km}(s)$ the *mismatch* between $\psi_k(s)$ and $\psi_m(s)$

on Γ_{km}. Thus:

$$\Delta_{km}(s) \equiv (\psi_k(s, \theta_k) - \psi_m(s, \theta_m))_{s \in \Gamma_{km}}. \tag{2.4}$$

Now, instead of requiring $\Delta_{km}(s)$ to vanish identically (or at selected interface points), we shall require only "weak" or "moment" continuity, in the sense that:

$$\int_{\Gamma_{km}} \Delta_{km}(s) w_{km}^{\gamma}(s)\, d\Gamma_{km} = 0, \qquad \gamma = 1, \cdots, L_{km}, \tag{2.5}$$

where $d\Gamma_{km}$ represents the surface element on Γ_{km}. Moment collocation of this kind does not seem to appear in the FEM literature, but it was used (for boundary conditions only) by A. Weinstein [19, p. 68] in his method of intermediate problems, which he used to solve for the vibrational modes of clamped elastic plates. Weinstein, however, like Rayleigh and Ritz, dealt only with single domains, and never applied this collocation method to interfaces. It is important to note that no effort is made in CD to ensure interface continuity of any order, but only this "weak" continuity. Hence the cell method is always a priori a C^{-1} method.

A more general definition of Δ_{km} often proves useful. It involves the "covariant" normal derivative, which appears in a natural way in variational treatments, and which is defined by:

$$\frac{\partial \psi_k}{\partial n_{km}} \equiv \sum_{\substack{i=1\\ j=1}}^{n} n_{km}^{i} a_{ij}^{(k)} \left(\frac{\partial \psi_k}{\partial x_j}\right)_{x \in \Gamma_{km}}. \tag{2.6}$$

This definition carries the implication that the derivative is in the direction from Ω_k to Ω_m, evaluated at Γ_{km}, whose unit normal is $\{n_{km}^{i}\}$.

With the convention that $\partial\psi_m/\partial n_{km}$ is the normal derivative at Γ_{km} taken in the *same* direction as $\partial\psi_k/\partial n_{km}$ (i.e., from Ω_k to Ω_m), the most general interface mismatch we shall consider has the form:

$$\Delta_{km}(s) \equiv \left(P_k\psi_k + Q_k\frac{\partial \psi_k}{\partial n_{km}} - R_k\right) - \left(P_m\psi_m + Q_m\frac{\partial \psi_m}{\partial n_{km}} - R_m\right), \tag{2.7}$$

with P_k, Q_k, etc. all being functions of s (i.e., evaluated on Γ_{km}). We shall refer to this Δ_{km} as one of "Type 2", whereas, the Δ_{km} defined in (2.4), we call "Type 1". (The same terminology is applied to the interface conditions containing Δ_{km}.) The Type 2 conditions are useful when the interface separates different media. They are of particular use in imposing exact conservation conditions across interfaces. For example, to impose the conservation of total "current" across Γ_{km}, we set the Ps and Rs to zero in (2.7), and set the Qs to unity. We then impose (2.5) only for $\gamma = 1$ (assuming that w_{km}^{1} is a constant). This is a particularly advantageous feature of CD, in that important conservation conditions can be carried over exactly to the discrete equations, in the very process of discretization. However, it is necessary to take account of possibly conflicting "natural interface conditions", which are induced by the variational procedure, and which can make the discrete problem insoluble. This difficulty can be circumvented in a systematic way by adding certain "nullifier" terms to the original functional. This point is treated fully in Appendix C of [12]. (Our computer program does the necessary correction automatically, when any imposed interface condition can lead to a potential inconsistency.)

If the approximations (2.2) are substituted into the collocation condition (2.5), this condition is discretized, and takes the form:

$$\sum_{\mu=1}^{M_k} U_{km}^{\mu\gamma}\theta_{k\mu} - \sum_{\nu=1}^{M_m} U_{mk}^{\nu\gamma}\theta_{m\nu} = W_{km}^{\gamma} - W_{mk}^{\gamma}, \tag{2.8}$$

where

$$U_{km}^{\mu\gamma} \equiv \int_{\Gamma_{km}} \left(P_k\phi_{k\mu} + Q_k\frac{\partial\phi_{k\mu}}{\partial n_{km}}\right) w_{km}^{\gamma}\, d\Gamma_{km} \tag{2.9}$$

and

$$W_{km}^{\gamma} \equiv \int_{\Gamma_{km}} R_k w_{km}^{\gamma}\, d\Gamma_{km}, \tag{2.10}$$

with corresponding expressions for the other discrete quantities.

If we regard $\{\theta_{k\mu}\}$ as the vector θ_k with M_k components, $U_{km}^{\mu\gamma}$ as a matrix U_{km} of order $M_k \times L_{km}$, and W_{km} as a vector with L_{km} components, etc., then (2.8) can be written in matrix form as follows:

$$U_{km}^T\theta_k - U_{mk}^T\theta_m = W_{km} - W_{mk}. \tag{2.11}$$

The boundary conditions also fit naturally into this framework. If we assume that, for $m = 0$ (i.e., in Ω_0), $\psi_0 \equiv 0$ (so that $\theta_0 \equiv 0$) and $W_{0k} = 0$, we then have:

$$U_{k0}^T\theta_k = W_{k0} \tag{2.12}$$

for all k which label boundary cells.

In [6] and [7], variational formulations were used to derive the discrete equations corresponding to the continuous equation (2.1). We choose the functional used in [7], i.e., the generalized Dirichlet integral:

$$\Phi[\psi] \equiv \int_{\Omega} \left\{\tfrac{1}{2}\sum_{i,j} a_{ij}\frac{\partial\psi}{\partial x_i}\frac{\partial\psi}{\partial x_j} + \tfrac{1}{2}b_1\psi^2 - b_2\psi\right\} d\Omega \tag{2.13}$$

where $d\Omega$ stands for the volume element in Ω. When we substitute the representation (2.2) for ψ, we obtain:

$$\Phi\{\psi_k\} = \sum_{k=1}^{K} \left\{\int_{\Omega_k} \left(\tfrac{1}{2}\sum_{i,j} a_{ij}^{(k)}\frac{\partial\psi_k}{\partial x_i}\frac{\partial\psi_k}{\partial x_j} + \tfrac{1}{2}b_1^{(k)}\psi_k^2 - b_2^{(k)}\psi_k\right) d\Omega_k\right\}. \tag{2.14}$$

As mentioned in the Introduction, we do not use penalty terms such as were used in [6]. These were of the form

$$\Lambda_{km} \equiv \lambda_{km}\int_{\Gamma_{km}} \Delta_{km}^2\, d\Gamma_{km} \tag{2.15}$$

with arbitrary λ_{km}. Not only is the choice of values for the $\{\lambda_{km}\}$ ambiguous, but the inclusion of the penalty terms also causes the discrete equations to have the undesirable property that they link noncontiguous cells directly. This constitutes a form of "action at a distance" in the discrete equations, which is not consistent with their role as representatives of *differential* equations, which are *local* in nature. For these reasons, we have concluded that it is preferable to solve the *constrained minimization* problem consisting of minimizing $\Phi\{\psi_k\}$ while having the $\{\theta_k\}$ satisfy the constraints (2.11).

When we substitute (2.2) into (2.14), we obtain the discrete functional in matrix form:

$$\Phi\{\theta_k\} = \sum_{k=1}^{K} \{\tfrac{1}{2}\theta_k^T S_k \theta_k - \theta_k^T T_k\} \tag{2.16}$$

where

$$[S_k]_{\mu\nu} \equiv \int_{\Omega_k} \left\{\sum_{i,j} a_{ij}^{(k)} \frac{\partial \phi_{k\mu}}{\partial x_i} \frac{\partial \phi_{k\nu}}{\partial x_j} + b_1^{(k)} \phi_{k\mu}\phi_{k\nu}\right\} d\Omega_k \tag{2.17}$$

and

$$[T_k]_\mu \equiv \int_{\Omega_k} b_2^{(k)} \phi_{k\mu}\, d\Omega_k. \tag{2.18}$$

In the next section, we shall show how, by transforming $\{\theta_k\}$ to new variables $\{\sigma_{km}\}$ and $\{\rho_k\}$, we can reduce the *constrained* minimization in the variables $\{\theta_k\}$ to an unconstrained minimization in the variables $\{\sigma_{km}\}$ and $\{\rho_k\}$. The resulting discrete equations turn out to have many attractive properties with regard to computation.

3. Transformation of the discrete equations. Because of their crucial role in describing the interface conditions, we should expect the matrices $\{U_{km}\}$ to play an important role in the transformation to new variables, as indeed they do. The first step is to collect all $\{U_{km[k]}\}$ into one matrix U_k. Recalling the notation introduced in the last section, this means:

$$U_k \equiv \{U_{km_1}, U_{km_2}, \cdots, U_{km_{J_k}}\}. \tag{3.1}$$

Thus, for all of the J_k *faces* of Ω_k (which are the same as the interfaces it shares with its J_k contiguous neighbors), we have collected the arrays of the coefficients of θ_k into one matrix. Since all of the matrices $\{U_{km}\}$ have M_k rows, it is clear that U_k also has M_k rows. Recalling that there are L_{km} columns in U_{km}, the quantity

$$L_k \equiv \sum_{m[k]} L_{km[k]} \tag{3.2}$$

is the number of columns in U_k.

In order to proceed further, we must make a *very* important assumption about U_k, namely, that U_k *is of full column rank*, i.e., the set of columns of U_k form a linearly independent set. If this condition is not met, U_k will be called *degenerate*. If U_k *is* degenerate, the attempt to transform to new variables leads to *very* severe complications in the resulting discrete equations (including "action at a distance"). For details about this state of affairs, we refer the reader to Appendix A of [12]. (In our computer program, we never proceed further with a degenerate U_k, but instead try to remove the degeneracy by increasing M_k.) Since L_k (the total number of interface conditions in cell Ω_k) obviously cannot be greater than M_k (the total number of θs in this cell), the difference $M_k - L_k$, which we shall denote by N_k, must be nonnegative. However, because a vanishing N_k leads to programming complications, it is preferable to keep N_k positive, in which case we must require that $L_k < M_k$.

When U_k is *not* degenerate, we can find matrices V_k and Z_k, of orders $M_k \times L_k$ and $M_k \times N_k$ respectively, all of whose M_k columns are linearly independent, and which satisfy the following equations:

$$U_k^T V_k = I, \tag{3.3a}$$

$$U_k^T Z_k = 0 \tag{3.3b}$$

where I is a unit matrix of order $L_k \times L_k$. (In future, any unit matrix which arises will be understood to be of the appropriate order.) Similarly, 0 is of order $N_k \times L_k$. Besides satisfying (3.3a), we can choose V_k so that it also has the property (whose importance will become clear later):

$$V_k^T S_k Z_k = 0. \tag{3.4}$$

A fairly efficient algorithm for finding V_k and Z_k, given U_k and S_k, is also covered in Appendix A of [12].

Once having found V_k, we repartition it into J_k submatrices $\{V_{km}\}$, each of the same order as its corresponding U_{km}. Thus, V_k has the form $\{V_{km_1}, V_{km_2}, \cdots, V_{km_{J_k}}\}$. Relations (3.3) and (3.4) can then be recast into "interface" form:

$$U_{km}^T V_{kp} = \delta_{mp} I, \tag{3.5a}$$

$$U_{km}^T Z_k = 0, \tag{3.5b}$$

$$V_{km}^T S_k Z_k = 0 \tag{3.5c}$$

for all $m[k]$ and all $p[k]$. δ_{mp} is, of course, the well-known Kronecker symbol.

We are now ready to change variables, from θ_k to σ_{km} and ρ_k, as follows:

$$\theta_k = \sum_{p[k]} V_{kp}(\sigma_{kp} + W_{kp}) + Z_k \rho_k. \tag{3.6}$$

σ_{kp} must obviously be a vector with L_{kp} components (as must W_{kp}), and ρ_k must be of order N_k. Hence, (3.6) can be rewritten in terms of the matrix-vector product:

$$\theta_k = (V_{kp_1}, V_{kp_2}, \cdots, V_{kp_{J_k}}, Z_k) \begin{pmatrix} \sigma_{kp_1} \\ \sigma_{kp_2} \\ \vdots \\ \sigma_{kp_{J_k}} \\ \rho_k \end{pmatrix} + \text{a constant}. \tag{3.7}$$

With the transformation in this form, it is clear that, since the $M_k \times M_k$ coefficient matrix in (3.7) is nonsingular by construction, the number of *independent* scalar quantities (degrees of freedom) in $\{\sigma_{kp}, \rho_k\}$ is the same as that in θ_k (namely, M_k).

If we substitute for θ_k in $U_{km}^T \theta_k$, using formula (3.6), we obtain

$$\begin{aligned} U_{km}^T \theta_k &= U_{km}^T \left\{ \sum_{p[k]} V_{kp}(\sigma_{kp} + W_{kp}) + Z_k \rho_k \right\} \\ &= \sum_{p[k]} U_{km}^T V_{kp}(\sigma_{kp} + W_{kp}) + U_{km}^T Z_k \rho_k \\ &= \sum_{p[k]} \delta_{mp}(\sigma_{kp} + W_{kp}) + 0 \cdot \rho_k = \sigma_{km} + W_{km} \end{aligned} \tag{3.8}$$

with the help of (3.5a) and (3.5b). Similarly, $U_{mk}^T \theta_m = \sigma_{mk} + W_{mk}$, so that (2.11) becomes:

$$U_{km}^T \theta_k - U_{mk}^T \theta_m = \sigma_{km} + W_{km} - \sigma_{mk} - W_{mk} = W_{km} - W_{mk}. \tag{3.9}$$

Hence, in the transformed variables, the interface conditions reduce to the extremely simple form:

$$\sigma_{km} - \sigma_{mk} = 0. \tag{3.10}$$

Equation (3.10) tells us that, in the subsequent calculations, we may simply *identify* σ_{mk} with σ_{km}, so that we are effectively dealing with one (vector) variable on Γ_{km} instead of two. By this means, we have reduced the total degrees of freedom in the problem, while retaining the correct interface constraints implicitly. The relation between σ_{km} and σ_{mk} will arise explicitly only when differentiation is performed. This can be taken care of by simply making use of the relation:

$$\frac{\partial \sigma_{mk}}{\partial \sigma_{km}} = I \qquad (I \text{ is of order } L_{km} \times L_{km}) \tag{3.11}$$

which will come into play when we differentiate the transformed version of the functional (2.16). We obtain the latter by again substituting for θ_k according to (3.6). The result is:

$$\begin{aligned}\Phi(\sigma, \rho) = \sum_{k=1}^{K} \Bigg\{ & \Bigg(\sum_{\substack{p[k] \\ q[k]}} (\tfrac{1}{2}\sigma_{kp}^T V_{kp}^T S_k V_{kq} \sigma_{kq} + \sigma_{kp}^T V_{kp}^T S_k V_{kq} W_{kq}) - \sum_{p[k]} \sigma_{kp}^T V_{kp}^T T_k \Bigg) \\ & + (\tfrac{1}{2}\rho_k^T Z_k^T S_k Z_k \rho_k - \rho_k^T Z_k^T T_k) \\ & + \Bigg(\sum_{p[k]} (\sigma_{kp}^T V_{kp}^T S_k Z_k \rho_k + W_{kp}^T V_{kp}^T S_k Z_k \rho_k) \Bigg) \\ & + \Bigg(\sum_{\substack{p[k] \\ q[k]}} \tfrac{1}{2} W_{kp}^T V_{kp}^T S_k V_{kq} W_{kq} - \sum_{p[k]} W_{kp}^T V_{kp}^T T_k \Bigg) \Bigg\}. \end{aligned} \tag{3.12}$$

We have divided this expression (in which both p and q range over all the contiguous neighbor-labels) into four grouped parts. Starting with the fourth group, we observe that it is a constant, and does not matter, because Φ is to be differentiated. The third group, on the other hand, vanishes *identically* because of (3.5c). This has the very important effect of *decoupling* the σs and ρs, so that the first group contains only σs and the second group contains only ρs. Thus, the differentiation of Φ, which now consists of two disjoint quadratic forms, will result in two sets of *uncoupled* discrete equations.

To simplify the differentiation, we replace k by r and define:

$$H_{rpq} \equiv V_{rp}^T S_r V_{rq}, \tag{3.13a}$$

$$A_r \equiv Z_r^T S_r Z_r \tag{3.13b}$$

so that the significant part of $\Phi(\sigma, \rho)$ reduces to:

$$\begin{aligned}\Phi(\sigma, \rho) = \sum_{r=1}^{K} \Bigg\{ & \Bigg(\sum_{\substack{p[r] \\ q[r]}} (\tfrac{1}{2}\sigma_{rp}^T H_{rpq} \sigma_{rq} + \sigma_{rp}^T H_{rpq} W_{rq}) - \sum_{p[r]} \sigma_{rp}^T V_{rp}^T T_r \Bigg) \\ & + (\tfrac{1}{2}\rho_r^T A_r \rho_r - \rho_r^T Z_r^T T_r) \Bigg\}. \end{aligned} \tag{3.14}$$

We are now ready to differentiate Φ with respect to σ_{km} (not forgetting (3.11)), and set the result to zero, which gives us:

$$\begin{aligned}\frac{\partial \Phi}{\partial \sigma_{km}} = & \sum_{p[k]} H_{kmp} \sigma_{kp} + \sum_{q[m]} H_{mkq} \sigma_{mq} + \sum_{p[k]} H_{kmp} W_{kp} \\ & + \sum_{q[m]} H_{mkq} W_{mq} - V_{km}^T T_k - V_{mk}^T T_m = 0. \end{aligned} \tag{3.15}$$

If we then define

$$\Lambda_{km} \equiv H_{kmm} + H_{mkk}, \tag{3.16a}$$

$$G_{km} \equiv -\sum_{p[k]} H_{kmp} W_{kp} - \sum_{q[m]} H_{mkk} W_{mq} + V_{km}^T T_k + V_{mk}^T T_m, \tag{3.16b}$$

then (3.15) can be written (remembering that $\sigma_{mk} = \sigma_{km}$):

$$\Lambda_{km}\sigma_{km} + \sum_{p[k]\neq m} H_{kmp}\sigma_{kp} + \sum_{q[m]\neq k} H_{mkq}\sigma_{mq} = G_{km}. \tag{3.17}$$

These discrete equations for the σs are in the standard form for iterative solution. The quantity σ_{km} can be separated out, and expressed in terms of the σs associated with the "nearest neighbors" of Γ_{km}. We note again the unusual feature that the geometric elements with which the σ variables are associated are *interfaces.*

The decoupling of the σs and ρs has the additional effect that every ρ is associated with only one cell. This is evidenced by the fact that each ρ can be solved-for independently of all other variables. In fact, the differentiation of Φ with respect to ρ_k gives the result:

$$\frac{\partial \Phi}{\partial \rho_k} = A_k\rho_k - Z_k^T T_k = 0 \tag{3.18}$$

which can be solved immediately for ρ_k (assuming A_k is nonsingular).

In order to be able to solve (3.17) and (3.18), we must assume that all the matrices $\{\Lambda_{km}\}$ and $\{A_k\}$ are nonsingular (there are special exceptions, e.g., the Neumann problem in a single cell). If they are not, we must consider either the problem or the discretization ill-formulated. Experience shows that a straightforward choice of basis functions for a straightforward problem leads to no difficulties; examples will be shown in § 5. In these cases, we have solved (3.17) by the generalized conjugate gradient method [2]. This is an "iterative" form of the classical CG method, and it has performed for us very well.

The decomposition of θ into two other variables, one of which can be thought of as belonging to the "interior" of Ω_k and the other to a face (or a boundary segment) of Ω_k, is reminiscent of the representation of the classical electrostatic potential in terms of an interior source function and a dipole layer on the boundary. While this analogy is not exact, a relationship can be shown between the two representations. We shall not pursue this here, but again refer the reader to [12] for details.

Another unusual feature of this form of the discretization is the fact that the σs cannot be *localized,* in the sense that they cannot be associated with *points.* This localization would be possible only if the weight functions $\{w_{km}^\gamma\}$ could be δ-functions (or δ-distributions) defined on the Γs. However, as we shall show in the next section, the $\{w_{km}^\gamma\}$ must have certain regularity properties, i.e., they must lie in certain function spaces, and it turns out that they cannot be δ-functions. Thus, each σ can only be associated with an entire interface (just as a Fourier coefficient of a function must be associated with the entire interval in which the function is defined). For this reason, we have the curious feature that (3.17) is a system of equations linking the variables on an *interface* to the variables on its "nearest neighbor" *interfaces,* because the "nearest neighbors" of Γ_{km} are all the other faces of Ω_k and Ω_m.

4. Some theoretical observations.

We shall sketch here some theoretical features of the CD method. We are making no claim to a complete theoretical treatment (which would include, for example, a convergence proof), or anything like it. We shall simply note a few important aspects of CD which can be discerned without a deeper analysis.

As with the FEM, the Rayleigh–Ritz and the Weinstein methods, the theoretical underpinning of CD involves functional analysis, and particularly the properties of certain special function spaces. First we observe that, in order for $\Phi\{\psi_k\}$ in (2.14) to have a meaning, each approximation ψ_k must have a gradient whose square is integrable over Ω_k. Out of functions with this property, the Sobolev space W_2^1 can be constructed (see, e.g., [15, Chap. 3]). This space is also often denoted by H^1, since it can be interpreted as a Hilbert space in a natural way.

As for the domain Ω_k, in which these functions are defined, certain restrictions are put on its shape (in order to obtain reasonably strong theorems), but these mostly amount to the exclusion of cusps and crinkles, and in any case, do not rule out any reasonable sort of cell that would arise in practice. Included in these restrictions is the rectifiability (or piecewise differentiability) of every interface Γ_{km}, since we must be able to form all the necessary surface integrals.

The most important aspect of the behavior of ψ_k itself is what happens to it when the boundary segment Γ_{km} of Ω_k is approached from the interior of Ω_k. In an appropriate sense, the limiting values of ψ_k form the *trace* of ψ_k on Γ_{km}, which we shall denote by $\mathrm{Tr}_m\,(\psi_k)$. Based on Sobolev's work, J. L. Lions et al. [14] proved that the set of traces of $\{\psi_k\}$ on Γ_{km} actually form a Hilbert space, which is denoted by $H^{1/2}(\Gamma_{km})$. Roughly, this notation indicates that the "$\frac{1}{2}$th" derivative of $\mathrm{Tr}_m\,(\psi_k)$ has a Lebesgue integrable square over Γ_{km} (naturally, all fractional derivatives in this theory have a rather indirect definition). Another of these "trace theorems" shows that the trace functions associated with $\partial\psi_k/\partial n$ coincide with the Hilbert space $H^{-1/2}(\Gamma_{km})$. This means that $\mathrm{Tr}_m\,(\partial\psi_k/\partial n)$ on Γ_{km} must be integrated "$\frac{1}{2}$ times" to have a Lebesgue integrable square, which shows that it is a much "rougher" function.

Because we wish to preserve the maximum generality in the class of approximations $\{\psi_k\}$, we shall find that certain restrictions have to be put on the weight functions $\{w_{km}^\gamma\}$. The trace functions $\mathrm{Tr}_m\,(\psi_k)$ and $\mathrm{Tr}_m\,(\partial\psi_k/\partial n)$ are multiplied by w_{km}^γ, and the product integrated over Γ_{km}, in the interface conditions (2.5) (using the generalized definition of Δ_{km}). In order to ensure that these integrations lead to finite results, the Δ and w are constrained to lie in *dual* function spaces. Integrated products of the type appearing in (2.5) are called *duality pairings*, and are written:

$$\langle \Delta_{km}, w_{km}^\gamma\rangle \equiv \int_{\Gamma_{km}} \Delta_{km} w_{km}^\gamma \, d\Gamma_{km}, \tag{4.1}$$

and Δ_{km} and w_{km}^γ are regarded as lying in *dual* spaces.

The spaces $H^s(\Gamma_{km})$ and $H^{-s}(\Gamma_{km})$ (for $s>0$) are defined so that they are dual spaces [14, p. 36]. Hence, if $\Delta_{km}\in H^s(\Gamma_{km})$, then $w_{km}^\gamma \in H^{-s}(\Gamma_{km})$, etc. Now, if Δ_{km} is of Type 1, it contains only $\mathrm{Tr}_m\,(\psi_k)$ and $\mathrm{Tr}_k\,(\psi_m)$, both of which belong to $H^{1/2}(\Gamma_{km})$, as noted previously. Therefore, w_{km}^γ may belong to $H^{-1/2}(\Gamma_{km})$ at *worst* (i.e., it may also belong to a "smoother" family of functions $H^t(\Gamma_{km})$, with $t>-\frac{1}{2}$). On the other hand, if Δ_{km} is of Type 2, it may belong to $H^{-1/2}(\Gamma_{km})$, so that w_{km}^γ may belong to $H^{1/2}(\Gamma_{km})$ at worst. Hence, no matter what type Δ_{km} is, the weight function can belong to no "worse" a space than $H^{-1/2}(\Gamma_{km})$. These relationships are independent of n, the dimension of the Euclidean space in which Ω is defined.

On the other hand, it has been shown [15, p. 109] that, based on the Sobolev imbedding theorem, the *smallest* space which contains the δ-function *does* depend on the dimension ν of the space on which the functions are defined. In fact, this space is $H^{-\nu/2-\varepsilon}$, with $\varepsilon>0$. Thus, even when Ω_k is in R^2 (so that $\nu=1$ on the Γs), a

δ-function on an interface would have to be in $H^{-1/2-\varepsilon}$, which is still too rough for a weight function. Thus, it is clear that no w^{γ}_{km} can contain a δ-distribution concentrated on the interface Γ_{km}, and therefore, (2.5) cannot take the form:

$$(4.2)\qquad \langle \Delta_{km}, w^{\gamma}_{km}\rangle = \int \Delta_{km}(s)\delta(s-s^{\gamma})\,d\Gamma_{km} = \Delta_{km}(s^{\gamma}) = 0.$$

Hence, the interface collocation in CD *cannot* be a *point* collocation, and therefore cell discretization cannot be equivalent to the finite element method or any other method using point interface collocation.

If this argument were reversed, i.e., if we were to allow a weight function to contain a δ-function, then $\mathrm{Tr}_m\,(\psi_k)$ could not belong to $H^{1/2}(\Gamma_{km})$, but would have to be a more regular, or "smoother" function, in which case ψ_k itself would have to belong to a correspondingly smoother Sobolev space in Ω_k. In fact, the FEM does insure the consistency of its point collocation method, by constructing representations (or approximating subspaces) which are sufficiently regular. (It would appear that the "patch test" of Irons [18, p. 174] is intended to ascertain whether the smoothness of the representation in the so-called *nonconforming* case is adequate for stability.) This regularity requirement, however, carries with it the drawback that, in case the original problem has a solution which becomes "rough" at the boundary, it becomes more difficult to approximate it well with the smoother representations. We shall see a concrete numerical example of this in the next section.

The use of moment interface collocation does not by any means relieve CD of its own approximation problems. While we are not in a position to prove the convergence of the representations used in CD, we can point out at least one of the characteristics of moment collocation which would certainly figure in a deeper analysis.

Let us begin by thinking of the set $\{w^{\gamma}_{km[k]}\}$, i.e., all of the weight functions associated with Ω_k, as being defined over the entire perimeter of Ω_k, rather than on a particular interface Γ_{km}. (Of course, the *support* of each w would still be on a single interface.) These weight functions are L_k in number, where L_k is defined as in (3.2), and we assume that they are part of a product-space. To keep this argument simple, we shall restrict ourselves to interface conditions of Type 1, in which case the product-space of the weight functions could be $H^{-1/2}(\Gamma_k)$. (For convenience, we shall temporarily suppress the index k, which labels the cell we are discussing.) This space contains $H^0(\Gamma)$, so any discontinuities of the ws across the intersections of the interfaces need not worry us. (We assume, of course, that the ws are all linearly independent of each other.)

Let us next consider the dual Hilbert space, $H^{1/2}(\Gamma)$, and a set of functions $\{\eta^{\sigma}(s)\}$ defined on Γ (with $\sigma = 1, \cdots, \infty$), which form a basis for $H^{1/2}(\Gamma)$. Out of this set, we shall choose (perhaps with the aid of the Schmidt orthogonalization process), L of the ηs, with the property:

$$(4.3)\qquad \langle \eta^{\sigma}, w^{\gamma}\rangle \equiv \int_{\Gamma} \eta^{\sigma}(s)w^{\gamma}(s)\,d\Gamma = \delta_{\sigma\gamma}.$$

Let us next consider the set of traces, on Γ, of the set of basis functions $\{\phi_{\mu}(x)\}$, M in number. For convenience, we shall denote $\mathrm{Tr}\,(\phi_{\mu}(x))$ by $\xi_{\mu}(s)$. For the special case of a Type 1 interface condition, definition (2.9) can be rewritten:

$$(4.4)\qquad \int_{\Gamma} \xi_{\mu}(s)w^{\gamma}(s)\,d\Gamma \equiv U^{\mu\gamma}.$$

We next recall that *when $U^{\mu\gamma}$ is nondegenerate*, we can find $V^{\mu\sigma}$ such that condition (3.3a) holds. In our present notation, this means:

$$\sum_{\mu} V^{\mu\sigma} U^{\mu\gamma} = \delta_{\sigma\gamma}. \tag{4.5}$$

If we substitute for U from (4.4), we obtain:

$$\sum_{\mu} V^{\mu\sigma} \int_{\Gamma} \xi_{\mu}(s) w^{\gamma}(s)\, d\Gamma = \int_{\Gamma} \Big(\sum_{\mu} V^{\mu\sigma} \xi_{\mu}(s)\Big) w^{\gamma}(s)\, d\Gamma = \delta_{\sigma\gamma}. \tag{4.6}$$

Comparing this with (4.3), and recalling that the *w*s are linearly independent, we see that the following must be true:

$$\eta^{\sigma}(s) = \sum_{\mu} V^{\mu\sigma} \xi_{\mu}(s) \equiv \sum_{\mu} V^{\mu\sigma} \operatorname{Tr}(\phi_{\mu}(x)), \tag{4.7}$$

which means that the traces of the basis set $\{\phi_{\mu}\}$ *must span the subspace which is dual to that spanned by the weight functions.* The failure of this property is the *functional* meaning of degeneracy, as compared with the *algebraic* meaning, as defined in § 3. There would appear to be a certain similarity of the nondegeneracy requirement to the "rank condition" in the FEM [15, p. 382].

Our last speculation will be about some of the factors that would figure in a convergence proof for CD. In the finite difference and finite element methods, the key parameter that characterizes the discretization is h, the grid spacing, or the "size" of an element of the partition of the domain Ω. On the other hand, in the Rayleigh–Ritz or Weinstein methods, the key parameter is the number of terms in the expansion of the trial function. In the cell discretization method, both of these parameters occur, for each cell, as well as the number of collocation moments, on each interface. Hence, we have to deal with three sets of characteristic parameters of the discretization: the cell sizes $\{h_k\}$, the number of moments $\{L_{km}\}$ on each interface, and the number of terms $\{M_k\}$ in each intracell expansion. We may think of these sets of parameters as "global" vectors, so that we can refer simply to h, L and M.

We are going to consider a sequence which will correspond to the *minimizing sequence* of the Ritz method, and we shall number the vector parameters which characterize this sequence as follows: $\{h_1, h_2, \cdots\}$, $\{L_1, L_2, \cdots\}$ and $\{M_1, M_2, \cdots\}$. If we say that, for example, $L_1 > L_2$, we shall mean that $\{L_{km}\}_1 \geqq \{L_{km}\}_2$ for all k and m, with strict inequality at least once. The same applies to the other parameters. The functional Φ, defined in (2.16), is thus a function of the three parameter vectors, h, L and M. Since it happens that, in most of our numerical work, we have not been concerned so much with h, as with L and M, and in part, to highlight the peculiarities of the convergence problem for CD, we shall keep h fixed in the argument which follows. (For comparison purposes, however, we shall show the results, in the next section, of a study of the error as a function of h).

What happens to the functional $\Phi(L, M)$ when L or M is varied? Consistent with the restriction of nondegeneracy, it is possible to find, for each choice L_i, a sequence $\{M_{i1}, M_{i2}, \cdots\}$ such that $M_{ia} > M_{ib}$ for $a > b$, and such that Φ can be evaluated. It is a fundamental principle of analysis that the addition of degrees of freedom in a function enables it to take on a minimum value which is *no higher* than before, i.e.,

$$\Phi(L_i, M_{ia}) \leqq \Phi(L_i, M_{ib}) \quad \text{for } a > b. \tag{4.8}$$

For an elliptic operator such as that in (2.1), the corresponding functional (2.13) (and hence (2.14)) can be made positive semi-definite by the addition of a constant

[20, p. 3]; hence, Φ is certainly bounded below. On the other hand, the sequence

$$\{\Phi(L_i, M_{i1}), \Phi(L_i, M_{i2}), \Phi(L_i, M_{i3}), \cdots\} \tag{4.9}$$

is monotone nonincreasing, as indicated above. Hence, we expect that the sequence in (4.9) will have a limit for each choice L_i. We are assuming that each M_{ia} ultimately becomes infinite, and the choice of $\{\phi_{k\mu}\}$ was such as to render it capable of representing any admissible ψ_k "perfectly". We shall thus have an "exact" solution for each set of interface constraint parameters L_i, with a corresponding limiting value, Φ_i.

The same fundamental principle of analysis can also be interpreted as the assertion that any *decrease* in the degrees of freedom—as by an increase in the number of constraints—will cause the minimum value of Φ to be *no lower* than before. Therefore, we can expect that, for a fixed M consistent with nondegeneracy, we will have $\Phi(L_i, M) \geqq \Phi(L_j, M)$ for $i > j$. Thus, we can contemplate a double sequence of values of Φ: for a fixed L_i, a nonincreasing sequence of $\Phi(L_i, M_{ia})$ going to a limit Φ_i, and for increasing i, a nondecreasing sequence $\{\Phi_i\}$.

We shall also assume that the set $\{w^{\gamma}_{km}\}$ was chosen so that increasing L_{km} without limit on each Γ_{km} would result in "exact" continuity of ψ across each interface ("exact" means, of course, *almost everywhere*). This is based on the lemma [13, p. 240] that, if $\{w^{\gamma}_{km}\}$ is a set of elements dense in the inner-product space $H^0(\Gamma_{km})$, and if

$$\langle \Delta_{km}, w^{\gamma}_{km}\rangle = 0 \tag{4.10}$$

for all $w \in H^0(\Gamma_{km})$, then Δ_{km} is the null element in $H^0(\Gamma_{km})$, which means that $\psi_k = \psi_m$ almost everywhere on Γ_{km} ("weak continuity"). We are therefore entitled to hope that, in the limit, the approximation $\{\psi_k\}$ would be equal to the solution ψ almost everywhere in Ω and, since $\Phi(\psi)$ is assumed to exist, each Φ_i is bounded above by this value. Thus we have a nondecreasing sequence $\{\Phi_i\}$ which is bounded above and which, therefore, converges.

Out of this double sequence, we could choose, by some variant of the Cantor diagonal process [17, p. 17], a single "diagonal" sequence:

$$\{\Phi(L_1, M_{1a_1}), \Phi(L_2, M_{2a_2}), \cdots\} \tag{4.11}$$

which would converge to the correct value $\Phi(\psi)$. This latter sequence would be the theoretical counterpart of the manner in which the problem would be successively solved approximately in the CD method.

Even assuming convergence, the analysis of error estimates is very different from that for the FEM, in that the space of approximants is not merely a subspace of the admissible solution candidates (but is in some ways a *super*space). Insofar as the errors in the representation $\{\psi_k\}$ depend on the parameters $\{L_{km}\}$ and $\{M_k\}$, rather than only on the cell-size modulus h, the error estimates would resemble those for Fourier series, rather than power series.

5. Numerical examples. As the reader will have gleaned by now, the cell discretization method is applicable, in principle, to very irregular geometries with quite irregular cell structures. It is therefore a source of great regret that we are not in a position to describe numerical tests in which these features are exhibited. Unfortunately, we have so far been afforded the opportunity to implement CD in rectangular geometry alone.

Part of the reason for this is that the original Fortran program was deliberately kept as simple as possible, so that the question of whether the cell method worked at all could be answered. When this answer turned out to be affirmative for several model problems, a great deal of effort was devoted to speeding up and generalizing the program so that it would prove practical for the solution of multigroup nuclear reactor problems. This effort succeeded, in that our program turned out to be reasonably competitive with existing nuclear codes, in terms of both execution times and the sizes of the problems solved. (This work will be described in a separate publication.)

Although the application of CD to problems involving irregular domains and cell structures is, as we have said, quite straightforward conceptually (in that no new ideas are involved beyond what has been described), its implementation in a program would require very complicated logistical procedures, involving, for example: the input descriptions of irregular cells; interface and boundary shapes (in 2 and 3 dimensions); the automatic identification and organization of contiguous-neighbor relationships; the calculation of various integrals over irregular cells and interfaces, etc. We hope some day to be able to produce such a program.

Meanwhile, we can still present the outcome of numerical tests which show the behavior of CD in other respects than the geometrical. We have applied it to Dirichlet, Neumann, mixed and eigenvalue problems, with interesting results. In order to have the reader understand these, it is necessary that we give a few details of the actual Fortran program.

The available basis sets are all built out of certain "factor" functions, each with a single argument. The most important of these which have been included in the existing program are listed in Table 1. The factor functions $f_i(u)$ we shall use are defined as follows:

(5.1a) $$f_i(u) = P_{i-1}(u), \qquad i = 1, \cdots,$$

(5.1b) $$f_i(u) = u^{i-1}, \qquad i = 1, \cdots,$$

(5.1c) $$\{f_i(u)\} = \{1, \sin u, \cos u, \sin 2u, \cos 2u, \cdots\}$$

TABLE 1
"Factor functions" used to form basis sets.

No.	Type	Interval
1	Legendre polynomials	$[-1, 1]$
2	Powers	$[0, 1]$
3	"	$[-1, 1]$
4	Trigonometric functions	$[-\pi, \pi]$
5	" "	$[-\pi/2, \pi/2]$
6	" "	$[-\pi/4, \pi/4]$

(where $P_i(u)$ is the ith unnormalized Legendre polynomial). Out of these factors, basis sets $\{\phi_\mu(u)\}$ are then constructed as follows:

(5.2a) $$\phi_\mu(u) \equiv f_{\mu_1}(u_1) \qquad \text{(1D)},$$

(5.2b) $$\phi_\mu(u) \equiv f_{\mu_1}(u_1) f_{\mu_2}(u_2) \qquad \text{(2D)},$$

(5.2c) $$\phi_\mu(u) \equiv f_{\mu_1}(u_1) f_{\mu_2}(u_2) f_{\mu_3}(u_3) \qquad \text{(3D)}.$$

The notations "(1D)", etc., indicate which basis functions are for 1-, 2- and 3-dimensional geometries.

The *weight* functions are formed from the same factor functions as the corresponding basis sets. If u_1 and u_2 are the appropriate variables *on* Γ_{km}, we define:

(5.3a) $$w^{\gamma}(u)=1 \qquad \text{(1D)},$$

(5.3b) $$w^{\gamma}(u)=f_{\gamma_1}(u_1) \qquad \text{(2D)},$$

(5.3c) $$w^{\gamma}(u)=f_{\gamma_1}(u_1)f_{\gamma_2}(u_2) \qquad \text{(3D)}.$$

The correspondence between the single index μ (as used in the previous sections) and the set of factor indices (μ_1, μ_2, μ_3) is set up in two ways (many others are possible). The first we shall call Type 1, and it is illustrated in the 2D case by the set of products of powers:

(5.4) $$\{\phi_\mu\}=\{1;\, u_1,\, u_2;\, u_1^2,\, u_1u_2,\, u_2^2;\, u_1^3,\, u_1^2u_2,\, u_1u_2^2,\, u_2^3;\cdots\}$$

so that the zeroth "level" basis set consists of a constant, that of the first level consists of all linear monomials, that of the second level of all quadratic monomials, etc.

Type 2 combinations are correspondingly illustrated by:

(5.5) $$\{\phi_\mu\}=\{1;\, u_1,\, u_1u_2,\, u_2;\, u_1^2,\, u_1^2u_2,\, u_1^2u_2^2,\, u_1u_2^2,\, u_2^2;\cdots\}$$

where the zeroth level is again a constant, but the first level contains all bilinear monomials, the second level all biquadratic monomials, etc. The assembly of the other factor functions (Legendre, trigonometric) into basis functions is done by the same rules. In all cases, a linear transformation of the arguments is used to match the fixed range of u-values to the actual intervals in the x-variables for a given rectangular cell.

Initially, a choice of level is made for any cell Ω_k (based on a rule of thumb derived from experience) which in most cases will prevent degeneracy. If degeneracy nevertheless occurs, the level is stepped up by one. Also, if $A_k(\equiv Z_k^TS_kZ_k)$ is singular (or very badly conditioned), the level is likewise stepped up. However, in some cases, as we shall show below, no amount of "escalation" can remove degeneracy.

Case 1. *Eigenvalue problem in square.* We shall use this simple eigenvalue problem to illustrate much of the behavior of CD for various choices of basis sets, interface conditions, etc. (The simple extension of the basic method to eigenproblems is described in [12, App. D].)

The equation to be solved is:

(5.6) $$-\nabla^2\psi=\lambda\psi$$

in the square domain shown in Fig. 2, with its four cells indicated. $\psi=0$ on the boundary, and the lowest eigenvalue which we are to find has the value: $\pi^2/2=4.934802$. The corresponding eigenfunction is $\psi=A\sin\pi x/2\times\sin\pi y/2$. We shall display results in tables, which we hope will be mostly self-explanatory. All calculations were done in IBM/370 double-precision (about 16 digits).

In Table 2 are shown the values of λ obtained using Legendre polynomials for various values L of the number of interface conditions and values M for the number of terms in the expansion. The trend to small λs for a fixed L, and the trend to larger λs for a fixed M (as described in § 4) are plain to see. The slight deviations from this pattern are due to variations in the termination of the iterations for ψ, and perhaps to rounding error (since we are dealing with full matrices up to order (66×66)). The blank entries are for those combinations of L and M for which degeneracy occurs. The expansion is of Type 1, and the corresponding degree for each M is shown. The final line contains the maximum departure of ψ from zero at the boundary for the first case in the corresponding column (i.e., the case with smallest M). (The common

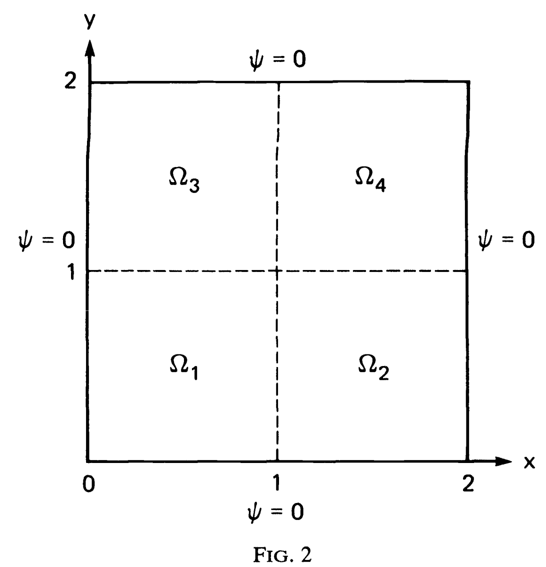


FIG. 2

TABLE 2
Values of λ *for Case* 1 *with Legendre polynomials* (*Type* 1).

	L:	1	2	3	4	5	6
M	Deg.						
6	2	4.212005					
10	3	4.117036					
15	4	4.115908	4.924406	4.935142			
21	5	4.115859	4.920742	4.934768			
28	6	4.115858	4.920645	4.934671	4.934805	4.934802	
36	7	4.115858	4.920403	4.934642	4.934803	4.934802	
45	8	4.115858	4.920403	4.934642	4.934801	4.934803	4.934804
55	9	4.115858	4.920378	4.934641	4.934801	4.934805	4.934802
66	10	4.115858	4.920378	4.934641	4.934801	4.934804	4.934803
ψ err.		4(−1)	3(−2)	5(−3)	9(−5)	4(−5)	3(−7)

convention is used of placing the appropriate exponent of 10 in parentheses following the significant figures.) Recalling again that CD yields discontinuous solutions in general, we can use the departure from zero on the boundary as a rough measure of the error in ψ.

In Table 3a are shown similar results when powers of u in the base interval [0, 1] are used for the factor functions. For the selected values shown, comparison with Table 2 indicates that the results for ψ are substantially the same as before, except that a severe deterioration sets in for larger values of M, especially for $M = 45$. This is a consequence of the much poorer condition of the matrices formed by integration of the basis functions, such as those in (2.17). The collocation error of ψ at the boundary is actually worse for $L = 6$ than for $L = 5$, clearly because of rounding error.

However, when the base interval (for u^i) is changed to [−1, 1], the results are very much better, as shown in Table 3b. The reason for this is that the odd and even

TABLE 3a
Case 1 *with powers in* [0, 1] (*Type* 1).

L	1	2	3	4	5	6
M	6	15	15	28	28	45
Degr.	2	4	4	6	6	8
λ	4.212005	4.924405	4.935144	4.934808	4.934777	4.989074
ψ err.	4(−1)	3(−2)	5(−3)	9(−5)	4(−5)	5(−4)

TABLE 3b
Case 1 *with powers in* [−1, 1] (*Type* 1).

L	1	2	3	4	5	6
M	6	15	15	28	28	45
λ	4.212005	4.924405	4.935142	4.934805	4.834804	4.934802
ψ err.	4(−1)	3(−2)	5(−3)	9(−5)	4(−5)	3(−7)

TABLE 3c
Case 1 *with Legendre polynomials* (*Type* 2).

L	1	2	3	4	5	6
M	9	16	25	36	49	64
Degr.	2	3	4	5	6	7
λ	4.212005	4.923545	4.934676	4.934801	4.934804	4.934804
ψ err.	4(−1)	4(−2)	4(−3)	3(−4)	2(−5)	1(−6)

TABLE 3d
Case 1 *with trigonometric functions in* [$-\pi/2$, $\pi/2$] (*Type* 1).

L	1	2	3	4	5	6
M	6	15	15	28	28	45
λ	4.935264	4.946600	4.963376	4.936114	4.936125	4.934855
ψ err.	4(−1)	6(−2)	2(−2)	4(−3)	3(−3)	5(−4)

TABLE 3e
Case 1 *with trigonometric functions in* [$-\pi/2$, $\pi/2$] (*Type* 2).

L	1	2	3	4	5	6
M	9	16	25	36	49	64
λ	4.395264	4.949639	4.958855	4.935983	4.935969	4.934846
ψ err.	4(−1)	6(−2)	1(−2)	2(−3)	1(−3)	2(−4)

TABLE 3f
Case 1 *with trigonometric functions in* [$-\pi/4$, $\pi/4$] (*Type* 1).

L	1	2	3	4	5	6
M	6	15	15	28	28	45
λ	4.132851	4.923178	4.934802	4.934802	4.934747	4.934906
ψ err.	4(−1)	3(−2)	4(−8)	5(−8)	1(−7)	4(−7)

powers are mutually orthogonal in $[-1, 1]$, which has the effect of improving the condition number of each matrix of integrals, so that larger orders are better conditioned than before.

In Table 3c, the results for Legendre expansions of Type 2 are shown. In this case, the numbers M of basis functions required for nondegeneracy are different from those for a Type 1 expansion, because the complete biquadratic contains more terms than a complete quadratic. The accuracy does not seem better than for the Type 1 expansion, although more degrees of freedom are used; this has been our finding in most cases.

The next choice of factor functions, the trigonometric functions in the base interval $[-\pi, \pi]$ is completely unsatisfactory, in that degeneracy occurs for all values of M. This is due to the combination of the symmetry of the rectangular cells, and the peculiarities of the factor functions. If we consider a basis function $\phi_{mn}(u, v)$ such that

$$\phi_{mn}(u, v) = \begin{Bmatrix} \cos mu \\ \sin mu \end{Bmatrix} \times \begin{Bmatrix} \cos nv \\ \sin nv \end{Bmatrix}, \tag{5.7}$$

then the values for $U^{mn\gamma}$ on the faces $u = \pm\pi$ are given, according to (2.9), by:

$$\begin{aligned} U^{mn\gamma} &= \begin{Bmatrix} \cos \pm m\pi \\ \sin \pm m\pi \end{Bmatrix} \times \int_{-\pi}^{\pi} \begin{Bmatrix} \cos nv \\ \sin nv \end{Bmatrix} \times \begin{Bmatrix} \cos \gamma v \\ \sin \gamma v \end{Bmatrix} dv \\ &= \begin{Bmatrix} \cos \pm m\pi \\ \sin \pm m\pi \end{Bmatrix} \times \{0 \text{ or } \pi\}. \end{aligned} \tag{5.8}$$

If we now compare the "left-face" value $(-m\pi)$ with the "right-face" value $(+m\pi)$, we see that the cosine, being an even function, gives the same value on both faces, whereas the sine, which is odd, gives only the value zero on both faces. Hence the elements of U on the left and right faces can be paired, since they are identical. This means that for every column in U, there is another, identical column, thus rendering U degenerate. The set of basis functions defined in (5.7) even violates the conditions of the classical Stone–Weierstrass theorem [20, p. 9] in that the basis set *does not separate the points on* Γ, i.e., there are distinct points (u_1, v_1) and (u_2, u_2) on the *boundary* of each cell, for which it is not possible to find a basis function ϕ such that $\phi(u_1, v_1) \neq \phi(u_2, u_2)$. Thus, the functions $\{\phi_{mn}(u, v)\}$ cannot even be dense (in the limit) in the space of continuous functions on the cell perimeter. This example illustrates the importance of a basis set which can be dense in $H^0(\Omega_k)$ *and* in $H^0(\Gamma_k)$.

If we now *reduce* the base interval to $[-\pi/2, \pi/2]$, the situation changes. The results for Types 1 and 2, as shown in Tables 3d and 3e, are respectable, although quite different from the polynomial results. However, when the base interval is reduced further to the "unusual" value of $[-\pi/4, \pi/4]$, the results shown in Table 3f are at first similar to those in Tables 3d and 3e, until L takes the value 3, when the *exact* solution is suddenly achieved! The reason for this strange occurrence is as follows: The true solution of the problem is $\psi = A \sin \pi x/2 \times \sin \pi y/2$, as remarked previously. In Ω_1, the x and y values satisfy $0 \leqq x, y \leqq 1$, so that when we switch to the base variables u and v, say, whose values satisfy $-\pi/4 \leqq u, \; v \leqq \pi/4$, the appropriate transformation is

$$x = \frac{2}{\pi}\left(u + \frac{\pi}{4}\right), \qquad y = \frac{2}{\pi}\left(v + \frac{\pi}{4}\right). \tag{5.9}$$

Hence ψ_1 becomes:

$$\begin{aligned}\psi_1(u, v) &= A \sin\left(u+\frac{\pi}{4}\right)\sin\left(v+\frac{\pi}{4}\right)\\ &= A\left(\sin u \cos\frac{\pi}{4}+\cos u \sin\frac{\pi}{4}\right)\left(\sin v \cos\frac{\pi}{4}+v \sin\frac{\pi}{4}\right) \qquad (5.10)\\ &= \frac{A}{\sqrt{2}}\{\sin u \sin v+\sin u \cos v+\cos u \sin v+\cos u \cos v\},\end{aligned}$$

which is a linear combination of products of the first three factor functions shown in (5.1c). It is only when $L=3$ that the weight functions include $\sin u$ and $\cos u$, thus forcing the expansion to fit the solution exactly. The same argument can be made for the other three cells. For higher values of L, there is nothing to prevent the CD approximation from fitting the true solution exactly, except the obvious appearance of rounding error effects. This is also a very good example of the effect, on the solution, of the "approximating power" of the basis set and the weight functions for a particular problem.

Case 2. *Neumann problem.* We shall illustrate the utility of the "nullifier" boundary correction alluded to before (and described in detail in Appendix C of [12]). We shall try the simple problem of solving Laplace's equation in the square $-1 \leqq x,\ y \leqq 1$, with the boundary conditions:

$$\frac{\partial\psi}{\partial x}(-1, y)=\frac{\partial\psi}{\partial x}(+1, y)=1, \qquad \frac{\partial\psi}{\partial y}(x, -1)=\frac{\partial\psi}{\partial y}(x, +1)=0. \qquad (5.11)$$

When this is solved *without* the boundary-condition correction—using a power expansion in the base interval $[-1, 1]$, four equal cells, Type 1 expansions and $L=3$—the result is (coefficients are correct to 4 figures):

$$\begin{aligned}\psi_{1,3} &= .04336-.2398x-1.874x^2-3.749x^3-2.187x^4,\\ \psi_{2,4} &= .04336=.2398x+1.874x^2-3.749x^3+2.187x^4\end{aligned} \qquad (5.12)$$

where we have written $\psi_{1,3}$ to mean "ψ_1 and ψ_3", etc. These solutions (correctly) do not depend on y. Although they are far from the exact solution (which is $\psi=x+$ constant), the values of the boundary normal derivatives are almost exact:

$$\left(\frac{\partial\psi_{1,3}}{\partial x}\right)_{x=-1}=\left(\frac{\partial\psi_{2,4}}{\partial x}\right)_{x=+1}=0.9992. \qquad (5.13)$$

Note also that $\psi_{1,3}(0, y)=\psi_{2,4}(0, y)$. The difference $(\max\psi-\min\psi)$ is .125, which indicates that ψ is very flat. It would seem that the variational procedure is attempting to approximate the solution $\psi=$ constant, which would minimize Φ by making it vanish—except that at $x=-1$ and $x=1$, the slope is required to be unity. This strange "solution" results from the fact that, without the boundary-condition correction, this is not a properly posed problem and, in reality, has *no* solution.

However, when the b.c. correction *is* included, the solution turns out to be

$$\psi_{1,3}=\psi_{2,4}=x \qquad \text{(to machine precision)}, \qquad (5.14)$$

which is one of the exact solutions.

Case 3. *Eigenproblem in L-shaped domain.* We shall now consider a problem that does not have a known analytic solution. It is the same eigenvalue problem as

Case 1, except that cell 4 is deleted, leaving an L-shaped domain. We have used a Legendre polynomial expansion of Type 1, which appears to be the most resistant to rounding error and, indeed, all of our subsequent examples will be solved with this basis set. In Table 4 is shown the sequence of eigenvalues for various values of L

TABLE 4
Case 3. L-shaped domain Legendre polynomials (Type 1).

L	M	λ
1	6	7.310712
2	15	9.479742
3	15	9.757295
4	28	9.644506
5	28	9.690872
6	45	9.652736
7	45	9.667149
8	66	9.650338
9	66	9.656302
10	91	9.647602

(and M). The true value is given as $\lambda = 9.63972$ by J. A. George in [5]. The lack of agreement with George's result is no doubt a result of the singularity in the solution at the reentrant corner (at $(1, 1)$), which causes it to be poorly approximated by polynomials. Hence, of course, the eigenvalue is also ill-determined. Because of the generality of the basis functions allowed in CD, the singular factor $r^{2/3}$ (where r is the distance to the reentrant corner) could have been incorporated. This procedure would address the problem of spanning "rough" spaces on the boundary, which was touched on in the previous section. However, we must again plead the inadequacy of our program in not including this feature.

Case 4. *Effect on accuracy of mesh refinement.* Although our main interest in this work has not centered on the change of accuracy with cell size, it is instructive to observe the effect for the simple square-shaped and L-shaped eigenvalue problems of successive halvings of the cell face lengths. For each computation, we assess the rough error (as defined before), by noting the maximum departure of the solution from zero on the boundary.

We have solved the eigenvalue problem for values of L (the number of interface moments) running from 1 to 5. The values of M (the total number of intracell coefficients) were chosen so as to prevent degeneracy, and to make the progression of maximum errors as smooth as possible, as far as the L-values were concerned. The initial value of unity for h, the cell side, was halved repeatedly, until there were 5 values for h. This set of values for L and h gives a sample of 25 cases for each problem.

The results for the square are given in Table 5a. We have included in this table the total number of interface variables for which we had to solve (i.e., all the σs). As can be seen, this number can get quite high. In spite of this, the generalized conjugate gradient method takes a very small number of iterations to get the estimated ψ-error low enough ($\sim$a factor of 0.1) for each new estimate of λ. For each case, the mean number of iterations was calculated, and appears in the table. This excellent rate of convergence is probably a result of the preconditioning effect of the diagonal coefficient matrices $\{\Lambda\}$, as shown in (3.17). It would be interesting to observe numerically what happens to the eigenvalues of the iteration matrix before and after this preconditioning, but we have not yet had the opportunity for such a study.

In order to test whether the error is substantially influenced by the regularity of the cell structure, the case for $L=2$ was redone, for all 5 h-partitions, but with randomly generated hs. These values were produced independently for the x- and y-directions, with the only restriction imposed that the ratio of largest to smallest h was not allowed to exceed 10. As can be seen from the last section of Table 5a, the effects of the nonuniform partition showed up essentially only in the average number of iterations. The "$h$max" is the largest h in both directions. In those cases where the error seems to be smaller than for the uniform mesh, it is probable that this is due to the sometimes smaller hs at the corners, where the worst error generally occurs. A similar study for the L-shape, shown in Table 5b, reveals that not only is λ not

TABLE 5a
Case 4. Parameter study for square eigenvalue problem (various L and h values).

L M ↓	h:	1	.5	.25	.125	.0625
	Tot. vars.	4	24	112	480	1984
1	Avg. iter.	1	3	9	21	47
15	λ	4.117036	4.695822	4.872347	4.919009	4.930842
	Max. err.	4.0(−1)	1.4(−1)	3.8(−2)	9.6(−3)	2.4(−3)
	Tot. vars.	8	48	224	960	3968
2	Avg. iter.	2	5	9	21	43
21	λ	4.924406	4.934309	4.934774	4.934801	4.934802
	Max. err.	2.6(−2)	1.3(−3)	7.6(−5)	4.7(−6)	2.9(−7)
	Tot. vars.	12	72	336	1440	5952
3	Avg. iter.	3	8	18	40	90
28	λ	4.934768	4.934804	4.934802	4.934802	4.934802
	Max. err.	1.7(−3)	1.1(−4)	7.6(−6)	4.9(−7)	3.0(−8)
	Tot. vars.	16	96	448	1920	7936
4	Avg. iter.	3	7	15	28	62
36	λ	4.934805	4.934802	4.934802	4.934802	4.934802
	Max. err.	3.3(−5)	9.8(−7)	4.2(−8)	6.4(−10)	4.4(−11)
	Tot. vars.	20	120	560	2400	9920
5	Avg. iter.	4	10	21	47	106
45	λ	4.934802	4.934802	4.934802	4.934802	4.934802
	Max. err.	3.0(−6)	8.4(−8)	2.9(−9)	3.3(−10)	6.8(−11)

With randomly selected cell partitions ($L=2$ only)

L M ↓	$h$max:	1.5	.67	.39	.21	.12
	Tot. vars.	8	48	224	960	3968
2	Avg. iter.	4	9	17	45	75
21	λ	4.873809	4.933892	4.934714	4.934798	4.934801
	Max. err.	9.6(−2)	2.2(−3)	7.0(−5)	1.9(−5)	1.6(−7)

TABLE 5b
Case 4. Parameter study for L-shape problem (various L and h values).

L M ↓	h:	1	.5	.25	.125	.0625
	Tot. vars.	2	16	80	352	1472
1	Avg. iter.	1	5	12	24	50
15	λ	7.127977	8.677621	9.323416	9.535168	9.603818
	Max. err.	1.03	.71	.42	.25	.16
	Tot. vars.	4	32	160	704	2944
2	Avg. iter.	2	7	12	23	47
21	λ	9.479742	9.577621	9.613936	9.629377	9.635605
	Max. err.	.37	.28	.18	.11	.072
	Tot. vars.	6	48	240	1056	4416
3	Avg. iter.	2	10	22	47	95
28	λ	9.631882	9.625882	9.633852	9.637379	9.638792
	Max. err.	.21	.15	.096	.061	.038
	Tot. vars.	8	64	320	1408	5888
4	Avg. iter.	3	9	17	32	64
36	λ	9.644506	9.639103	9.639428	9.639603	9.639676
	Max. err.	.14	.095	.060	.038	.024
	Tot. vars.	10	80	400	1760	7360
5	Avg. iter.	2	13	26	53	110
45	λ	9.654209	9.643454	9.641149	9.640288	9.639948
	Max. err.	.086	.062	.039	.025	.015

well-defined numerically, but the maximum error in ψ, which always occurs at the reentrant corner, is very large, and really does not decrease much.

In order to try to throw some light on this behavior, we attempted straight-line fits of the logarithm of the error versus the logarithm of h, on the assumption that the error would turn out to be proportional to a power of h, as is true for the finite-difference and FEM methods. This assumption turned out to be very accurate, as is revealed by the results shown in Table 6. The actual equation used has the form

$$-\log_{10}\varepsilon = a + b(-\log_{10}h) \tag{5.15}$$

where ε is the maximum error in ψ at the boundary. The best-fit values of a and b are shown, as is the correlation coefficient r of each regression. For $L > 1$, the error seems to depend on the 4th power of h in the square case, although this varies for particular values of L and M.

On the other hand, it is extremely interesting that for the L-shape problem the error appears to depend on a power of h which is suspiciously close to $\frac{2}{3}$! It seems reasonable that, if the factor $r^{2/3}$ (r being the distance from the reentrant corner) had been incorporated into the basis functions (and the appropriate *inverse* power into the weight functions) the dependence of ε on h would then have been the same as in the square case. This is a fascinating supposition which, alas, we cannot test at this time.

The rest of our cases are simple examples taken out of L. Collatz's well-known book [1].

TABLE 6
Case 4. Log-log *regressions of error vs. h.*

L	a	b	r
Uniformly partitioned square			
1	.3400	1.8437	.99842
2	1.6400	4.0528	.99987
3	2.8000	3.8867	.99989
4	4.5000	4.9829	.99867
5	5.8000	3.9199	.98773
Randomly partitioned square			
2	1.8892	4.9870	.98629
L-shape			
1	−.0318	.68963	.99871
2	.39800	.60791	.99580
3	.66000	.60459	.99837
4	.83800	.62458	.99932
5	1.0420	.61788	.99718

Case 5. *Temperature distribution* [1, p. 361]. This example, shown in Fig. 3, is noteworthy in that the boundary values have two jumps, one at each of the upper corners. We have solved this case for $L=4$ $(M=28)$ with the domain divided by vertical lines at $x=\frac{1}{3}$ and $\frac{2}{3}$, and with a horizontal line at $y=\frac{1}{2}$, giving a total of 6 cells. The solution values are given in Table 7a only for $x=\{\frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1\}$ (because the solution is symmetric around $x=\frac{1}{2}$) and for $y=\{0, \frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}, 1\}$. Clustered values occur in the table when the output points fall on an interface; the values may be different in the 2 (or 4) different cells which happen to abut the point. The discontinuity of the solution may easily be seen, as may a kind of "Gibbs phenomenon" near the corners where the boundary values are discontinuous.

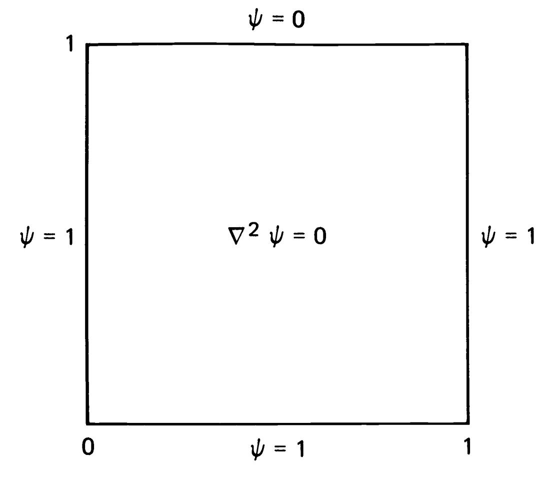


FIG. 3

These effects are mitigated by choosing the dividing lines at $x = .1$ and $.9$, and at $y = .9$. The upper corners where the discontinuities lie are thus isolated in two small cells, and the remaining large cells have only "smooth" interface conditions. The results (Table 7b) show less of the Gibbs effect, and the values agree better with those given by Collatz (Table 7c), which were computed by a standard difference method.

TABLE 7a
Case 5. Temperature distribution—6 equal cells.

x	$\frac{1}{2}$	$\frac{2}{3}$		$\frac{5}{6}$	1
y					
0	1.004	1.010	.9938	.9962	.9857
$\frac{1}{6}$	.9371	.9415	.9446	.9729	1.009
$\frac{1}{3}$	.8659	.8866	.8829	.9265	.9902
$\frac{1}{2}$	.7462	.7696	.7328	.9168	.9299
	.7615	.8105	.7406	.8972	.8176
$\frac{2}{3}$	.5706	.5961		.7730	1.075
$\frac{5}{6}$	.3244	.3692	.4010	.5088	.9148
1	.003372	.008991	.1281	.08733	.3377

TABLE 7b
Case 5. 6 unequal cells.

x	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{5}{6}$	1
y				
0	1.002	.9993	.9994	.9994
$\frac{1}{6}$	.9367	.9467	.9735	1.011
$\frac{1}{3}$	.8631	.8825	.9303	1.010
$\frac{1}{2}$	.7495	.7800	.8690	.9726
$\frac{2}{3}$	.5764	.6177	.7589	1.007
$\frac{5}{6}$	.3249	.3635	.5278	1.017
1	.04311	−.01658	−.01372	.5018

TABLE 7c.
Case 5. Collatz' values.

x	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{5}{6}$	1
y				
0	1	1	1	1
$\frac{1}{6}$	.9387	.9466	.9687	1
$\frac{1}{3}$	.8616	.8789	.9292	1
$\frac{1}{2}$	.7501	.7789	.8654	1
$\frac{2}{3}$	.5809	.6213	.7545	1
$\frac{5}{6}$	.3306	.3708	.5313	1
1	0	0	0	—

Case 6. *Mixed boundary-condition temperature distribution* [1, p. 410]. This case offers another opportunity to use the boundary compensation procedure used in Case

2. The problem is fully explained by Fig. 4, and the calculation was done with $L=4$ and 4 equal cells. Collatz does not give a table of values for this case, but only various approximate values of $\psi(0,0)$, of which the most accurate appears to be $\psi(0,0)=.82156$. Without the b.c. correction, CD gives the value 51.33, but with the correction, the CD value is .8217, which agrees very well with Collatz.

Case 7. *Acoustic vibration in a room* [1, p. 451]. In this 3-dimensional problem, whose geometry is shown in Fig. 5, we wish to find the fundamental mode of vibration

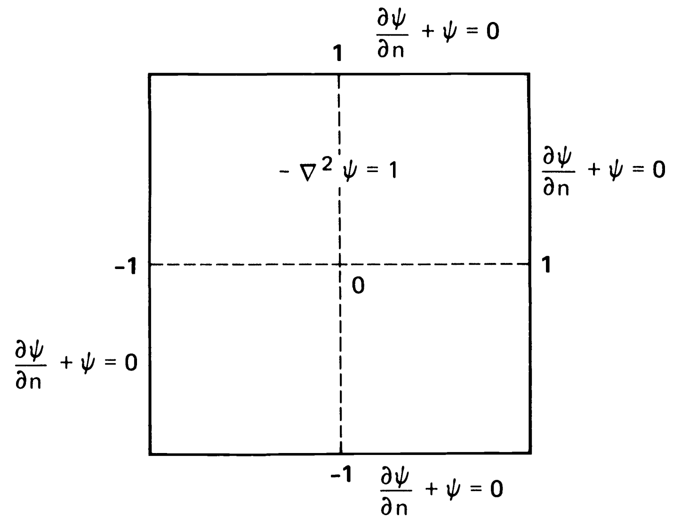

$\frac{\partial\psi}{\partial n}+\psi=0$
1
$-\nabla^2\psi=1$
$\frac{\partial\psi}{\partial n}+\psi=0$
-1
1
0
$\frac{\partial\psi}{\partial n}+\psi=0$
-1
$\frac{\partial\psi}{\partial n}+\psi=0$

FIG. 4

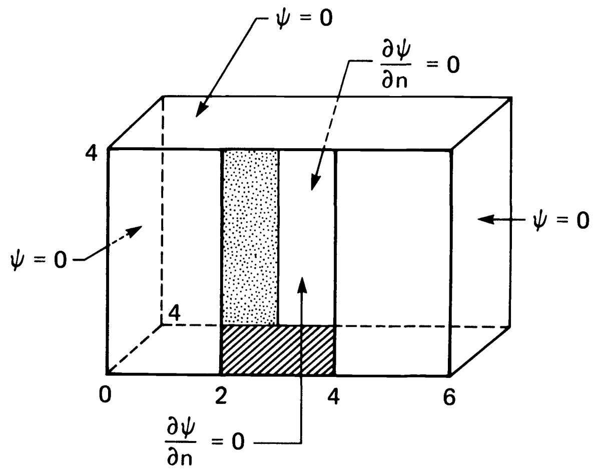

$\psi=0$
$\frac{\partial\psi}{\partial n}=0$
4
$\psi=0$
$\psi=0$
4
0
2
4
6
$\frac{\partial\psi}{\partial n}=0$

FIG. 5

of the air in the boxlike room with one doorway from floor to ceiling in each of the front and back walls. Therefore, we must solve the Helmholtz equation:

$$-\left(\frac{\partial^2\psi}{\partial x^2}+\frac{\partial^2\psi}{\partial y^2}+\frac{\partial^2\psi}{\partial z^2}\right)=\lambda\psi \tag{5.16}$$

with $\psi = 0$ on walls, ceiling and floor, and $\partial\psi/\partial n = 0$ in the two doorways. By setting

$$\psi = P(x, y)\sin\frac{\pi z}{4} \tag{5.17}$$

we automatically have $\psi(x, y, 0) = \psi(x, y, 4) = 0$, and (5.16) reduces to

$$-\left(\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2}\right) + \left(\frac{\pi}{4}\right)^2 P = \lambda P, \tag{5.18}$$

with

$$\begin{aligned} &P(0, y) = P(6, y) = 0, \\ &P(x, 0) = P(x, 4) = 0, \qquad 0 \leqq x \leqq 2, \quad 4 \leqq x \leqq 6, \\ &\frac{\partial P}{\partial y}(x, 0) = \frac{\partial P}{\partial y}(x, 4) = 0, \qquad 2 \leqq x \leqq 4. \end{aligned} \tag{5.19}$$

With interface divisions at $x = 2$ and 4, $y = 2$, and $z = 2$, we form 12 cells in 3D and 6 cells in 2D. To apply interface moment collocation up to the third order, we take $L = 4$ in 2D, and $L = 10$ in 3D. The results are:

$$\lambda = 1.238138 \qquad \text{(2D cells)}, \tag{5.20a}$$

$$\lambda = .7329 + \left(\frac{\pi}{4}\right)^2 = 1.3498 \qquad \text{(Collatz, lowest mode)}, \tag{5.20b}$$

$$\lambda = 1.228986 \qquad \text{(3D cells)}. \tag{5.20c}$$

In this example, we would not expect Collatz' result to be very accurate, since his mesh spacing was unity, so that only 15 interior nodes were used for his 2D calculation; the CD solution is almost certainly better.

6. Concluding remarks. In the description of cell discretization just presented, we noted a few directions that extension and generalization could take. We wish now to amplify these observations.

The most obvious next step is, of course, the programming of CD for irregular geometries. This is no mean task, mostly because the handling of irregular geometrical objects in a computer is inherently complicated and difficult. An additional feature, which should be in a "next" program, is the facility for having different *kinds* of basis functions in different cells, as well as the facility for having weight functions which are not so closely related (as to functional form) to the intracell basis functions. This, of course, includes the possibility of having functions which are nonanalytic at various important points (as described for the L-shape problem, for example).

The basic approach of CD can obviously be applied to partial differential equations, or systems of equations which are of higher order, or not self-adjoint (by the use of dual dependent variables), or which involve time. Moreover (although no work has actually been done on this problem), there does not seem to be any real difficulty in applying CD to integro-differential equations, such as Boltzmann's. In this case, the interface moment collocation could be done in phase space, rather than just in configuration space, with the result that the splitting of the velocity space into "outward" and "inward" directions would be quite natural in terms of cell specification. In this way, some of the problems which arise in neutron or photon transport calculations, arising from the necessity to approximate discontinuous distribution functions (across boundaries) by means of continuous functions (usually polynomials) alone would be avoided.

The CD approach can even be applied to the calculation of molecular wave functions in quantum chemistry. The classical scheme of Roothaan [16] has been expressed in terms of the cell method, with most of the details worked out (unpublished). Interestingly, it turns out there are no 3- or 4-center integrals appearing in this formulation, but difficult integrations over irregular volumes and surfaces arise in their place. However, if the cell interfaces are placed away from the nuclear centers, the interface collocation can then be done in places where the wave functions do not vary so rapidly as they do near the nuclei.

Finally, we should mention another method, developed by Delves and Hall [4], which bears a closer resemblance to CD than does any other method. They have integrated the additional terms in the functional (which we referred to as a "nullifier") into their basic method, and do not regard them as an afterthought. The result is a very natural treatment of the interface problem, with which they have been able to treat quite irregular geometries.

Acknowledgment. Over many years, I have received useful advice, criticism, encouragement and discussion from many people, among them R. A. Blaine, A. Bomberault, P. G. Comba, F. Engel, G. Golub, S. Hahn, D. A. Quarles, E. Wachspress and H. H. Wang. To all of them I express my gratitude. Special thanks are due to Professors A. R. Mitchell and Mary F. Wheeler for their encouragement at critical times.

REFERENCES

[1] L. COLLATZ, *The Numerical Treatment of Differential Equations*, Springer-Verlag, New York, 1966.

[2] 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.

[3] R. COURANT, *Variational methods for the solution of problems of equilibrium and vibrations*, Bull. Amer. Math. Soc., 49 (1943) pp. 1–23.

[4] L. M. DELVES AND C. A. HALL, *An implicit matching principle for global element calculations*, J. Inst. Math. Appl., 23 (1979), pp. 223–234.

[5] J. A. GEORGE, *Computer implementation of the finite element method*, Stanford Univ. Rep. STAN-CS-71-208, Feb. 1971.

[6] J. GREENSTADT, *On the reduction of continuous problems to discrete form*, IBM. J. Res. Develop., 3 (1959), pp. 355–363.

[7] ———, *Cell discretization I—Variational basis*, IBM New York Scientific Center Rep. 320–2944, Feb. 1967.

[8] ———, *Cell discretization II—Treatment of discrete equations (elliptic case)*, IBM New York Scientific Center Rep. 320–2909, Aug. 1967.

[9] ———, *Cell discretization III—Treatment of discrete equations (hyperbolic and parabolic cases)*, IBM New York Scientific Center Rep. 320-2914, Sept. 1967.

[10] ———, *Cell discretization*, in Conference on Applications of Numerical Analysis, J. H. Morris, ed., Springer-Verlag, New York, 1971.

[11] ———, *Some numerical tests of cell discretization*, IBM Palo Alto Scientific Center Rep. 320-3303, Aug. 1972.

[12] ———, *Cell discretization algorithm with numerical examples*, IBM Palo Alto Scientific Center Rep. G320-3405, May 1980.

[13] E. HEWITT AND K. STROMBERG, *Real and Abstract Analysis*, Springer-Verlag, New York, 1965.

[14] J. L. LIONS AND E. MAGENES, *Non-Homogeneous Boundary Value Problems and Applications*, Springer-Verlag, New York, 1972.

[15] J. T. ODEN AND J. N. REDDY, *An Introduction to the Mathematical Theory of Finite Elements*, Wiley–Interscience, New York, 1976.

[16] C. C. J. ROOTHAAN, *New developments in molecular orbital theory*, Rev. Modern Phys., 23 (1951), pp. 69 et seq.

[17] D. VAN DALEN AND A. F. MONNA, *Sets and Integration*, Wolters-Noordhoff, Groningen, 1972.

[18] G. STRANG AND G. FIX, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[19] A. WEINSTEIN AND W. STENGER, *Methods of Intermediate Problems for Eigenvalues*, Academic Press, New York, 1972.

[20] K. YOSIDA, *Functional Analysis*, 2nd ed., Springer-Verlag, New York, 1968.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

ON GENERATING ORTHOGONAL POLYNOMIALS*

WALTER GAUTSCHI†

Abstract. We consider the problem of numerically generating the recursion coefficients of orthogonal polynomials, given an arbitrary weight distribution of either discrete, continuous, or mixed type. We discuss two classical methods, respectively due to Stieltjes and Chebyshev, and modern implementations of them, placing particular emphasis on their numerical stability properties. The latter are being studied by analyzing the numerical condition of appropriate finite-dimensional maps. A number of examples are given to illustrate the strengths and weaknesses of the various methods and to test the theory developed for them.

Key words. orthogonal polynomials, recurrence relations for orthogonal polynomials, Stieltjes procedure, discretized Stieltjes procedure, Chebyshev algorithm, modified Chebyshev algorithm, condition numbers

1. Introduction.

Let $d\lambda(t)$ be a nonnegative measure on the real line $\mathbb{R}$, with compact or infinite support, for which all *moments*

$$B7\mu_r = \int_{\mathbb{R}} t^r \, d\lambda(t), \qquad r = 0, 1, 2, \cdots, \tag{1.1}$$

exist and are finite, and $\mu_0 > 0$. With $d\lambda$ there is associated a unique system of (monic) *orthogonal polynomials*, i.e., a system of polynomials $\pi_r = \pi_r(\cdot\,; d\lambda)$ such that

(i) $\pi_r(t)$ has exact degree r and leading coefficient 1,

(ii) $$\int_{\mathbb{R}} \pi_r(t)\pi_s(t)\, d\lambda(t) \begin{cases} >0 & \text{if } r = s, \\ =0 & \text{if } r \neq s. \end{cases}$$

In general, the system $\{\pi_r\}$ consists of infinitely many polynomials, but reduces to a finite number N of polynomials $\pi_0, \pi_1, \cdots, \pi_{N-1}$, if $\lambda(t)$ has exactly N points of increase. We denote such a measure by $d\lambda_N(t)$, and call it a *discrete measure* and the associated polynomials *discrete orthogonal polynomials.*

The problem we wish to consider is the actual (numerical) construction of the polynomials $\pi_r(\cdot\,; d\lambda)$, given an arbitrary measure $d\lambda(t)$. The problem has received surprisingly little attention in the literature, even though orthogonal polynomials originated in connection with concrete questions of applied analysis (e.g., numerical integration, least squares approximation, series expansions, continued fractions, etc.). The reasons for this are probably twofold: In the first place, much of the practical work involving orthogonal polynomials is based on special measures $d\lambda(t)$ of the classical types, for which the orthogonal polynomials are explicitly known and the constructive aspects are therefore trivial. Secondly, even in the case of general measures, our problem has a straightforward mathematical solution: It is well known how to express, or how to compute, orthogonal polynomials in terms of the moments (1.1). This point of view, in fact, is typical for the "pre-computer" era; when executed in finite precision on a computer, however, the approach via moments is utterly ineffective on account of the explosive growth of rounding errors. Other more effective procedures have been proposed and analyzed by Forsythe [6] for discrete orthogonal polynomials, in connection with least-squares data fitting, and more recently by Sack

* Received by the editors September 9, 1981, and in final form January 6, 1982. This research was sponsored in part by the National Science Foundation under grant MCS-7927158.

† Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47907.

and Donovan [25], Gautschi [11], [13] and Wheeler [29], for continuous distributions, in connection with the problem of Gaussian quadrature. It is our purpose, here, to bring this work into better historical perspective, to reorient it towards the problem of constructing orthogonal polynomials (rather than Gaussian quadrature rules), and to expand upon it and refine it in various directions.

First we must clarify what we mean by "constructing orthogonal polynomials." We take the position, here, that the fundamental quantities in the constructive theory of orthogonal polynomials are the coefficients in the basic three-term recurrence relation. As is well known, every system $\{\pi_r(\cdot\,; d\lambda)\}$ of (monic) orthogonal polynomials satisfies a recurrence relation of the form

$$\tag{1.2} \begin{aligned} &\pi_{k+1}(t)=(t-\alpha_k)\pi_k(t)-\beta_k\pi_{k-1}(t), \qquad k=0,1,2,\cdots, \\ &\pi_{-1}(t)=0, \qquad \pi_0(t)=1, \end{aligned}$$

where α_k, β_k are real constants with $\beta_k>0$. It is the coefficients α_k, β_k that provide the key for the constructive use of orthogonal polynomials. There are many reasons for this, chief among which are the following:

(i) The coefficients α_k, β_k provide a compact way of representing orthogonal polynomials, requiring only a linear array of parameters. The coefficients of orthogonal polynomials, or their zeros, in contrast need two-dimensional arrays.

(ii) Knowing the coefficients α_k, β_k it is easy to compute the polynomials π_r and their derivatives recursively by (1.2) and the relations obtained from (1.2) by differentiation, for any fixed t within or without the spectrum of $d\lambda$. The procedure is not only straightforward, but also quite stable, much in contrast to the evaluation of π_r in terms of its coefficients.

(iii) Equally simple is the evaluation of finite sums $\sum c_r\pi_r(t)$ of orthogonal polynomials by Clenshaw's algorithm (see, e.g., [14, § 1.5.3]).

(iv) The functions of the second kind,

$$\tag{1.3} \rho_r(z)=\int_{\mathbb{R}}\frac{\pi_r(t)}{z-t}\,d\lambda(t), \qquad r=0,1,2,\cdots,$$

where z is outside the spectrum of $d\lambda$, also satisfy the same recurrence relation as in (1.2) (where t is to be replaced by z), and in fact, under very weak assumptions on the measure $d\lambda$, represent the *minimal solution* of (1.2) normalized by $\rho_{-1}(z)=1$. They, too, therefore can be calculated accurately by known algorithms (cf. [18]).

(v) From the coefficients α_k, β_k we can construct the *Jacobi matrix* associated with $d\lambda$, i.e., the symmetric tridiagonal matrix

$$\tag{1.4} J=\begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & \\ & \sqrt{\beta_2} & \alpha_2 & \sqrt{\beta_3} \\ & \ddots & \ddots & \ddots \end{bmatrix},$$

which in turn allows us to compute the zeros of π_n rapidly and efficiently as eigenvalues of the nth order segment of J, using modern procedures of numerical linear algebra, notably the QR (or QL) algorithm. The first components of the corresponding eigenvectors, indeed, also yield immediately the Christoffel numbers associated with $d\lambda$ (see, e.g., [20], [16]).

(vi) The coefficients α_k, β_k enter in Jacobi's continued fraction associated with the integral $\int_{\mathbb{R}} d\lambda(t)/(z-t)$, as well as in the corresponding Stieltjes continued fraction (cf. [14, § 1.4.2]).

(vii) The coefficients β_k determine the normalization constants by virtue of $\int_{\mathbb{R}} \pi_r^2(t)\, d\lambda(t) = \beta_0\beta_1 \cdots \beta_r$ (cf. [3, Chapt. 1, Thm. 4.2(b)] or (2.2)).

We thus consider the following problem to be fundamental in the constructive theory of orthogonal polynomials: *Given $d\lambda(t)$, compute as many of the coefficients α_k, β_k in* (1.2) *as are desired.*

The next important question concerns the "codification" of the measure $d\lambda$: In what form should $d\lambda(t)$ be given or what do we assume known about $d\lambda$? The classical way of codifying $d\lambda$ is through its moments (1.1). The problem then becomes: *Given, for some integer $n > 0$, the first $2n$ moments $\mu_0, \mu_1, \cdots, \mu_{2n-1}$ of $d\lambda$, compute the first n coefficients $\alpha_k, \beta_k, k = 0, 1, \cdots, n-1$.* (It will be assumed throughout that $\beta_0 = \int_{\mathbb{R}} d\lambda(t) = \mu_0$, even though β_0 in (1.2) is arbitrary.) The solution of this problem gives us access to the first $n+1$ orthogonal polynomials $\pi_0, \pi_1, \cdots, \pi_n$.

There are several known procedures for solving this problem, of which two will be discussed in §§ 2.1 and 2.3. Unfortunately, the problem itself is highly sensitive to small perturbations in the moments, so that any algorithm which (theoretically) solves the problem will be subject to severe growth of errors when executed in finite precision. It is this unfortunate experience which motivates a careful study of the underlying (nonlinear) map $\mathbb{R}^{2n} \to \mathbb{R}^{2n}$, i.e., in the present case, the map from the first $2n$ moments μ_r to the first n recursion coefficients α_k, β_k. What we need to know is the *numerical condition* of this map, and of analogous maps for other related problems. This will be the subject of § 3, the particular map above being discussed in § 3.2. The novelty of our treatment, in part, consists in representing the respective map as a composition of two maps, the first being from the moments (or related quantities) to the Gaussian quadrature rule, the second from the Gaussian quadrature rule to the desired recursion coefficients. Each component map can be analyzed individually with regard to its numerical condition, which in turn yields a bound on the condition of the composite map.

A better codification of the measure $d\lambda$ was first proposed by Sack and Donovan [25] and involves the so-called *modified moments* $\nu_r = \int_{\mathbb{R}} p_r(t)\, d\lambda(t)$, where $\{p_r\}$ is an appropriate system of polynomials, often already orthogonal with respect to some classical measure. An algorithm that obtains the recursion coefficients from modified moments will be described in § 2.4. The condition of the underlying map is studied in § 3.3, where the principal result (Theorem 3.1) supersedes earlier results of ours, with regard to both generality and sharpness.

Methods based on the idea of discretizing the measure $d\lambda(t)$ were proposed in [11] and [19], and will be further dealt with in §§ 2.2, 2.5, and 3.4. They are applicable whenever $d\lambda$ has the form $d\lambda(t) = \omega(t)\, dt$, where ω is continuous on some open interval, or on the union of a finite number of open intervals, and zero on the complementary set in $\mathbb{R}$, whereby integrable singularities are allowed at the endpoints of the interval(s). The methods, in fact, are applicable to more general measures which in addition to the piecewise continuous component also contain a discrete point spectrum.

In § 4 the use and performance of the various methods discussed in § 2 will be illustrated by means of concrete and nontrivial examples involving orthogonal polynomials with respect to both discrete and continuous (also piecewise continuous) distributions $d\lambda$. Detailed comparisons are made between the actual performance of the algorithms and the expected performance based on the theory of § 3.

2. Basic procedures. There are essentially two classical procedures for generating the recursion coefficients of orthogonal polynomials. The first is based on the explicit inner product representation (2.2) of these coefficients, the constructive potential of which appears to have first been recognized by Stieltjes. We call the resulting method the *Stieltjes procedure.* The second method, due in the case of discrete orthogonal polynomials to Chebyshev, derives the desired coefficients directly from the moments of the underlying measure. We call this the *Chebyshev algorithm.*

Both procedures require substantial additional implementation work in order to make them effective tools of modern high-speed computation. The Stieltjes method can be implemented effectively by a discretization procedure proposed by Gautschi [11]. The resulting *discretized Stieltjes procedure,* especially in the refined form described at the end of § 2.2, is by far the most reliable and the most generally applicable procedure. Its major limitation is the possibility of relatively slow convergence, particularly in cases of integration measures with infinite support. Chebyshev's algorithm, in a more effective form involving modified moments, has been rediscovered by Sack and Donovan [25] and Wheeler [29]. We refer to their procedure as the *modified Chebyshev algorithm.* Its major difficulties are two-fold. First, there is the possibility of moderate to severe ill-conditioning, particularly, but not exclusively, in the case of infinite intervals of orthogonality. Secondly, the algorithm requires the accurate computation of modified moments, which is usually a highly nontrivial task. The latter difficulty can be alleviated to some extent by a suitable discretization, as is briefly proposed in [19, § 5.3] and further discussed in § 2.5. If modified moments are easily available and ill-conditioning poses no problem, the modified Chebyshev algorithm is certainly the method of choice, on account of its superior speed.

In the following subsections we present a more detailed description and discussion of each of these individual procedures. Applications to specific examples will be given in § 4.

2.1. Stieltjes procedure. It is well known that the system of (monic) polynomials orthogonal with respect to the measure $d\lambda(t)$ satisfies a three-term recurrence relation of the form

$$\begin{aligned} &\pi_{k+1}(t)=(t-\alpha_k)\pi_k(t)-\beta_k\pi_{k-1}(t), \qquad k=0,1,2,\cdots, \\ &\pi_{-1}(t)=0, \qquad \pi_0(t)=1, \end{aligned} \tag{2.1}$$

where

$$\begin{aligned} \alpha_k&=\frac{\int_{\mathbb{R}} t\pi_k^2(t)\,d\lambda(t)}{\int_{\mathbb{R}} \pi_k^2(t)\,d\lambda(t)}, \qquad k=0,1,2,\cdots, \\ \beta_k&=\frac{\int_{\mathbb{R}} \pi_k^2(t)\,d\lambda(t)}{\int_{\mathbb{R}} \pi_{k-1}^2(t)\,d\lambda(t)}, \qquad k=1,2,\cdots. \end{aligned} \tag{2.2}$$

In particular, $\beta_k>0$ for all $k\geqq 1$; β_0 in (2.1) is arbitrary, but is conveniently defined as $\beta_0=\int_{\mathbb{R}} d\lambda(t)$.

The general formulas (2.1) were given already by Christoffel [4], who has different expressions for the α_k, β_k. The formulas in (2.2) are due, independently, to Darboux [5, pp. 411–413] and Stieltjes [26, Oeuvres I, p. 382]. Stieltjes observed how (2.1), (2.2) can be used to successively generate $\pi_1, \pi_2, \pi_3, \cdots$. Indeed, since $\pi_0=1$, the first coefficient α_0 can be computed from (2.2) with $k=0$, which then allows us to obtain $\pi_1(t)$ from (2.1). Knowing π_0, π_1, we can get α_1, β_1 from (2.2), hence $\pi_2(t)$ from (2.1), etc. We call this procedure, alternating recursively between (2.1) and (2.2), the *Stieltjes procedure*. Stieltjes did not elaborate on how to evaluate the integrals in (2.2). Presumably, he considered this a straightforward task, since, given the first $2k+2$ moments in (1.1) and given the coefficients of π_k and π_{k-1}, it is an easy matter to compute α_k and β_k in terms of these quantities, and then to obtain the coefficients of π_{k+1} from (2.1).

Unfortunately, in this form Stieltjes's procedure is practically useless, since rounding errors propagate very rapidly. As pointed out in [11], the rapid growth of errors is a reflection of the highly ill-conditioned nature of the map from the first $2n$ moments $\mu_0, \mu_1, \cdots, \mu_{2n-1}$ to the first n coefficients $\alpha_0, \cdots, \alpha_{n-1}, \beta_0, \cdots, \beta_{n-1}$. We will have more to say about this in § 3.2.

If the measure $d\lambda(t) = d\lambda_N(t)$ is a discrete N-point measure, the integrals in (2.2) become sums and can be computed directly, without recourse to moments. In this case, the Stieltjes procedure, also publicized by Forsythe [6], is generally quite stable, although it may happen that the coefficients α_k, β_k, when k is approaching N, and N is large, will suffer in accuracy (cf. Examples 4.1 and 4.3).

2.2. Discretized Stieltjes procedure. Suppose, to begin with, that

$$d\lambda(t) = \omega(t)\,dt \quad \text{on } (-1, 1), \tag{2.3}$$

where ω is a nonnegative weight function with finite moments $\mu_r = \int_{-1}^{1} t^r \omega(t)\,dt$, $r = 0, 1, \cdots, 2n-1$, and $\mu_0 > 0$. Our objective is to compute α_k, β_k, $k = 0, 1, \cdots, n-1$. In an attempt to escape ill-conditioning in the Stieltjes procedure we proposed in [11] to approximate the integrals in (2.2) by a suitable quadrature rule,

$$\int_{-1}^{1} p(t)\omega(t)\,dt = \sum_{m=1}^{N} w_m p(t_m)\omega(t_m) + R_N(p\omega), \qquad N > n, \tag{2.4}$$

with nodes $t_m = t_m^{(N)} \in (-1, 1)$ and weights $w_m = w_m^{(N)} > 0$. We require this rule to converge as $N \to \infty$ whenever p is a polynomial. It is easily seen that this procedure amounts to approximating the desired orthogonal polynomials π_k, $k = 0, 1, \cdots, n$, by the discrete orthogonal polynomials $\pi_{k,N}$, $k = 0, 1, \cdots, n$, orthogonal with respect to the N-point measure $d\lambda_N(t)$ having abscissas $t_m^{(N)}$ and jumps $w_m^{(N)}\omega(t_m^{(N)})$. In fact, under the assumption made, i.e., under the assumption

$$\int_{-1}^{1} p(t)\,d\lambda_N(t) \to \int_{-1}^{1} p(t)\,d\lambda(t), \qquad N \to \infty, \quad \text{all } p \in \mathbb{P}, \tag{2.5}$$

one can prove that [11]

$$\pi_{k,N}(t) \to \pi_k(t) \quad \text{as } N \to \infty,$$

for each fixed k, $0 \leqq k \leqq n^-$.

An attractive quadrature rule to be used in (2.4) is Fejér's rule, i.e., the interpolatory quadrature rule with nodes at the Chebyshev points $t_m^{(N)} = \cos((2m-1)\pi/2N)$. Among the considerations favoring this choice are the following:

(i) $t_m^{(N)}$ and $w_m^{(N)}$ are expressible in closed form in terms of trigonometric functions (cf. [11], [12]). Accordingly, the Fejér rule can be computed more rapidly than, say, the Gauss–Legendre quadrature rule. Some relevant timings are given in Table 2.1, where the method of Golub and Welsch [20] was used to generate the Gaussian rule.

(ii) Convergence (2.5) takes place not only for continuous functions ω, but also for functions ω that have singularities at the endpoints of $[-1, 1]$, provided they are monotonic and integrable [10].

TABLE 2.1
Timings (in seconds) for generating the n-point Fejér and the n-point Gauss–Legendre quadrature rule on the CDC 6500.

n	Fejér	Gauss
10	2.0 (−3)	4.3 (−2)
20	6.0 (−3)	1.5 (−1)
40	2.2 (−2)	5.5 (−1)
80	7.8 (−2)	2.1 (0)
160	2.9 (−1)	7.5 (0)
320	1.1 (0)	2.9 (1)
640	4.5 (0)	1.1 (2)

(Numbers in parentheses indicate decimal exponents.)

(iii) The discrete polynomials $\pi_{k,N}$, or rather, their coefficients $\alpha_{k,N}$, $\beta_{k,N}$, can be generated efficiently by the modified Chebyshev algorithm (see § 2.4), which ought to be quite stable on account of the point spectrum consisting of Chebyshev points (cf. Example 4.2).

(iv) If $\omega \in \mathbb{P}_s$ then $R_N(p\omega) = 0$ for all $p \in \mathbb{P}_{2n-1}$, whenever $N \geqq 2n + s$, and consequently our discretization process gives exact answers, when $N \geqq 2n + s$, except for rounding errors.

Nevertheless, when singularities are present, convergence in (2.5) can be rather slow. An example in point is the case of square-root singularities, say $\omega(t) = \omega_1(t)(1-t^2)^{-1/2}$, where ω_1 is smooth on $[-1, 1]$. In this case, the Fejér rule converges too slowly to be of practical use. Much more effective is the Gauss–Chebyshev rule

$$\int_{-1}^{1} p(t)\omega_1(t)(1-t^2)^{-1/2}\,dt = \frac{\pi}{N}\sum_{m=1}^{N} p(t_m)\omega_1(t_m) + R_N(p\omega_1),$$

where again $t_m = \cos((2m-1)\pi/2N)$. Similarly, one may wish to apply a Gauss–Jacobi rule in cases where $\omega(t) = \omega_1(t)(1-t)^\alpha(1+t)^\beta$, $\alpha > -1$, $\beta > -1$ (cf. Example 4.10).

If the basic interval is not $[-1, 1]$, but $[a, b]$, $-\infty \leqq a < b \leqq \infty$, we select a monotone function ϕ (a linear function, if $[a, b]$ is finite) which maps $[-1, 1]$ on $[a, b]$ and use

$$(2.6) \qquad \int_a^b p(t)\omega(t)\,dt = \sum_{m=1}^{N} w_m p(\phi(t_m))\omega(\phi(t_m))\phi'(t_m) + R_N(p\omega)$$

in place of (2.4) (cf. [11]). This again leads to a discrete measure $d\lambda_N(t)$, the abscissas and jumps now being $\phi(t_m)$ and $w_m\omega(\phi(t_m))\phi'(t_m)$, respectively. In this way, arbitrary finite or infinite intervals can be handled.

The method described, actually, has still wider applicability. We may indeed allow $d\lambda(t)$ to be composed of a piecewise continuous weight distribution and a discrete distribution, whereby the former is supported on the union of a finite number of disjoint intervals, and the latter contains a finite number of distinct points. One or both of the extreme intervals of the piecewise continuous component may extend to infinity. To cope with this more general situation, all we need to do is to apply our discretization process, with suitable functions ϕ in (2.6), individually to each component interval, add up all the contributions, and then add to the resulting discrete measure the discrete measure of the given point spectrum. The convergence of the process, of course, is in no way affected by the addition of the given point spectrum, although its stability properties may be altered significantly (see Example 4.8).

As a simple example, suppose ω is piecewise constant on the continuous part of the spectrum. Then the process not only converges, but in fact is exact for $N \geqq 2n$, since each integral in (2.2) for $\alpha_k, \beta_k, 0 \leqq k \leqq n-1$, is integrated exactly by the composite N-point Fejér rule (cf. (iv) above; see also Example 4.7).

Our treatment of the piecewise continuous part of the spectrum can also be interpreted as expressing the weight function in the form of a sum of individual weight functions (each equal to zero, except in one of the component intervals). This suggests further generalizations, whereby the weight function is assumed to be a sum of weight functions, each supported on its own interval, and each treated by a separate quadrature rule. Some of these intervals may then in fact coincide. This will be very effective in cases where different components of the weight function (possibly on the same interval) must be dealt with by different quadrature rules; see Examples 4.6, 4.9 and 4.10 for illustrations.

2.3. Chebyshev algorithm. Chebyshev [2, Oeuvres I, p. 482], in the case of discrete orthogonal polynomials, observed that the coefficients α_k, β_k can be obtained directly in terms of the quantities

$$\sigma_{k,l} = \int_{\mathbb{R}} \pi_k(t) t^l \, d\lambda(t) \tag{2.7}$$

(for which Chebyshev used the symbol (k, l)), by means of

$$\begin{aligned} &\alpha_0 = \frac{\sigma_{0,1}}{\sigma_{0,0}}, \qquad \beta_0 = \sigma_{0,0}, \\ &\alpha_k = \frac{\sigma_{k,k+1}}{\sigma_{k,k}} - \frac{\sigma_{k-1,k}}{\sigma_{k-1,k-1}}, \\ &\beta_k = \frac{\sigma_{k,k}}{\sigma_{k-1,k-1}}, \end{aligned} \qquad k = 1, 2, 3, \cdots. \tag{2.8}$$

The σ_{kl}, in turn, can be generated recursively from the moments μ_l by [2, Oeuvres I, p. 482]

$$\begin{aligned} &\sigma_{kl} = \sigma_{k-1,l+1} - \alpha_{k-1}\sigma_{k-1,l} - \beta_{k-1}\sigma_{k-2,l}, \qquad l = k, k+1, \cdots, 2n-k-1, \\ &\sigma_{-1,0} = 0, \qquad \sigma_{0l} = \mu_l. \end{aligned} \tag{2.9}$$

Given the first $2n$ moments $\mu_0, \mu_1, \cdots, \mu_{2n-1}$, this will produce the first n coefficients $\alpha_0, \cdots, \alpha_{n-1}$ and $\beta_0, \cdots, \beta_{n-1}$. We refer to (2.8), (2.9) as the *Chebyshev algorithm.*

Being based on the moments μ_r, the algorithm unfortunately suffers from the same effects of ill-conditioning as does the Stieltjes procedure, when implemented in terms of moments. In many cases it is possible, however, to stabilize the algorithm by introducing modified moments in place of ordinary moments.

2.4. Modified Chebyshev algorithm. Let $\{p_k(t)\}$ denote a system of (monic) polynomials satisfying a recurrence relation

$$\begin{aligned} &p_{k+1}(t) = (t - a_k)p_k(t) - b_k p_{k-1}(t), \qquad k = 0, 1, 2, \cdots, \\ &p_{-1}(t) = 0, \qquad p_0(t) = 1, \end{aligned} \tag{2.10}$$

where a_k, b_k are assumed known. We then call

$$\nu_r = \int_{\mathbb{R}} p_r(t) \, d\lambda(t), \qquad r = 0, 1, 2, \cdots, \tag{2.11}$$

the *modified moments* of the measure $d\lambda$ relative to the polynomial system $\{p_r\}$. If $a_k = b_k = 0$ for all k, then $p_k(t) = t^k$, and the modified moments reduce to the ordinary moments (1.1).

Chebyshev's algorithm generalizes very naturally to the case of modified moments. One defines

$$\sigma_{kl} = \int_{\mathbb{R}} \pi_k(t) p_l(t)\, d\lambda(t), \tag{2.12}$$

and obtains the first n coefficients α_k, β_k, $k = 0, 1, \cdots, n-1$, from the first $2n$ modified moments ν_r, $r = 0, 1, \cdots, 2n-1$, by the following *modified Chebyshev algorithm.*

Initialization:

$$\begin{aligned} &\sigma_{-1,l} = 0, \qquad l = 1, 2, \cdots, 2n-2, \\ &\sigma_{0,l} = \nu_l, \qquad l = 0, 1, \cdots, 2n-1, \\ &\alpha_0 = a_0 + \frac{\nu_1}{\nu_0}, \\ &\beta_0 = \nu_0. \end{aligned} \tag{2.13$_0$}$$

Continuation: For $k = 1, 2, \cdots, n-1$

$$\begin{aligned} \sigma_{kl} &= \sigma_{k-1,l+1} - (\alpha_{k-1} - a_l)\sigma_{k-1,l} - \beta_{k-1}\sigma_{k-2,l} \\ &\quad + b_l \sigma_{k-1,l-1}, \qquad l = k, k+1, \cdots, 2n-k-1, \\ \alpha_k &= a_k + \frac{\sigma_{k,k+1}}{\sigma_{kk}} - \frac{\sigma_{k-1,k}}{\sigma_{k-1,k-1}}, \\ \beta_k &= \frac{\sigma_{kk}}{\sigma_{k-1,k-1}}. \end{aligned} \tag{2.13$_k$}$$

The algorithm is summarized schematically in Fig. 2.1, where the "computing star" shows which of the σ_{kl} (indicated by black dots) are related to one another in the

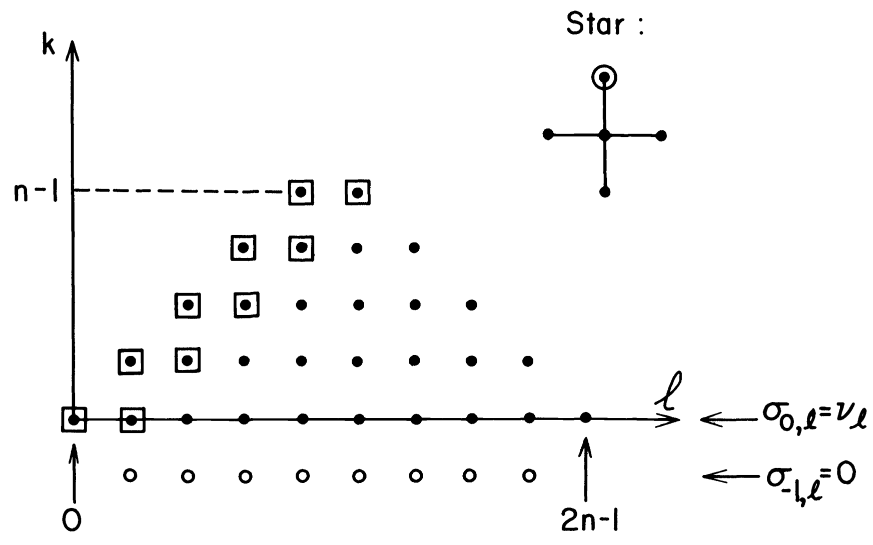

k
n-1
Star :
l
$\leftarrow \sigma_{0,l} = \nu_l$
$\leftarrow \sigma_{-1,l} = 0$
0
2n-1

FIG. 2.1. *The modified Chebyshev algorithm (schematically for $n = 5$).*

identity (2.13_k). The circled dot indicates the quantity which is computed in terms of the others in the star. The entries in boxes are those used to compute α_k, β_k. The diagonal entries, incidentally, furnish the normalization constants

$$\sigma_{kk} = \int_{\mathbb{R}} \pi_k^2(t)\, d\lambda(t), \qquad k = 0, 1, \cdots, n-1. \tag{2.14}$$

(Chebyshev's algorithm proceeds similarly, except that it starts from the moments μ_l, and the left arm of the computing star is missing.) The algorithm (2.13), in a somewhat different form, was first proposed by Sack and Donovan [25], and in the form given here by Wheeler [29]. A derivation can also be found in [15].

The modified Chebyshev algorithm often proves to be exceptionally stable, particularly when π_k and p_k are orthogonal polynomials on finite intervals (see, e.g., Examples 4.4 and 4.5). On infinite intervals, disjoint intervals, and also in the case of discrete spectra, the underlying map, however, is likely to become ill-conditioned, sometimes even severely so (see, e.g., Examples 4.1, 4.3, 4.6, 4.7 and 4.8). Some new theoretical insights into these questions of condition are given in § 3.3.

2.5. Discretized (modified) Chebyshev algorithm. The tendency of becoming ill-conditioned is one of the limitations of the modified Chebyshev algorithm. Another is the difficulty inherent in the accurate calculation of the modified moments (2.11). It is possible, however, as suggested in [19, § 5.3], to apply the same discretization $d\lambda(t) \approx d\lambda_N(t)$ that was used in the Stieltjes procedure (cf. § 2.2) to the modified moments. One thus approximates ν_r by

$$\nu_{r,N} = \int_{\mathbb{R}} p_r(t)\, d\lambda_N(t), \tag{2.15}$$

and then calculates the associated recursion coefficients $\alpha_{k,N}$ and $\beta_{k,N}$ by the modified Chebyshev algorithm. The procedure converges under the same conditions as the discretized Stieltjes procedure. It is essential, however, that convergence (in relative accuracy) be tested on the $\beta_{k,N}$, and not on the $\nu_{r,N}$, since the latter may vanish and, besides, need not be required to have full relative precision (cf. Example 4.4). The range of applicability of this procedure can be extended, as in the case of the discretized Stieltjes procedure, to measures $d\lambda(t)$ composed of piecewise continuous components as well as point spectra. It is important to realize, however, that any ill-conditioning present in the modified Chebyshev algorithm will manifest itself also in its discretized version. There are fewer problems of this kind with the discretized Stieltjes procedure.

3. Questions of numerical condition.

The modified Chebyshev algorithm of § 2.4 realizes the map $K_n : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ which associates to the first $2n$ modified moments ν_r the recursion coefficients $\alpha_k, \beta_k, k = 0, 1, \cdots, n-1$, for the respective orthogonal polynomials:

$$\begin{gathered} K_n : \nu \to \rho, \\ \nu^T = [\nu_0, \nu_1, \cdots, \nu_{2n-1}], \qquad \rho^T = [\alpha_0, \cdots, \alpha_{n-1}, \beta_0, \cdots, \beta_{n-1}]. \end{gathered} \tag{3.1}$$

Throughout this section we assume that $\beta_0 = \int_{\mathbb{R}} d\lambda(t) = \nu_0$.

For the purpose of studying the numerical condition of the map K_n, it is convenient to think of K_n as the composition of two maps,

$$K_n = H_n \circ G_n, \tag{3.2}$$

where G_n is the map from the modified moments ν_r to the Gauss–Christoffel quadrature rule,

$$G_n : \nu \to \gamma, \qquad \gamma^T = [\lambda_1, \cdots, \lambda_n, \tau_1, \cdots, \tau_n], \tag{3.3}$$

and H_n the map from the Gauss–Christoffel quadrature rule to the recursion coefficients,

$$H_n : \gamma \to \rho. \tag{3.4}$$

Here, $\lambda_\nu = \lambda_\nu^{(n)}$, $\tau_\nu = \tau_\nu^{(n)}$ are the Christoffel numbers and Gaussian abscissas (the zeros of $\pi_n(\cdot\,; d\lambda)$), respectively, associated with the measure $d\lambda(t)$, so that

$$\int_{\mathbb{R}} f(t)\, d\lambda(t) = \sum_{\nu=1}^{n} \lambda_\nu f(\tau_\nu) + R_n(f), \qquad R_n(\mathbb{P}_{2n-1}) = 0. \tag{3.5}$$

As one would expect, the map G_n is the more sensitive of the two. The map H_n is usually fairly well-conditioned; its condition is discussed in § 3.1. In § 3.2 we briefly recall the ill-conditioned nature of the map G_n when the vector ν consists of ordinary moments. The condition of G_n in the general case of modified moments is studied in § 3.3. As condition number of a map $M: x \to y$ from one finite-dimensional space into another we generally adopt the quantity (see, e.g., [15])

$$(\operatorname{cond} M)(x) = \frac{\|x\| \|J_M(x)\|}{\|y\|}, \qquad y = Mx, \tag{3.6}$$

where J_M is the Jacobian matrix of M, and $\|\cdot\|$ a suitable vector norm and subordinate matrix norm. In cases where the vector x (or y) has components of widely varying magnitude, this condition number may not be very meaningful on account of the falsification introduced by the factor $\|x\|$ (or $1/\|y\|$). In such cases we use more refined measures of the condition. Recall also that cond $(H_n \circ G_n) \leq$ cond (H_n) cond (G_n).

3.1. Condition of the map H_n. By virtue of (2.2) and (3.5), the map H_n can be described by

$$\begin{aligned} \alpha_k &= \frac{\sum_{\nu=1}^n \lambda_\nu \tau_\nu \pi_k^2(\tau_\nu)}{\sum_{\nu=1}^n \lambda_\nu \pi_k^2(\tau_\nu)}, \qquad k = 0, 1, 2, \cdots, n-1, \\ \beta_0 &= \sum_{\nu=1}^n \lambda_\nu, \\ \beta_k &= \frac{\sum_{\nu=1}^n \lambda_\nu \pi_k^2(\tau_\nu)}{\sum_{\nu=1}^n \lambda_\nu \pi_{k-1}^2(\tau_\nu)}, \qquad k = 1, 2, \cdots, n-1. \end{aligned} \tag{3.7}$$

We are here in a case where the condition number (3.6) is often not appropriate, since the Christoffel numbers λ_ν vary greatly in magnitude, particularly if the interval of orthogonality is infinite. We therefore use the one-dimensional equivalent of (3.6), applied to each α_k and β_k individually, considered as functions of one particular λ_ν or τ_ν. Thus, we write

$$(\operatorname{cond} \alpha_k)(\lambda_\nu) = \left| \frac{\lambda_\nu (\partial \alpha_k / \partial \lambda_\nu)}{\alpha_k} \right| \qquad (\text{if } \alpha_k \neq 0), \quad \text{etc.},$$

where in the case $\alpha_k = 0$ the division by α_k is omitted.

An elementary calculation then yields

$$
(3.8)\qquad
\begin{aligned}
&\sum_{\nu=1}^{n} (\operatorname{cond} \alpha_k)(\lambda_\nu) = d_k^{-1} \sum_{\nu=1}^{n} \lambda_\nu |\Delta_{\nu,k}| \pi_k^2(\tau_\nu),\\
&\sum_{\nu=1}^{n} (\operatorname{cond} \alpha_k)(\tau_\nu) = d_k^{-1} \sum_{\nu=1}^{n} \lambda_\nu |\tau_\nu| \, |\delta_k \pi_k^2(\tau_\nu) + 2\Delta_{\nu,k} \pi_k(\tau_\nu) \pi_k'(\tau_\nu)|,\\
&\sum_{\nu=1}^{n} (\operatorname{cond} \beta_k)(\lambda_\nu) = d_k^{-1} \sum_{\nu=1}^{n} \lambda_\nu |\pi_k^2(\tau_\nu) - \beta_k \pi_{k-1}^2(\tau_\nu)|,\\
&\sum_{\nu=1}^{n} (\operatorname{cond} \beta_k)(\tau_\nu) = 2d_k^{-1} \sum_{\nu=1}^{n} \lambda_\nu |\tau_\nu| \, |\pi_k(\tau_\nu) \pi_k'(\tau_\nu) - \beta_k \pi_{k-1}(\tau_\nu) \pi_{k-1}'(\tau_\nu)|,
\end{aligned}
$$

where

$$
d_k = \sum_{\nu=1}^{n} \lambda_\nu \pi_k^2(\tau_\nu) = \int_{\mathbb{R}} \pi_k^2(t)\, d\lambda(t), \qquad k = 0, 1, \cdots, n-1,
$$

$$
\delta_k = \begin{cases} \dfrac{1}{\alpha_k} & \text{if } \alpha_k \neq 0,\\[2mm] 1 & \text{if } \alpha_k = 0, \end{cases} \qquad
\Delta_{\nu,k} = \begin{cases} \dfrac{\tau_\nu - \alpha_k}{\alpha_k} & \text{if } \alpha_k \neq 0,\\[2mm] \tau_\nu & \text{if } \alpha_k = 0. \end{cases}
$$

A suitable condition number, cond H_n, for the map H_n is now the maximum of all the numbers in (3.8), as k varies over $0, 1, \cdots, n-1$. Numerical values of this condition number, for some classical polynomials, are shown in Table 3.1.

TABLE 3.1
The numerical condition of the map H_n for some classical orthogonal polynomials.

n	Legendre	Chebyshev	Laguerre	Hermite
5	6.968 (0)	7.186 (0)	6.724 (0)	1.596 (1)
10	1.785 (1)	1.823 (1)	2.143 (1)	6.254 (1)
20	4.530 (1)	4.742 (1)	4.269 (1)	2.042 (2)
40	1.071 (2)	1.135 (2)	8.525 (1)	6.181 (2)
80	2.526 (2)	2.644 (2)	1.761 (2)	1.807 (3)

It is seen that the map H_n in these cases is relatively well-conditioned, cond H_n growing about linearly in n. The well-conditioning of H_n, however, is not always assured; see, e.g., Examples 4.1, 4.3 and 4.8.

The coefficients α_k, β_k are less sensitive to perturbations in the λ_ν than to perturbations in the τ_ν. Indeed, from (3.8) it follows immediately that

$$
\sum_{\nu=1}^{n} (\operatorname{cond} \alpha_k)(\lambda_\nu) \leqq \max_{\nu,k} |\Delta_{\nu,k}|, \qquad \sum_{\nu=1}^{n} (\operatorname{cond} \beta_k)(\lambda_\nu) \leqq 2.
$$

If the polynomials are orthogonal on $[-1, 1]$ with respect to a symmetric weight function, then the bound in the first inequality is <1.

Results in terms of the "global" condition number (3.6) are comparable to those in Table 3.1 in the case of Legendre and Chebyshev polynomials, but completely unrealistic in case of Laguerre and Hermite polynomials. For Laguerre polynomials, e.g., the condition numbers based on (3.6), using the uniform norm, range from 1.06×10^5 for $n = 5$ to 7.65×10^{61} for $n = 40$. This is a rather striking example of how the introduction of inappropriate norms in the study of condition may completely distort the true nature of sensitivity.

3.2. Condition of the map G_n in case of ordinary moments. We assume in (3.3) that ν is the vector of ordinary moments,

$$\nu^T = [\mu_0, \mu_1, \cdots, \mu_{2n-1}], \qquad \mu_r = \int_{\mathbb{R}} t^r \, d\lambda(t).$$

The map G_n then amounts to solving the nonlinear system of equations,

$$\sum_{\nu=1}^{n} \lambda_\nu \tau_\nu^r = \mu_r, \qquad r = 0, 1, 2, \cdots, 2n-1. \tag{3.9}$$

If F_n is the map $\gamma \to \nu$ defined by (3.9), its Jacobian J_{F_n} is readily computed to be

$$J_{F_n} = T\Lambda,$$

where

$$\Lambda = \operatorname{diag}(1, \cdots, 1, \lambda_1, \cdots, \lambda_n),$$

$$T = \begin{bmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 \\ \tau_1 & \cdots & \tau_n & 1 & \cdots & 1 \\ \tau_1^2 & \cdots & \tau_n^2 & 2\tau_1 & \cdots & 2\tau_n \\ \vdots & & \vdots & \vdots & & \vdots \\ \tau_1^{2n-1} & \cdots & \tau_n^{2n-1} & (2n-1)\tau_1^{2n-2} & \cdots & (2n-1)\tau_n^{2n-2} \end{bmatrix}.$$

Since $J_{G_n} = J_{F_n}^{-1}$, we have according to (3.6),

$$(\text{cond } G_n)(\nu) = \frac{\|\nu\| \|\Lambda^{-1} T^{-1}\|}{\|\gamma\|}.$$

From the analysis in [11], in particular Theorem 2.1 and the discussion preceding it (which assumed [0, 1] as the support of $d\lambda(t)$, but extends easily to arbitrary measures on the *positive* real line), one gets

$$\text{cond } G_n \geqq \frac{\|\nu\|}{\|\gamma\|} \frac{1}{\max_\nu \lambda_\nu} \cdot \max_\nu \left\{ (1+\tau_\nu) \prod_{\mu \neq \nu} \left(\frac{1+\tau_\mu}{\tau_\nu - \tau_\mu} \right)^2 \right\},$$

or, equivalently,

$$\text{cond } G_n \geqq \frac{\|\nu\|}{\|\gamma\|} \frac{1}{\max_\nu \lambda_\nu} \cdot \frac{[\pi_n(-1)]^2}{\min_\nu \{(1+\tau_\nu)[\pi_n'(\tau_\nu)]^2\}}, \tag{3.10}$$

where $\|\nu\| = \max_{0 \leqq r \leqq 2n-1} \mu_r$, $\|\gamma\| = \max(\max_\nu \lambda_\nu, \max_\nu \tau_\nu)$ and supp $(d\lambda) \in \mathbb{R}_+$. Since the point -1 is well outside the spectrum of $d\lambda$, the lower bound in (3.10) must be expected to grow rapidly to infinity as $n \to \infty$, on account of the asymptotic behavior of orthogonal polynomials outside the spectrum (for relevant results see, e.g. [27, Thms. 8.21.7, 8.22.3]).

Here also, the result (3.10) may be misleading if the interval of orthogonality is infinite, since the moments μ_r then likely grow rapidly. In analogy to § 3.1, it is better, in these cases, to use the more refined condition numbers

$$\begin{aligned} \sum_{r=0}^{2n-1} (\operatorname{cond} \lambda_\nu)(\mu_r) &= \sum_{r=0}^{2n-1} \frac{|\mu_r (T^{-1})_{\nu,r+1}|}{\lambda_\nu}, \\ \sum_{r=0}^{2n-1} (\operatorname{cond} \tau_\nu)(\mu_r) &= \sum_{r=0}^{2n-1} \frac{|\mu_r (T^{-1})_{n+\nu,r+1}|}{|\lambda_\nu \tau_\nu|}, \end{aligned} \qquad \nu = 1, 2, \cdots, n, \tag{3.11}$$

and take as cond G_n the maximum of these $2n$ quantities. Unfortunately, they no longer admit simple expressions in closed form, but can be readily computed, for example by means of the algorithm for T^{-1} in [8, § 4].

In Table 3.2 we illustrate the condition of the map G_n for the examples $d\lambda(t) = dt$ on $[0, 1]$, $d\lambda(t) = \ln(1/t)\, dt$ on $[0, 1]$, $d\lambda(t) = e^{-t}\, dt$ on $[0, \infty]$ and $d\lambda(t) = e^{-t^2}\, dt$ on $[0, \infty]$. The third column gives the lower bound in (3.10), the fourth column the maximum of the $2n$ condition numbers in (3.11), and the last one the actual error growth observed. The latter is taken to mean the largest relative error in the α_k, β_k, $k = 0, 1, \cdots, n-1$, divided by the machine precision, in our case 3.553×10^{-15}. The coefficients α_k, β_k were computed by Chebyshev's algorithm; cf. § 2.3. (Since the moments of the Laguerre distribution are integers, we first subjected them to random perturbations at the level of the machine precision before applying Chebyshev's algorithm.) In the first two examples, for $n = 12$ one of the β_k (the last one) came out to be negative; hence no results are shown for $n = 14$. Note that the second and fourth example involve nonclassical orthogonal polynomials.

It is seen that in the first two examples, where the interval of orthogonality is finite, the observed error magnification indeed follows the trend predicted by either of the two condition numbers. In the last two examples, involving infinite intervals of orthogonality, this is only true for the condition number based on (3.11); the other grossly overestimates the error growth, for reasons explained earlier.

TABLE 3.2
The condition of the map G_n in the case of ordinary moments and $d\lambda(t) = \omega(t)\, dt$.

$\omega(t)$	n	cond G_n (3.10)	cond G_n (3.11)	err. growth
1	2	1.997 (1)	4.132 (1)	2.400 (1)
on $[0, 1]$	5	6.803 (4)	3.802 (5)	6.280 (4)
	8	7.080 (8)	6.161 (9)	8.977 (8)
	11	1.111 (13)	1.302 (14)	2.108 (13)
	14	–	–	–
$\ln(1/t)$	2	4.863 (1)	1.932 (1)	2.655 (1)
on $[0, 1]$	5	2.133 (5)	7.071 (4)	2.411 (4)
	8	2.391 (9)	7.370 (8)	8.010 (7)
	11	3.889 (13)	1.156 (13)	3.851 (12)
	14	–	–	–
e^{-t}	2	1.665 (1)	1.549 (1)	3.500 (0)
on $[0, \infty]$	5	4.416 (6)	9.665 (3)	5.991 (2)
	8	7.006 (13)	5.968 (6)	1.600 (5)
	11	1.078 (22)	3.829 (9)	6.508 (7)
	14	8.170 (30)	2.521 (12)	9.164 (9)
e^{-t^2}	2	6.823 (0)	2.106 (1)	1.162 (2)
on $[0, \infty]$	5	2.698 (4)	4.691 (4)	8.070 (3)
	8	8.044 (8)	1.073 (8)	3.890 (6)
	11	6.445 (13)	2.555 (11)	3.274 (10)
	14	1.001 (19)	6.243 (14)	5.373 (13)

3.3. Condition of the map G_n in case of modified moments. We now assume in (3.3) that the vector ν contains the modified moments,

$$\nu^T = [\nu_0, \nu_1, \cdots, \nu_{2n-1}], \qquad \nu_r = \int_{\mathbb{R}} p_r(t)\, d\lambda(t),$$

where $\{p_k\}$ is a system of (monic) polynomials orthogonal with respect to some measure $dl(t)$,

$$\int_{\mathbb{R}} p_r(t)p_s(t)\, dl(t) = 0, \qquad r \neq s.$$

The support of $dl(t)$ may be finite or infinite, and need not necessarily coincide with the support of $d\lambda(t)$. The condition of the map G_n in this case has previously been studied in [13]. Our treatment here improves upon that work in several respects. First, we obtain considerably more realistic bounds for the condition number. Secondly, our new bound is valid irrespectively of whether $dl(t)$ has finite or infinite support, in contrast, e.g., to Theorem 2.1 of [13]. Finally, the bound can be evaluated exactly by Gaussian quadrature, in contrast, e.g., to the bound (2.33) in [13], where $L_{n,2}$, and hence $k_n^{(2)}$, does not allow exact evaluation by quadrature. The improvement is achieved by employing the more natural L_2-norm in place of the L_1-norm used in [13], and rests on the fact that for any real matrix A,

$$\|A\|_2 = \sqrt{\rho(A^TA)} \leqq \sqrt{\operatorname{tr}(A^TA)} = \|A\|_F, \tag{3.12}$$

where $\rho(\cdot)$ denotes the spectral radius, $\operatorname{tr}(\cdot)$ the trace, and $\|\cdot\|_F$ the Frobenius norm

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}, \qquad A = [a_{ij}].$$

As before, we let $\tau_\nu = \tau_\nu^{(n)}$, $\lambda_\nu = \lambda_\nu^{(n)}$ denote the Gaussian abscissas and Christoffel numbers, respectively, belonging to the measure $d\lambda(t)$. Furthermore,

$$\begin{aligned} h_\nu(t) &= l_\nu^2(t)[1 - 2l_\nu'(\tau_\nu)(t-\tau_\nu)], \\ k_\nu(t) &= l_\nu^2(t)(t-\tau_\nu), \end{aligned} \qquad \nu = 1, 2, \cdots, n \tag{3.13}$$

will denote the fundamental Hermite interpolation polynomials associated with the abscissas $\tau_1, \cdots, \tau_n$, i.e.,

$$\begin{aligned} h_\nu(\tau_\mu) &= \delta_{\nu\mu}, & h_\nu'(\tau_\mu) &= 0, \\ k_\nu(\tau_\mu) &= 0, & k_\nu'(\tau_\mu) &= \delta_{\nu\mu}, \end{aligned} \qquad \nu, \mu = 1, 2, \cdots, n, \tag{3.14}$$

where $\delta_{\nu\mu}$ is the Kronecker symbol and l_ν are the fundamental Lagrange interpolation polynomials.

It will be convenient to consider not the map $\nu \to \gamma$ in (3.3), but the map

$$\tilde{G}_n : \tilde{\nu} \to \gamma,$$

where $\tilde{\nu}$ is the vector of *normalized* modified moments

$$\tilde{\nu}_r = d_r^{-1/2}\nu_r, \qquad d_r = \int_{\mathbb{R}} p_r^2(t)\, dl(t), \qquad r = 0, 1, \cdots, 2n-1.$$

This has the theoretical advantage of making the $\tilde{\nu}_r$ independent of the normalization of the orthogonal polynomials $\{p_k\}$. For algorithmic purposes, however, the passage from ν to $\tilde{\nu}$ is not required, and in fact not recommended; cf. § 2.4. The additional diagonal map introduced,

$$D_n : \nu \to \tilde{\nu},$$

of course, is harmless, since each individual transformation $\nu_r \to \tilde{\nu}_r$ involves just one multiplication and is therefore perfectly well-conditioned.

For the map $\tilde{G}_n$ we now have:

THEOREM 3.1. *The condition of the map $\tilde{G}_n$, in the sense of* (3.6), *with $\|\cdot\|$ the Euclidean norm, can be estimated as follows*:

$$(3.15) \qquad (\operatorname{cond} \tilde{G}_n)(\tilde{\nu}) \leq \frac{\|\tilde{\nu}\|}{\|\gamma\|} \left\{ \int_{\mathbb{R}} \sum_{\nu=1}^{n} \left(h_\nu^2(t) + \frac{1}{\lambda_\nu^2} k_\nu^2(t) \right) dl(t) \right\}^{1/2},$$

where

$$(3.16) \qquad \|\tilde{\nu}\|^2 = \sum_{r=0}^{2n-1} \tilde{\nu}_r^2, \qquad \|\gamma\|^2 = \sum_{\nu=1}^{n} (\lambda_\nu^2 + \tau_\nu^2).$$

Proof. The map $\tilde{G}_n$ amounts to solving the system of nonlinear equations

$$\Phi(\gamma) = \tilde{\nu},$$

where

$$\Phi_r(\gamma) = d_r^{-1/2} \sum_{\nu=1}^{n} \lambda_\nu p_r(\tau_\nu), \qquad r = 0, 1, \cdots, 2n-1.$$

The Jacobian $J_{\tilde{G}_n}(\tilde{\nu})$ of $\tilde{G}_n$, therefore, is the inverse of the Jacobian Φ_γ of Φ, so that

$$(3.17) \qquad (\operatorname{cond} \tilde{G}_n)(\tilde{\nu}) = \frac{\|\tilde{\nu}\|}{\|\gamma\|} \|\Phi_\gamma^{-1}(\gamma)\|.$$

An elementary calculation yields

$$\Phi_\gamma(\gamma) = D^{-1} P \Lambda,$$

where

$$D = \operatorname{diag}(d_0^{1/2}, d_1^{1/2}, \cdots, d_{2n-1}^{1/2}), \qquad \Lambda = \operatorname{diag}(1, \cdots, 1, \lambda_1, \cdots, \lambda_n),$$

and

$$P = \begin{bmatrix} p_0(\tau_1) & \cdots & p_0(\tau_n) & p_0'(\tau_1) & \cdots & p_0'(\tau_n) \\ p_1(\tau_1) & \cdots & p_1(\tau_n) & p_1'(\tau_1) & \cdots & p_1'(\tau_n) \\ \vdots & & \vdots & \vdots & & \vdots \\ p_{2n-1}(\tau_1) & \cdots & p_{2n-1}(\tau_n) & p_{2n-1}'(\tau_1) & \cdots & p_{2n-1}'(\tau_n) \end{bmatrix}.$$

Therefore, using (3.12),

$$(3.18) \qquad \|\Phi_\gamma^{-1}(\gamma)\| = \|\Lambda^{-1} P^{-1} D\| \leq \|\Lambda^{-1} P^{-1} D\|_F.$$

As previously observed in [13], the inverse of P can be expressed in terms of the expansion coefficients in

$$h_\nu(t) = \sum_{\mu=1}^{2n} a_{\nu\mu} p_{\mu-1}(t), \qquad k_\nu(t) = \sum_{\mu=1}^{2n} b_{\nu\mu} p_{\mu-1}(t)$$

as

$$P^{-1} = \begin{bmatrix} A \\ B \end{bmatrix}, \quad A = [a_{\nu\mu}], \quad B = [b_{\nu\mu}].$$

Since

$$(\Lambda^{-1}P^{-1}D)_{\nu,\mu}=d_{\mu-1}^{1/2}a_{\nu\mu},\qquad (\Lambda^{-1}P^{-1}D)_{\nu+n,\mu}=\frac{1}{\lambda_\nu}d_{\mu-1}^{1/2}b_{\nu\mu},$$

one obtains

$$\|\Lambda^{-1}P^{-1}D\|_F^2=\sum_{\nu=1}^{n}\sum_{\mu=1}^{2n}d_{\mu-1}\left(a_{\nu\mu}^2+\frac{1}{\lambda_\nu^2}b_{\nu\mu}^2\right).$$

On the other hand,

$$\begin{aligned}\int_{\mathbb{R}}h_\nu^2(t)\,dl(t)&=\int_{\mathbb{R}}\sum_{\mu=1}^{2n}a_{\nu\mu}p_{\mu-1}(t)\sum_{\kappa=1}^{2n}a_{\nu\kappa}p_{\kappa-1}(t)\,dl(t)\\&=\sum_{\mu,\kappa}a_{\nu\mu}a_{\nu\kappa}\int_{\mathbb{R}}p_{\mu-1}(t)p_{\kappa-1}(t)\,dl(t)\\&=\sum_{\mu=1}^{2n}d_{\mu-1}a_{\nu\mu}^2,\end{aligned}$$

by virtue of the orthogonality of the p_r with respect to $dl(t)$. Similarly,

$$\int_{\mathbb{R}}k_\nu^2(t)\,dl(t)=\sum_{\mu=1}^{2n}d_{\mu-1}b_{\nu\mu}^2.$$

Therefore,

$$\|\Lambda^{-1}P^{-1}D\|_F^2=\int_{\mathbb{R}}\sum_{\nu=1}^{n}\left(h_\nu^2(t)+\frac{1}{\lambda_\nu^2}k_\nu^2(t)\right)dl(t),$$

which in view of (3.17), (3.18) proves the theorem. □

We remark that the integral in (3.15), since the integrand is a polynomial of degree $\leqq 4n-2$, can be evaluated exactly (up to rounding errors) by the $2n$-point Gauss–Christoffel quadrature formula associated with $dl(t)$. This causes little problem, since dl is usually one of the standard integration measures and, besides, the integrand is positive. Furthermore, the integrand is conveniently evaluated in the form

$$(3.19)\quad \sum_{\nu=1}^{n}\left(h_\nu^2(t)+\frac{1}{\lambda_\nu^2}k_\nu^2(t)\right)=\frac{\sum_{\nu=1}^{n}[\rho_\nu(t-\tau_\nu)]^{-4}([1-2\sigma_\nu(t-\tau_\nu)]^2+\lambda_\nu^{-2}(t-\tau_\nu)^2)}{(\sum_{\nu=1}^{n}[\rho_\nu(t-\tau_\nu)]^{-1})^4},$$

where

$$\rho_\nu=\prod_{\substack{\mu=1\\ \mu\neq\nu}}^{n}(\tau_\nu-\tau_\mu),\qquad \sigma_\nu=\sum_{\substack{\mu=1\\ \mu\neq\nu}}^{n}\frac{1}{\tau_\nu-\tau_\mu},$$

as follows directly from (3.13) and the fact that $\sum_\nu l_\nu(t)=1$ and $l_\nu(t)=\rho_\nu^{-1}\prod_{\mu\neq\nu}(t-\tau_\mu)$ for the Lagrange polynomials. Of course, the evaluation of (3.19), as well as of $\|\gamma\|$ in (3.16), requires knowledge of the Gaussian abscissas and Christoffel numbers for $d\lambda(t)$.

We will have occasion to comment further on the result (3.15), when we discuss specific examples in § 4. Suffice it to say, here, that Gaussian abscissas τ_ν that are distributed approximately uniformly (as they tend to be for discrete orthogonal polynomials based on an equally spaced point spectrum) give rise to integrals in (3.15) that are likely to be very large for large n on account of the violent oscillations of h_ν and k_ν near the extreme nodes τ_μ. Abscissas τ_μ, on the other hand, that are distributed more like Chebyshev points are expected to yield much smaller values for these integrals, hence better condition for the map $\tilde{G}_n$.

The sharpness of (3.15) can be tested by considering $dl(t) = d\lambda(t)$, in which case the map $\tilde{G}_n$ essentially reduces to the (well-conditioned) map H_n^{-1}. The integral in (3.15), nevertheless, does not appear to allow an easy evaluation or estimation in simple form, except in special cases. One such special case is the Chebyshev measure $d\lambda(t) = (1-t^2)^{-1/2}\,dt$ on $[-1, 1]$, for which the integral in question can be evaluated by the Turán quadrature formula [28],

$$\int_{-1}^{1} f(t)(1-t^2)^{-1/2}\,dt = \sum_{\nu=1}^{n} [\lambda_\nu f(\tau_\nu) + \lambda'_\nu f'(\tau_\nu) + \lambda''_\nu f''(\tau_\nu)] + R_n(f),$$

where $\tau_\nu = \cos((2\nu-1)\pi/2n)$, which is exact for all $f \in \mathbb{P}_{4n-1}$ and known in closed form [22]:

$$\lambda_\nu = \frac{\pi}{n}, \quad \lambda'_\nu = -\frac{\pi}{4n^3}\tau_\nu, \quad \lambda''_\nu = \frac{\pi}{4n^3}(1-\tau_\nu^2).$$

One finds, in view of (3.14),

$$\int_{-1}^{1} \sum_{\mu=1}^{n} \left[h_\mu^2(t) + \frac{1}{\lambda_\mu^2} k_\mu^2(t)\right](1-t^2)^{-1/2}\,dt = \sum_{\nu=1}^{n} \left[\lambda_\nu + 2\lambda''_\nu\left(\frac{1}{\lambda_\nu^2} + h''_\nu(\tau_\nu)\right)\right]$$
$$\leqq \pi + 2\sum_{\nu=1}^{n} \frac{\lambda''_\nu}{\lambda_\nu^2} = \pi + \frac{2n^2}{\pi^2}\sum_{\nu=1}^{n} \lambda''_\nu,$$

since $h''_\nu(\tau_\nu) < 0$ for all ν. Observing that

$$\sum_{\nu=1}^{n} \lambda''_\nu = \frac{\pi}{4n^3}\sum_{\nu=1}^{n} \sin^2\left(\frac{2\nu-1}{2n}\pi\right) = \frac{\pi}{4n^3}\cdot\frac{n}{2} = \frac{\pi}{8n^2},$$

one gets

$$\int_{-1}^{1} \sum_{\mu=1}^{n} \left[h_\mu^2(t) + \frac{1}{\lambda_\mu^2} k_\mu^2(t)\right](1-t^2)^{-1/2}\,dt \leqq \pi + \frac{1}{4\pi}.$$

Finally, since $\nu_0 = \pi$, $\nu_r = 0$ for $r > 0$, hence $\tilde{\nu}_0 = \sqrt{\pi}$, $\tilde{\nu}_r = 0$ for $r > 0$, and since $\sum_{\nu=1}^{n} (\lambda_\nu^2 + \tau_\nu^2) = (\pi^2/n) + (n/2)$, we obtain

$$(\text{cond}\, \tilde{G}_n)(\tilde{\nu}) \leqq \sqrt{\frac{\pi^2 + 1/4}{\pi^2/n + n/2}} \sim \sqrt{\frac{2\pi^2 + 1/2}{n}} \quad \text{as } n \to \infty,$$

admittedly a somewhat too optimistic result (made so by the factor $1/\|\gamma\|$ in (3.15)).

The same considerations apply to the example $dl(t) = (1-t^2)^{-1/2}\,dt$ on $[-1, 1]$ and $d\lambda(t) = d\lambda_N(t)$ the discrete N-point measure with abscissas at the Chebyshev points $t_m = \cos((2m-1)\pi/2N)$, $m = 1, 2, \cdots, N$, and jumps equal to π/N, provided that $n \leqq N$.

3.4. The condition underlying the discretized Stieltjes procedure. It is not entirely clear what should be the appropriate map that underlies the discretized Stieltjes procedure. In the simplest case $d\lambda(t) = \omega(t)\,dt$ on $[-1, 1]$, the input data surely include the values $\omega(t_m^{(N)})$ of the weight function at the discretization points $t_m^{(N)}$, $m = 1, 2, \cdots, N$, but may also include these points themselves, as well as the quadrature weights $w_m^{(N)}$. From these data the procedure then determines the desired coefficients α_k, β_k, $k = 0, 1, \cdots, n-1$, or more precisely, their discrete approximations

$\alpha_{k,N}$, $\beta_{k,N}$, $k=0, 1, \cdots, n-1$. Analogous considerations apply to the more general measures $d\lambda(t)$ considered at the end of § 2.2.

The map in question, therefore, is similar to the map H_n considered in § 3.1, and in fact may be thought of as an approximation $H_{n,N}$ of H_n. Since our interest is in the condition of these maps, where orders of magnitude is all that matters, we may as well take the condition of H_n as indicative of the sensitivities inherent in the discretized Stieltjes procedure. It will be seen by numerical examples that cond H_n indeed agrees reasonably well with the actual error growth observed in the discretized Stieltjes procedure.

4. Examples. The purpose of this section is to illustrate the performance of the procedures of § 2, and the underlying theory of § 3, in a number of examples that we hope are representative. All computations reported were carried out on the CDC 6500 computer in single precision, except for the computation of errors, which was done in double precision.

4.1. Discrete orthogonal polynomials.

Example 4.1. The discrete orthogonal polynomials $t_r(x)$ of Chebyshev.

These are orthogonal with respect to the N-point discrete measure with abscissas at the integers $0, 1, \cdots, N-1$ and jumps equal to $1/N$:

$$\frac{1}{N}\sum_{k=0}^{N-1} t_r(k)t_s(k)=0, \qquad r\neq s, \quad r, s=0, 1, \cdots, N-1. \tag{4.1}$$

We prefer to deal with the (monic) polynomials

$$\pi_r(x)=\frac{r!^2}{(2r)!}N^{-r}t_r(Nx), \tag{4.2}$$

which satisfy the recurrence relation (2.1) with

$$\begin{aligned} &\alpha_k=\frac{1}{2}\left(1-\frac{1}{N}\right), \qquad k=0, 1, \cdots, N-1,\\ \beta_0=1, \quad &\beta_k=\frac{1-(k/N)^2}{4(4-1/k^2)}, \qquad k=1, 2, \cdots, N-1, \end{aligned} \tag{4.3}$$

and have their point spectrum on the interval [0, 1]. As $N\to\infty$, the polynomials (4.2) tend to the monic Legendre polynomials (shifted to the interval [0, 1]).

We first illustrate in Table 4.1 the ill-conditioning of the map G_n from the ordinary moments

$$\mu_r=\int_0^1 t^r\, d\lambda_N(t)=\frac{1}{N}\sum_{k=0}^{N-1}\left(\frac{k}{N}\right)^r, \qquad r=0, 1, \cdots, 2n-1,$$

TABLE 4.1
The condition of the map G_n in the case of ordinary moments and discrete Chebyshev measure $d\lambda(t)=d\lambda_N(t)$, $N=20$.

n	cond G_n (3.10)	cond G_n (3.11)	err. growth
2	1.957 (1)	4.110 (1)	2.456 (1)
5	6.109 (4)	5.047 (5)	2.110 (5)
8	5.318 (8)	1.768 (10)	1.028 (10)
11	4.366 (12)	1.406 (15)	3.286 (14)

with $N=20$, to the n-point Gauss–Christoffel formula, in the format that was already used in Table 3.2. As is evident from Table 4.1, Chebyshev's original algorithm rapidly loses accuracy, at the rate of somewhat more than one decimal digit per degree!

More stable, though not entirely unproblematic, is the modified Chebyshev algorithm, which we illustrate in Table 4.2 by recording the bound (3.15) for the condition of $\tilde{G}_n$, as well as the actual error growth observed. The latter is now defined as the L_2-norm of the relative errors in the coefficients $\alpha_k, \beta_k, k=0,1,\cdots,n-1$, divided by $\varepsilon\sqrt{2n}$, where ε is the machine precision. We feel that this is the appropriate measure, since the result (3.15) is based on the Euclidean norm. The modified moments chosen are those relative to the (monic) Legendre polynomials for the interval [0, 1].

TABLE 4.2
The condition of the map $\tilde{G}_n$ in the case of Legendre moments and discrete Chebyshev measure $d\lambda(t)=d\lambda_N(t)$, $N=10, 20, 40, 80$.

N	n	cond $\tilde{G}_n$	err. growth	N	n	cond $\tilde{G}_n$	err. growth
10	5	2.515 (0)	4.713 (0)	40	15	2.020 (1)	1.679 (2)
	10	6.311 (4)	7.349 (4)		25	1.311 (6)	1.016 (7)
20	5	7.859 (−1)	3.537 (0)		35	5.015 (14)	1.110 (15)
	10	1.932 (1)	1.105 (2)	80	10	4.885 (−1)	3.126 (0)
	15	2.952 (4)	1.421 (5)		20	6.480 (0)	1.240 (2)
	20	3.328 (10)	9.646 (10)		30	3.936 (3)	8.320 (4)
40	5	6.463 (−1)	2.106 (0)		40	4.800 (7)	1.013 (9)
	10	9.953 (−1)	7.182 (0)		50	1.738 (13)	1.759 (16)

The magnitude of cond $\tilde{G}_n$ is solely determined by the integral in (3.15), since $\|\tilde{\nu}\|$ and $\|\gamma\|$ in this example both have order of magnitude 1. The steady growth of cond $\tilde{G}_n$ can be explained by the fact that as n approaches N, the Gaussian nodes of $d\lambda_N(t)$ become more and more equally distributed. (They are equally spaced when $n=N$.) The Hermite interpolation polynomials h_ν and k_ν in (3.13) therefore exhibit the violent oscillations characteristic of equally spaced nodes, which accounts for the large values of the integral in (3.15). Chebyshev nodes on [0, 1], according to this explanation, ought to result in substantially smaller conditions, a fact that will indeed be confirmed in the next example.

The error growth shown in Table 4.2 is consistently somewhat larger than what is indicated by cond $\tilde{G}_n$. This is because the growth of error in the coefficients α_k, β_k includes also the effects of the map H_n, the condition of which is shown in Table 4.3.

One might think that the large oscillations of h_ν and k_ν could be filtered out by choosing a measure $dl(t)$ in (3.15) which is very small (or even equal to zero) near the end zones of the interval [0, 1]. While this indeed reduces the magnitude of the bothersome integral, the other factor $\|\tilde{\nu}\|$ in (3.15) increases so much more that the condition of $\tilde{G}_n$ in fact gets worse.

Substantially more stable is the Stieltjes procedure, measured both in terms of the condition of the map H_n (cf. § 3.4) and in terms of actual performance. For $N=10$ and 20, cond H_n is less than 22.08 and 50.80, respectively, for all $n \leqq N$, whereas the actual error growth observed is by factors of at most 10.86 and 16.66, respectively. For $N=40$ and $N=80$ we have the situation indicated in Table 4.3. It shows that ill-conditioning and consequent instability set in as n approaches N, relatively late for $N=40$, but sooner for $N=80$. The condition of H_n is seen to correctly predict the trend of instability, but overestimates it by several orders of magnitude.

TABLE 4.3
Stability of the Stieltjes procedure for the discrete Chebyshev measure $d\lambda(t) = d\lambda_N(t)$, $N = 40$ *and* 80.

N	n	cond H_n	err. growth	N	n	cond H_n	err. growth
40	$\leqq 35$	$\leqq 7.2$ (3)	$\leqq 2.4$ (1)	80	$\leqq 50$	$\leqq 2.6$ (3)	$\leqq 1.95$ (1)
	36	1.030 (5)	2.122 (2)		55	4.751 (6)	7.015 (3)
	37	1.755 (6)	3.835 (3)		60	2.937 (10)	4.284 (7)
	38	4.419 (7)	9.216 (4)		65	2.220 (12)	1.088 (12)
	39	1.641 (9)	3.361 (6)		70	2.172 (12)	1.668 (15)
	40	1.205 (11)	2.487 (8)		75	2.006 (12)	1.668 (15)

We report these results solely to illustrate the behavior of the various procedures in a typical case of a discrete measure involving equally spaced points. There is, of course, no need to apply these procedures, since the recurrence relation is known explicitly (cf. (4.3)).

Example 4.2. Polynomials orthogonal with respect to the discrete inner product

$$[p, q]_N = \sum_{k=1}^{N} w_k p(t_k) q(t_k),$$

where $t_k = t_k^{(N)}$ are the Chebyshev points on $[-1, 1]$ and $w_k = w_k^{(N)}$ the weights of the N-point Fejér quadrature rule. This example is of interest in connection with our discretization of the Stieltjes procedure (cf. § 2.2).

It seems natural, in this case, to run the modified Chebyshev algorithm with the modified moments relative to the (monic) Chebyshev polynomials of the first kind. The map $\tilde{G}_n$ then turns out to be perfectly well-conditioned; see Table 4.4. For $N = 10$, 20, 40, 80 and for selected values $n \leqq N$, we found cond $\tilde{G}_n$ never to exceed 1.2, and to be usually less than 1. The map H_n, likewise, appears to be quite well-conditioned. Accordingly, both the modified Chebyshev algorithm, as well as the Stieltjes procedure, perform exceedingly well. The respective error growths are shown in the last two columns of Table 4.4.

TABLE 4.4
Performance of the modified Chebyshev algorithm and the Stieltjes procedure in Example 4.2.

N	n	cond $\tilde{G}_n$	cond H_n	err. growth in Chebyshev algorithm	err. growth in Stieltjes procedure
10	5	1.169 (0)	6.968 (0)	1.969 (0)	3.750 (0)
	10	8.925 (−1)	2.133 (1)	1.969 (0)	1.472 (1)
20	5	1.169 (0)	6.968 (0)	1.969 (0)	6.000 (0)
	10	9.152 (−1)	1.785 (1)	1.994 (0)	9.969 (0)
	20	6.473 (−1)	5.054 (1)	1.045 (1)	2.053 (1)
40	10	9.152 (−1)	1.785 (1)	1.994 (0)	1.200 (1)
	20	6.684 (−1)	4.530 (1)	5.996 (0)	1.200 (1)
	40	4.597 (−1)	1.213 (2)	3.146 (1)	2.697 (1)
80	20	6.684 (−1)	4.530 (1)	5.996 (0)	2.300 (1)
	40	4.773 (−1)	1.071 (2)	9.998 (0)	2.300 (1)
	80	3.250 (−1)	2.827 (2)	7.948 (1)	8.435 (1)

Example 4.3. "Truncated Charlier polynomials", orthogonal with respect to the inner product

$$[p, q]_N = \sum_{k=0}^{N-1} \frac{e^{-a}a^k}{k!} p(k)q(k), \qquad a > 0.$$

For $N \to \infty$, these become the Charlier polynomials, whose recurrence formula is known explicitly.

The modified Chebyshev algorithm, at least when used in conjunction with modified moments based on Laguerre polynomials, performs rather poorly on this example. The main reason is the rapidly deteriorating condition of the respective map $\tilde{G}_n$. This is illustrated in Table 4.5 for the case $a = 1$ and $N = 40$. Practically identical results are obtained for larger N, and quite similar ones for smaller values of N.

TABLE 4.5
Performance of the modified Chebyshev algorithm with Laguerre moments in Example 4.3.

N	n	cond $\tilde{G}_n$	err. growth
40	2	4.113 (0)	0.0
	4	1.832 (3)	1.365 (2)
	6	3.963 (6)	5.622 (4)
	8	2.006 (10)	2.217 (9)
	10	1.793 (14)	2.907 (13)

For comparison we give in Table 4.6 some analogous information for the Stieltjes procedure.

TABLE 4.6
Performance of the Stieltjes procedure in Example 4.3.

N	n	cond H_n	err. growth
40	5	8.130 (0)	5.995 (0)
	10	2.740 (1)	1.027 (1)
	15	4.635 (1)	2.241 (1)
	20	6.661 (1)	3.547 (1)
	25	7.215 (5)	8.444 (7)
	30	1.532 (11)	7.290 (14)

4.2. Polynomials orthogonal on an interval.

Example 4.4. An example of Christoffel [4, Ex. 6]: $d\lambda(t) = \omega(t)\,dt$ with $\omega(t) = [(1-k^2t^2)(1-t^2)]^{-1/2}$ on $[-1, 1]$, $0 < k < 1$.

What intrigued Christoffel was the fact that the associated orthogonal polynomials $\{\pi_r(t)\}$, when considered as functions of $x = \int_0^t \omega(t)\,dt$, constitute a sequence of doubly periodic functions orthogonal in the sense

$$\int_{-K}^{K} \pi_r(t)\pi_s(t)\,dx = 0, \qquad r \neq s,$$

where K denotes the complete elliptic integral $K = \int_0^1 \omega(t)\,dt$.

Since $(1-k^2t^2)^{-1/2}$ is analytic in a neighborhood of the segment $[-1, 1]$, the desired polynomials must be "close" to the Chebyshev polynomials of the first kind.

This suggests the use of the latter as input to the modified Chebyshev algorithm, i.e., the construction of the desired recursion coefficients from the modified moments

$$\nu_r = \int_{-1}^{1} p_r(t)\omega(t)\,dt \tag{4.4}$$

with respect to the monic Chebyshev polynomials $p_0 = T_0, p_r(t) = T_r(t)/2^{r-1}, r = 1, 2, \cdots$. These moments can be computed as follows. Letting first $t = \cos\varphi$ in (4.4) gives

$$\nu_0 = \int_0^\pi \frac{d\varphi}{(1-k^2\cos^2\varphi)^{1/2}}, \qquad \nu_r = \frac{1}{2^{r-1}}\int_0^\pi \frac{\cos r\varphi}{(1-k^2\cos^2\varphi)^{1/2}}\,d\varphi, \qquad r \geqq 1. \tag{4.5}$$

Now put $\theta = \pi/2 - \varphi$ in the Fourier expansion

$$\frac{1}{(1-k^2\sin^2\theta)^{1/2}} = C_0(k^2) + 2\sum_{n=1}^{\infty} C_n(k^2)\cos 2n\theta$$

and substitute the result in (4.5). By the orthogonality of the cosine functions one immediately obtains

$$\begin{aligned} \nu_0 &= \pi C_0(k^2), \\ \nu_{2m} &= (-1)^m \frac{\pi}{2^{2m-1}} C_m(k^2), \qquad m = 1, 2, 3, \cdots, \end{aligned} \tag{4.6}$$

while of course $\nu_{2m-1} = 0, m = 1, 2, 3, \cdots$. On the other hand, $y_n = C_n(k^2), n = 0, 1, 2, \cdots$, is a minimal solution of the three-term recurrence relation

$$\left(n+\frac{1}{2}\right)y_{n+1} + n\frac{1+q^2}{q}y_n + \left(n-\frac{1}{2}\right)y_{n-1} = 0, \qquad n = 1, 2, 3, \cdots, \tag{4.7}$$

satisfying

$$y_0 + 2\sum_{n=1}^{\infty} y_n = 1, \tag{4.8}$$

where

$$q = \frac{k^2}{2-k^2+2(1-k^2)^{1/2}}$$

(see, e.g., Luke [23, p. 36]). Our algorithm in [9, Eq. (3.9)], in conjunction with the normalizing condition (4.8), then yields the Fourier coefficients $C_n(k^2)$, hence the modified moments (4.6), very accurately and efficiently. The algorithm works well even when k^2 is quite close to 1. Note, in fact, that (4.7) is a difference equation of the Poincaré type, with characteristic equation

$$u^2 + \frac{1+q^2}{q}u + 1 = 0, \qquad 0 < q < 1,$$

having two real roots u_1, u_2 with $|u_1| > 1 > |u_2|$ and

$$\left|\frac{u_1}{u_2}\right| = \frac{1}{q^2}.$$

(The minimal solution $y_n = C_n(k^2)$ "corresponds" to u_2.) If $k^2 = 1-\varepsilon$, $0<\varepsilon \ll 1$, then

$$\frac{1}{q^2} = \left(\frac{1+\sqrt{\varepsilon}}{1-\sqrt{\varepsilon}}\right)^2 = 1+4\sqrt{\varepsilon}+8\varepsilon+12\varepsilon\sqrt{\varepsilon}+16\varepsilon^2+o(\varepsilon^2), \qquad \varepsilon \to 0,$$

so that for $k^2 = .999$, for example, we have $\varepsilon = 10^{-3}$, hence

$$\left|\frac{u_1}{u_2}\right| \approx 1.13,$$

which is still an adequate separation of the roots.

In addition to the modified moments being accurately computable, it turns out that the modified Chebyshev algorithm is extremely stable. For all values of k^2 that we tried ($0<k^2 \leqq .999$), and for degrees n up to 80, the error growth factor never exceeded 3.258, and the condition number cond $\tilde{G}_n$ never 3.153.

We have also used the discretized Stieltjes procedure, as well as the discretized modified Chebyshev algorithm (cf. §§ 2.2 and 2.5), with good success, using the Gauss–Chebyshev quadrature rule in place of Fejér's. The advantage of testing convergence on the relative accuracy of the coefficients $\beta_{k,N}$, rather than on that of the modified moments $\nu_{r,N}$ (cf. § 2.5), can be clearly demonstrated in this example. The modified moments indeed decrease very rapidly (unless k^2 is close to 1), so that insistence on high relative accuracy in these moments would not be meaningful. For example, if $k^2 = .5$, $n = 20$, we find that

$$\max_{0\leqq k\leqq n-1} \left|\frac{\beta_{k,N}-\beta_k}{\beta_k}\right| = 8.81\times 10^{-14} \quad \text{for } N = 60,$$

while for the same value of N,

$$\max_{\substack{0\leqq r\leqq 2n-1\\ r \text{ even}}} \left|\frac{\nu_{r,N}-\nu_r}{\nu_r}\right| = 1.76\times 10^{1},$$

the maximum being attained for $r = 36$, where $\nu_r = 2.3825\cdots\times 10^{-25}$.

Example 4.5. Logarithmic singularity: $d\lambda(t) = \ln(1/t)\,dt$ on $[0, 1]$.

The modified moments relative to (shifted) Legendre and Jacobi polynomials are known explicitly for this measure, and even for more general measures such as $d\lambda(t) = t^\alpha(1-t)^\beta \ln(1/t)\,dt$, $\alpha, \beta > -1$ (cf. [1], [7], [17], [21]). The modified Chebyshev algorithm, based on Legendre moments, produces results which are essentially accurate to machine precision; the largest error growth factor observed in the range $1\leqq n\leqq 80$ is 2.82. The reason for this excellent performance is to be found in the well-conditioning of the maps $\tilde{G}_n$ and H_n, for which we show in Table 4.7 the bound (3.15) for cond $\tilde{G}_n$ and cond H_n computed on the basis of (3.8).

TABLE 4.7
The condition of the maps $\tilde{G}_n$ and H_n in Example 4.5.

n	cond $\tilde{G}_n$	cond H_n
5	5.903 (0)	7.835 (0)
10	1.090 (1)	2.040 (1)
20	2.058 (1)	4.623 (1)
40	3.981 (1)	1.095 (2)
80	7.818 (1)	2.548 (2)

The discretized Stieltjes procedure, in contrast, converges rather slowly, making it difficult to obtain an accuracy much higher than 6 or 7 significant decimal digits.

Example 4.6. Half-range Hermite measure $d\lambda\ (t) = e^{-t^2}\,dt$ on $[0, \infty]$.

Here, the map H_n is quite well-conditioned (cond $H_n \leqq 2.28 \times 10^2$ for $n \leqq 80$), in contrast to the map $\tilde{G}_n$, which becomes rapidly ill-conditioned if modified moments relative to Hermite or Laguerre polynomials are used, which appear to be natural choices. Interestingly, Laguerre polynomials give significantly worse conditionings than Hermite polynomials, which is also borne out by a correspondingly faster error growth in the coefficients α_k, β_k; see Table 4.8. Accordingly, the modified Chebyshev algorithm is not effective in this example. Acceptable results, with some effort, can be had by the discretized Stieltjes procedure, which for $n = 40$, e.g., produces α_k, β_k, $k = 0, 1, \cdots, n-1$, to about 12 correct decimal digits, requiring a discretization parameter $N = 560$. Much better results are obtained if the interval $[0, \infty]$ is decomposed as $[0, 3] \cup [3, 6] \cup [6, 9] \cup [9, \infty]$ and the discretized Stieltjes procedure is applied in the manner described at the end of § 2.2, using Fejér's quadrature rule (suitably transformed) in each subinterval. Again for $n = 40$, this will yield 15 correct decimal digits with $N = 80$. The method is similarly applicable to more general measures $d\lambda(t) = e^{-t^p}\,dt$ on $[0, \infty]$, $p > 1$.

TABLE 4.8
The condition of $\tilde{G}_n$ in Example 4.6 for modified moments based on Hermite and Laguerre polynomials.

n	Hermite moments		Laguerre moments	
	cond $\tilde{G}_n$	err. growth	cond $\tilde{G}_n$	err. growth
2	1.554 (1)	5.713 (0)	7.270 (1)	2.296 (1)
4	2.524 (3)	3.261 (2)	1.349 (6)	1.045 (6)
6	6.739 (5)	6.300 (4)	1.297 (11)	9.663 (10)
8	2.206 (8)	2.386 (8)	3.127 (16)	3.112 (16)
10	8.026 (10)	2.696 (11)	5.547 (21)	–

4.3. Polynomials orthogonal with respect to multiple component distributions. As already observed in § 2.2, the discretized Stieltjes procedure can also handle measures $d\lambda(t)$ of a more general type, for example, measures on a set of disjoint intervals or measures including a point spectrum. The discretized Stieltjes procedure in these circumstances is often far superior to the modified Chebyshev algorithm, which tends to become unstable. We illustrate this by a number of examples, of which Example 4.11 may prove useful in the numerical solution of large systems of linear algebraic equations by iterative methods [31].

Example 4.7. Piecewise constant weight function: $d\lambda(t) = \omega(t)\,dt$, where $\omega(t) = 1$ on $[-1, -\xi] \cup [\xi, 1]$ and $\omega(t) = 0$ elsewhere, $0 < \xi < 1$.

Equivalently, $d\lambda(t) = [\omega_1(t) + \omega_2(t)]\,dt$, where ω_1, ω_2 are the characteristic functions of the intervals $[-1, -\xi]$ and $[\xi, 1]$, respectively. The discretized Stieltjes procedure, as amended at the end of § 2.2, works extremely well, even for ξ relatively close to 1; the map H_n remains well-conditioned. Some relevant data are given in Table 4.9. (The discretized Stieltjes procedure in this example converges after one iteration, if the discretization parameter N is chosen appropriately; cf. § 2.2).

The modified moments ν_r, $r = 0, 1, \cdots, 2n-1$, based on Legendre polynomials are easily computed (exactly) by n-point Gauss–Legendre quadrature. The ensuing

TABLE 4.9
Performance of the discretized Stieltjes procedure in Example 4.7.

ξ	n	cond H_n	err. growth	ξ	n	cond H_n	err. growth
.3	5	7.683 (0)	8.233 (0)	.7	5	1.974 (1)	1.392 (1)
	10	2.138 (1)	9.956 (0)		10	5.520 (1)	1.392 (1)
	20	5.230 (1)	1.522 (1)		20	1.295 (2)	1.800 (1)
	40	1.248 (2)	1.937 (1)		40	3.072 (2)	1.803 (1)
.5	5	1.060 (1)	1.390 (1)	.9	5	6.585 (1)	2.463 (1)
	10	2.979 (1)	1.426 (1)		10	1.919 (2)	2.463 (1)
	20	7.159 (1)	1.426 (1)		20	4.469 (2)	3.894 (1)
	40	1.709 (2)	1.598 (1)		40	1.073 (3)	5.773 (1)

modified Chebyshev algorithm, however, becomes severely unstable, even for moderately large ξ, on account of ill-conditioned maps $\tilde{G}_n$. This is documented in Table 4.10. The same is true for the discretized modified Chebyshev algorithm.

TABLE 4.10
Performance of the modified Chebyshev algorithm in Example 4.7.

ξ	n	cond $\tilde{G}_n$	err. growth	ξ	n	cond $\tilde{G}_n$	err. growth
.3	5	1.335 (0)	2.118 (2)	.7	5	6.454 (1)	7.081 (2)
	10	7.754 (0)	1.955 (2)		10	6.457 (4)	4.198 (5)
	20	1.276 (3)	6.537 (3)		20	8.012 (11)	8.015 (12)
	40	1.169 (8)	7.077 (8)		40	3.748 (26)	9.648 (15)
.5	5	4.830 (0)	1.914 (2)	.9	5	9.263 (3)	1.244 (4)
	10	3.658 (2)	1.640 (3)		10	1.171 (9)	1.838 (9)
	20	7.650 (6)	4.445 (7)		20	2.630 (21)	8.940 (16)
	40	1.057 (16)	3.683 (14)		40	-	-

By virtue of symmetry, the orthogonal polynomials $\{\pi_r\}$ of Example 4.7 can be expressed in terms of polynomials orthogonal on a single interval. Indeed, letting $\pi_{2r}(t)=p_r^+(t^2)$, $\pi_{2r+1}(t)=tp_r^-(t^2)$, $r=0,1,2,\cdots$, the polynomials $p_r^\pm(x)$ are orthogonal on $[\xi^2,1]$ with respect to the weight function $\omega^\pm(t)=t^{\mp 1/2}$. If α_k^+,β_k^+ are the recursion coefficients for $\{p_r^+(x)\}$, then

$$\beta_0=2(1-\xi), \qquad \beta_1=\alpha_0^+,$$

$$\left.\begin{aligned}\beta_{2k}&=\frac{\beta_k^+}{\beta_{2k-1}},\\ \beta_{2k+1}&=\alpha_k^+-\beta_{2k},\end{aligned}\right\} \quad k=1,2,3,\cdots$$

are those for the desired polynomials $\{\pi_r(t)\}$ (cf. [3, Chapt. I, §§ 8–9]). The discretized Stieltjes procedure could also be used to generate α_k^+,β_k^+, but would then require an infinite process, rather than the finite one when applied directly to the weight function ω.

Example 4.8. Adding a point spectrum to the distribution $d\lambda(t)$ of Example 4.7, where $\xi=.5$.

We make the distribution asymmetric if we add a point spectrum consisting of a single point, say at $t_1=2$, with jump $w_1=1$. The effect of this is a slight worsening of the condition of $\tilde{G}_n$ and a profound impairment of cond H_n. As a result, one now has difficulty not only with the modified Chebyshev algorithm, but also with the discretized Stieltjes procedure, although the latter "survives" a bit longer; see Table 4.11.

TABLE 4.11
Performance of the modified Chebyshev algorithm and the discretized Stieltjes procedure in Example 4.8.

			Error growth	
n	cond $\tilde{G}_n$	cond H_n	mod. Chebyshev	discr. Stieltjes
4	1.612 (1)	2.021 (1)	6.621 (1)	2.898 (2)
8	1.174 (3)	4.088 (1)	1.146 (7)	4.627 (2)
12	9.217 (4)	6.658 (1)	4.712 (11)	5.700 (2)
16	6.950 (6)	4.218 (4)	4.686 (17)	1.876 (4)
20	5.120 (8)	4.334 (9)	-	1.910 (9)
24	2.135 (10)	2.833 (14)	-	4.456 (14)

Adding another point, $t_2=-2$, with jump $w_2=1$, restores symmetry, but neither significantly improves, nor worsens, the condition of H_n.

We know of no stable method to compute orthogonal polynomials of the type introduced in Example 4.8.

Example 4.9.[1] Adding a constant to the Chebyshev weight function: $d\lambda(t)=[(1-t^2)^{-1/2}+a]\,dt$ on $[-1,1]$, $a>0$.

The discretized Stieltjes procedure applied directly to $d\lambda(t)=\omega(t)\,dt$, $\omega(t)=(1-t^2)^{-1/2}+a$ converges extremely slowly, regardless of whether the discretization is effected by Fejér's or the Gauss–Chebyshev quadrature rule. The reason for this is easily seen if one writes $\omega(t)=(1-t^2)^{-1/2}[1+a(1-t^2)^{1/2}]$ and notes that the function in brackets has infinite derivatives at $t=\pm1$. On the other hand, treating the two additive components of ω independently, as suggested at the end of § 2.2, and applying the Gauss–Chebyshev quadrature rule to the first, and Fejér's to the second, Stieltjes's procedure converges trivially. Results obtained for selected values of a in the range $0\leqq a\leqq 1000$, and $0\leqq n\leqq 80$, are accurate almost to machine precision, the largest error growth factor being 9.219(1). The condition numbers cond H_n are slowly decreasing as a function of a, from the values for Chebyshev polynomials, when $a=0$, to those for Legendre polynomials, when $a\to\infty$ (cf. Table 3.1).

Equally accurate, but considerably faster (by a factor of more than 10 for $n=80$) is the modified Chebyshev algorithm, based on Chebyshev moments

$$\nu_0=\pi+2a,$$
$$\nu_r=\frac{1}{2^{r-1}}\int_{-1}^{1}T_r(t)\,d\lambda(t)=-\frac{a}{2^{r-2}(r^2-1)},\qquad r \text{ even}\neq 0,$$
$$\nu_r=0,\qquad r \text{ odd}.$$

The maximum bound (3.15) for the condition of $\tilde{G}_n$ is found to be 6.748 (for $a=1000$, $n=80$), and the maximum error growth factor 1.146(1).

[1] This example was proposed to the author by Professor M. Golomb.

The recursion coefficients β_k, $k>0$, behave as expected, when a varies from 0 to ∞: There is a smooth (but not monotone) transition from the Chebyshev case to the Legendre case.

Example 4.10. A weight distribution involving a modified Bessel function: $d\lambda(t)=t^{\mu}K_0(t)\,dt$ on $[0,\infty]$, $\mu>-1$.

Gauss–Christoffel quadrature rules with this weight distribution are proposed by Wong [30] to obtain asymptotic approximations to oscillatory integrals.

It is known that

$$K_0(t)=\begin{cases} R(t)+I_0(t)\ln\left(\dfrac{1}{t}\right), & 0<t\leqq 1,\\ t^{-1/2}e^{-t}S(t), & 1\leqq t\leqq\infty,\end{cases}$$

where R and S are well-behaved smooth functions on their respective intervals and I_0 is the "regular" modified Bessel function. For R, S and I_0, high-accuracy rational approximations are available; see Russon and Blair [24]. Using the "multiple component" version of the discretized Stieltjes procedure, as described at the end of § 2.2, we decompose the inner product $(p,q)=\int_0^\infty t^{\mu}K_0(t)p(t)q(t)\,dt$ as follows,

$$(p,q)=\int_0^1 t^{\mu}[R(t)p(t)q(t)]\,dt+\int_0^1 t^{\mu}\ln\left(\frac{1}{t}\right)[I_0(t)p(t)q(t)]\,dt$$
$$+e^{-1}\int_0^\infty e^{-t}[(1+t)^{\mu-1/2}S(1+t)p(1+t)q(1+t)]\,dt,$$

and discretize the first integral by an N-point Gauss–Jacobi quadrature rule with parameters $\alpha=0$, $\beta=\mu$, the second by an N-point Gauss–Christoffel quadrature rule relative to the weight distribution $t^{\mu}\ln(1/t)\,dt$ on $[0,1]$, and the last one by an N-point Gauss–Laguerre quadrature rule. The first and last of these quadrature rules are easily obtained from the respective Jacobi matrices (see § 1, Eq. (1.4), and the remarks following this equation), while the second can be generated by the modified Chebyshev algorithm, as indicated in Example 4.5.

In this way, the desired orthogonal polynomials (and Gauss–Christoffel quadrature rules) can be generated accurately and in a stable manner. If $\mu=0$, or $\mu=-1/2$, for example, one gets the recursion coefficients α_k, β_k, $0\leqq k\leqq n$, accurately to 15 significant decimal digits by taking $N=100$ for $n=20$, and $N=160$ for $n=40$. In contrast, the discretized Stieltjes procedure based on the (transformed) Fejér quadrature rule requires $N=230$ for $\mu=0$ and $n=10$, just to get six correct decimal digits, and becomes prohibitively expensive for much larger values of n or higher accuracy.

Example 4.11. Find a polynomial $P_n(t)$ of degree $\leqq n$, with $P_n(1)=1$, such that $\int_0^1 P_n^2(t)\omega(t)\,dt=\min$, where $\omega(t)=\varepsilon$ on $[0,\xi]$, $\omega(t)=1$ on $[\xi,\eta]$, $\omega(t)=0$ on $[\eta,1]$, and $\varepsilon>0$, $0<\xi<\eta<1$.

The solution is known to be the polynomial orthogonal on $[0,1]$ with respect to the weight function $(1-t)\omega(t)$ (cf. [3, Chapt. I, § 7]). If $\pi_n(\cdot)=\pi_n(\cdot\,;(1-t)\omega(t)\,dt)$ denotes the monic orthogonal polynomial, then $P_n(t)=\pi_n(t)/\pi_n(1)$, and the desired minimum value is $\beta_0\beta_1\cdots\beta_n/\pi_n^2(1)$. The recursion coefficients α_k, β_k for $\{\pi_r\}$ are obtained in a stable manner by the discretized Stieltjes procedure, which can be made to converge after one iteration. Selected results (for the minimum value of $\int_0^1 P_n^2(t)\omega(t)\,dt$) in the case $\xi=1/3$, $\eta=2/3$, are shown in Table 4.12.

TABLE 4.12
Minimum values for the extremum problem of Example 4.11.

ε	n	$\min \int_0^1 P_n^2\omega\, dt$	ε	n	$\min \int_0^1 P_n^2\omega\, dt$
.0	5	4.890 (−9)	.6	5	7.984 (−7)
	10	1.107 (−16)		10	1.551 (−12)
	20	5.479 (−32)		20	5.733 (−24)
	40	1.317 (−62)		40	7.653 (−47)
.2	5	5.382 (−7)	.8	5	8.758 (−7)
	10	1.107 (−12)		10	1.707 (−12)
	20	4.038 (−24)		20	6.308 (−24)
	40	5.347 (−47)		40	8.420 (−47)
.4	5	6.950 (−7)	1.0	5	9.386 (−7)
	10	1.364 (−12)		10	1.842 (−12)
	20	5.030 (−24)		20	6.802 (−24)
	40	6.703 (−47)		40	9.075 (−47)

REFERENCES

[1] J. L. BLUE, *A Legendre polynomial integral*, Math. Comp., 33 (1979), pp. 739–741.

[2] P. L. CHEBYSHEV, *Sur l'interpolation par la méthode des moindres carrés*, Mém. Acad. Impér. Sci. St. Pétersbourg, (7) 1, no. 15 (1859), pp. 1–24. [Oeuvres I, pp. 473–498.]

[3] T. S. CHIHARA, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[4] E. B. CHRISTOFFEL, *Sur une classe particulière de fonctions entières et de fractions continues*, Ann. Mat. Pura Appl., (2) 8 (1877), pp. 1–10. [Ges. Math. Abhandlungen II, pp. 42–50.]

[5] G. DARBOUX, *Mémoire sur l'approximation des fonctions de très-grands nombres et sur une classe étendue de développements en série*, J. Math. Pures Appl., (3) 4 (1878), pp. 5–56; 377–416.

[6] G. E. FORSYTHE, *Generation and use of orthogonal polynomials for data-fitting with a digital computer*, J. Soc. Indust. Appl. Math., 5 (1957), pp. 74–88.

[7] L. GATTESCHI, *On some orthogonal polynomial integrals*, Math. Comp., 35 (1980), pp. 1291–1298.

[8] W. GAUTSCHI, *On inverses of Vandermonde and confluent Vandermonde matrices II*, Numer. Math., 5 (1963), pp. 425–430.

[9] ———, *Computational aspects of three-term recurrence relations*, SIAM Rev., 9 (1967), pp. 24–82.

[10] ———, *Numerical quadrature in the presence of a singularity*, SIAM J. Numer. Anal., 4 (1967), pp. 357–362.

[11] ———, *Construction of Gauss–Christoffel quadrature formulas*, Math. Comp., 22 (1968), pp. 251–270.

[12] ———, *Algorithm 331—Gaussian quadrature formulas*, Comm. ACM, 11 (1968), pp. 432–436.

[13] ———, *On the construction of Gaussian quadrature rules from modified moments*, Math. Comp., 24 (1970), pp. 245–260.

[14] ———, *Computational methods in special functions—A survey*, in Theory and Application of Special Functions, R. A. Askey, ed., Academic Press, New York, 1975, pp. 1–98.

[15] ———, *Questions of numerical condition related to polynomials*, in Recent Advances in Numerical Analysis, C. de Boor and G. H. Golub, eds., Academic Press, New York, 1978, pp. 45–72.

[16] ———, *On generating Gaussian quadrature rules*, in Numerische Integration, G. Hämmerlin, ed., ISNM vol. 45, Birkhäuser, Basel, 1979, pp. 147–154.

[17] ———, *On the preceding paper "A Legendre polynomial integral" by James L. Blue*, Math. Comp., 33 (1979), pp. 742–743.

[18] ———, *Minimal solutions of three-term recurrence relations and orthogonal polynomials*, Math. Comp., 36 (1981), pp. 547–554.

[19] ———, *A survey of Gauss–Christoffel quadrature formulae*, in E. B. Christoffel: The Influence of his Work in Mathematics and the Physical Sciences; International Christoffel Symposium; A Collection of Articles in Honour of Christoffel on the 150th Anniversary of his Birth, P. L. Butzer and F. Fehér, eds., Birkhäuser, Basel, 1981, pp. 72–147.

[20] G. H. GOLUB AND J. H. WELSCH, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.

[21] S. L. KALLA, S. CONDE AND Y. L. LUKE, *Integrals of Jacobi functions*, Math. Comp., 38 (1982), pp. 207–214.

[22] O. KIS, *Remark on mechanical quadrature* (Russian), Acta Math. Acad. Sci. Hungar., 8 (1957), pp. 473–476.

[23] Y. L. LUKE, *The Special Functions and their Approximations*, vol. II, Academic Press, New York, 1969.

[24] A. E. RUSSON AND J. M. BLAIR, *Rational function minimax approximations for the Bessel functions $K_0(x)$ and $K_1(x)$*, Rep. AECL-3461, Atomic Energy of Canada Limited, Chalk River, Ontario, October 1969.

[25] R. A. SACK AND A. F. DONOVAN, *An algorithm for Gaussian quadrature given modified moments*, Numer. Math., 18 (1971/72), pp. 465–478.

[26] T. J. STIELTJES, *Quelques recherches sur la théorie des quadratures dites mécaniques*, Ann. Sci. École Norm. Paris Sér. 3, 1 (1884), pp. 409–426. [Oeuvres I, pp. 377–396.]

[27] G. SZEGÖ, *Orthogonal Polynomials*, 4th ed., AMS Colloquium Publications 23, American Mathematical Society, Providence, RI, 1975.

[28] P. TURÁN, *On the theory of mechanical quadrature*, Acta Sci. Math. (Szeged), 12 (1950), pp. 30–37.

[29] J. C. WHEELER, *Modified moments and Gaussian quadrature*, Rocky Mountain J. Math., 4 (1974), pp. 287–296.

[30] R. WONG, *Quadrature formulas for oscillatory integral transforms*, submitted for publication.

[31] D. M. YOUNG, personal communication, July 1981.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

RADIATION CONDITIONS FOR WAVE GUIDE PROBLEMS*

GREGORY A. KRIEGSMANN†

Abstract. An incident mode propagates down a two-dimensional wave guide until it strikes a localized obstruction which creates reflected and transmitted waves. The numerical determination of these waves is difficult because the classical radiation condition does not apply for an infinite wave guide. In this note we derive a sequence of "localized radiation conditions" which can be applied a few wavelengths away from the scattering object. These conditions allow us to numerically solve the Helmholtz equation on a finite domain.

Key words. wave guide, radiation conditions, Helmholtz equation, limiting amplitude principle, finite difference scheme, numerical methods

1. Introduction and formulation. In this paper we describe a numerical method for studying the interaction of an incident mode with a compact obstacle placed in a parallel plate wave guide. The technique is essentially the numerical implementation of an appropriate limiting amplitude principle [1]. This method has been recently exploited to study the numerical solutions of the Helmholtz equation on exterior domains [2]. The difference between the problem considered here and the scattering problem is the geometry: the wave guide is infinite only in the z direction. Accordingly, a different radiation condition is needed as $|z|\to\infty$. We have generated a sequence of boundary conditions which are analogous to those used for the exterior problem [2], [3]. These "local" conditions are applied at $\pm z_\infty$ and are responsible for numerical errors which tend rapidly to zero as $z_\infty \gg 1$. Our boundary conditions are distinctly different from the "global" radiation conditions used on underwater acoustical problems [4]. The later become increasingly more accurate as the wavelength approaches zero.

The determination of reflection and transmission coefficients for a wave guide is often accomplished by using an appropriate variational technique. A nice account of these methods can be found in [5]. Usually, the accuracy and validity of these methods depend upon the assumptions that the obstacle is "small" and that only one mode is present. In principle, no such restrictions are needed for the method described in this paper. In practice, multimode propagation requires a larger wave number which demands a finer mesh for resolution. This will increase computer running time and become costly. Another problem arises from the complexity of the boundary conditions. In this paper we present only the first two elements (in the infinite sequence of boundary operators) which are easily implemented and are sufficient to handle two propagating modes.

Consider the parallel plane wave guide shown in Fig. 1 where the dimensionless variables y and z have been scaled with respect to the width (a) of the real guide. We wish to determine the transmission and reflection coefficients produced by a scattering obstacle $\mathcal{R}$ placed in the wave guide. This obstruction may be a dielectric target, a metal object, or an iris. The incident wave which scatters off the obstacle is assumed to be given and is usually the lowest mode allowed in the wave guide. The fields are taken to be time harmonic and proportional to $\exp(-i\omega t)$.

* Received by the editors April 20, 1981, and in revised form December 15, 1981. This research was supported by the U.S. Air Force Office of Scientific Research under contract 80-0016, the Office of Naval Research under contract N00014-80-0196, and the U.S. Department of Energy under contract DE-AC02-78ERO-4650.

† Department of Engineering Sciences and Applied Mathematics, the Technological Institute, Northwestern University, Evanston, Illinois 60201.

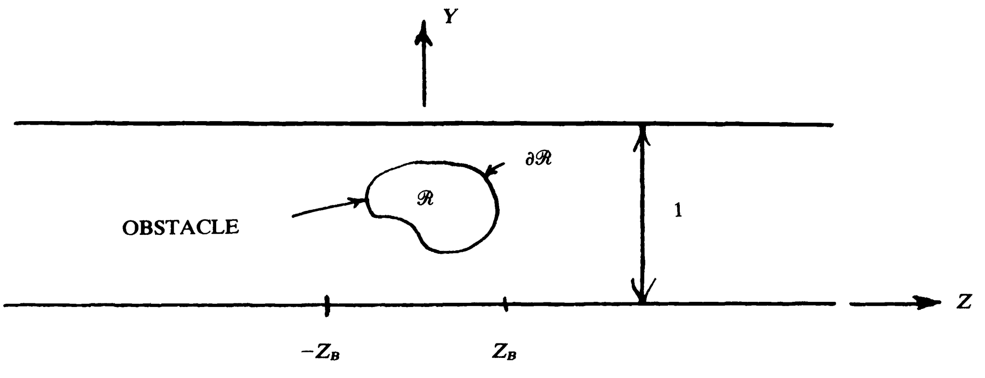


FIG. 1. *Schematic representation of the wave guide configuration.*

We are therefore concerned with solving the Helmholtz equation

$$u_{zz}+u_{yy}+k^2n^2u=0, \qquad |z|<\infty, \quad 0<y<1 \tag{1.1}$$

subject to

$$u(0,z)=u(1,z)=0, \qquad |z|<\infty \tag{1.2}$$

where $k = a\omega/c_0$ and c_0 is the constant wave speed in the guide away from the obstacle. If the obstacle is a dielectric, then n^2 is prescribed. If the obstruction is metallic, then $n^2=1$ and

$$u=0 \quad \text{on } \partial\mathcal{R}. \tag{1.3}$$

In either case we have for $z>z_B$

$$u=e^{-ik_1z}\sin \pi y+\sum_{l=1}^{\infty} R_l e^{ik_lz}\sin l\pi y \tag{1.4}$$

where $k_l=\sqrt{k^2-l^2\pi^2}$ for $l=1, 2, 3, 4, \cdots$. The first term is the incident mode while the infinite sum is the reflected field, u_R. For $z<-z_B$, u is given by

$$u=\sum_{l=1}^{\infty} T_l e^{-ik_lz}\sin l\pi y, \tag{1.5}$$

which is just the transmitted field u_T. The problem is to determine the transmission coefficients T_l and the reflection coefficients R_l. These coefficients depend upon the incident mode and the obstacle. They are interrelated by the conservation law

$$k_1-\sum_{l=1}^{M} k_l\{|T_l|^2-|R_l|^2\}=k^2\iint_{\mathcal{R}} |u|^2\,\mathrm{Im}\,(n^2)\,dxdy \tag{1.6}$$

where $M=\min\{l|l^2\geqq k^2/\pi^2\}$ and $\mathrm{Im}\,(n^2)$ is the imaginary part of n^2. When the obstacle is "metallic" or a lossless dielectric, the term on the right-hand side vanishes and (1.5) implies that the flux is conserved in the waveguide.

2. Numerical boundary conditions. We first restrict k to lie in the interval $(\pi, 4\pi)$ which gives $M=1$ and insures that only the lowest mode propagates for $|z|>z_B$ (i.e.,

$|ik_l| = -|k_l|$, $l \geqq 2$). Let us define the operator B_l by

$$B_l = \frac{\partial}{\partial z} - ik_l, \qquad l = 1, 2, \cdots. \tag{2.1}$$

Applying B_1 to u_R gives

$$B_1 u_R = i \sum_{l=2}^{\infty} (k_l - k_1) R_l e^{ik_l z} \sin l\pi y = O(e^{-|k_2| z}). \tag{2.2}$$

If we apply the condition $B_1 u_R = 0$ at $z = z_\infty < \infty$, then we introduce an error which can be made small by choosing z_∞ large enough. Keeping in mind that a numerical solution is sought, this shows the tradeoff between a larger numerical grid and a smaller boundary error. Actually we can do much better, for

$$B_2 B_1 u_R = O(e^{-|k_3| z_\infty}) \tag{2.3}$$

and $|k_3| > |k_2|$ for any k. Thus for a fixed allowable error, we can pull z_∞ closer to the origin and have a smaller numerical domain. Explicitly, we take

$$B_2 B_1 u_R = i(k_1 + k_2)\frac{\partial u_R}{\partial z} + (k^2 + k_1 k_2) u_R + \frac{\partial^2 u_R}{\partial y^2} = 0 \quad \text{at } z = z_\infty. \tag{2.4}$$

This result follows from (2.1) and (1.1). Equation (2.4) is the boundary condition we use in our numerical method described in § 3. It is interesting to note that this procedure can be continued any number of times. For any $N \geqq 1$ we find

$$\left(\prod_{i=1}^{N} B_i\right)(u_R) = (-i)^N \sum_{l=N+1}^{\infty} P_{Nl} R_l e^{ik_l z} \sin l\pi y \tag{2.5}$$

where $P_{Nl} = \prod_{i=1}^{N} (k_i - k_l)$. This formula is analogous to the boundary conditions presented in [3] and [2] for exterior scattering problems. It is important to note that our boundary conditions are local in nature when the right side of (2.5) is neglected. They are different from the global conditions presented in [4].

The same calculations can be performed on the transmitted field u_T. We find that

$$\left(\prod_{m=1}^{N} D_m\right) u_T = (i)^N \sum_{l=N+1}^{\infty} P_{nl} T_l e^{-ik_l z} \sin l\pi y \quad \text{at } z = -z_\infty \tag{2.6}$$

where $D_m = \partial/\partial z + ik_m$. We shall use this formula with $N = 2$ and take

$$-i(k_1 + k_2)\frac{\partial u_T}{\partial z} + (k^2 + k_1 k_2) u_T + \frac{\partial^2 u_T}{\partial y^2} = 0 \quad \text{at } z = -z_\infty. \tag{2.7}$$

When k lies in the interval $(4\pi, 9\pi)$ two modes propagate. To introduce a small boundary error we must either take $N = 3$ in (2.5) and (2.6) or increase the size of z_∞. We choose the latter tactic since the formula for $B_3 B_2 B_1 u_R$ becomes cumbersome. We end this section by stating formulae for the first two reflection and transmission coefficients when $4\pi < k < 9\pi$. They are

$$R_1 = \frac{-iB_2(u_R)\, e^{-ik_1 z_\infty}}{k_1 - k_2} + \varepsilon, \tag{2.8}$$

$$R_2 = \frac{iB_1(u_R)\, e^{-ik_2 z_\infty}}{k_1 - k_2} + \varepsilon, \tag{2.9}$$

$$T_1 = \frac{+iD_2(u_T)\,e^{+ik_1 z_\infty}}{k_1 - k_2} + \varepsilon, \tag{2.10}$$

$$T_2 = \frac{-iD_1(u_T)\,e^{+ik_2 z_\infty}}{k_1 - k_2} + \varepsilon, \tag{2.11}$$

where $\varepsilon = O(e^{-|k_3| z_\infty})$. These results follow from (1.4), (1.5), (2.1) and the definition of D_m. The functions u_T and u_R are evaluated at $y = \frac{1}{2}$ for R_1 and T_1 and at $y = \frac{1}{4}$ for R_2 and T_2.

3. The numerical method. Our problem is to solve (1.1) subject to (1.2), (2.4) and (2.7) numerically on the the finite domain $0 < y < 1$, $|z| < z_\infty$. We find it convenient to set

$$u = e^{-ik_1 z} \sin \pi y + v(y, z) \tag{3.1}$$

in this region, where v satisfies

$$v_{zz} + v_{yy} + k^2 n^2 v = k^2(1 - n^2)\,e^{-ik_1 z} \sin \pi y, \tag{3.2}$$

$$v(0, z) = v(1, z) = 0, \tag{3.3}$$

$$B_2 B_1 v = 0 \quad \text{at } z = z_\infty, \tag{3.4}$$

$$D_2 D_1 v = 0 \quad \text{at } z = -z_\infty. \tag{3.5}$$

If the scatterer is not a dielectric, $n = 1$, and the additional condition

$$v = -\{e^{-ik_1 z} \sin \pi y\} \quad \text{on } \partial\mathcal{R}. \tag{3.6}$$

is necessary.

This is a well posed problem on a finite domain. There are many numerical methods available for solving the elliptic p.d.e. (3.2). We shall use an iterative scheme based on the limiting amplitude principle [1]. This principle states that the solution of a hyperbolic partial differential equation with a periodic forcing function, e^{-ikt}, will approach a time periodic solution with the same period as $t \to \infty$.

The equation we solve numerically is

$$n_1^2 w_{tt} = w_{zz} + w_{yy} + k^2(n^2 - n_1^2) w + k^2(n^2 - 1)\,e^{-i(k_1 z + kt)} \sin \pi y \tag{3.7}$$

subject to the boundary conditions

$$w(0, z) = w(1, z) = 0, \tag{3.8}$$

$$w = \{-e^{-ik_1 z} \sin \pi y\}\,e^{-ikt} \quad \text{on } \partial\mathcal{R} \ (\text{if } n^2 = 1), \tag{3.9}$$

$$w_z + \theta w_t = \beta \int_0^t w_{yy}\,dt' \quad \text{at } z = z_\infty, \tag{3.10}$$

$$w_z - \theta w_t = -\beta \int_0^t w_{yy}\,dt' \quad \text{at } z = -z_\infty \tag{3.11}$$

and the initial conditions

$$w(y, z, 0) = w_t(y, z, 0) = 0 \tag{3.12}$$

where $\theta = (k^2 + k_1 k_2)/k(k_1 + k_2)$ and $\beta = k/(k_1 + k_2)$.

The time dependent boundary conditions (3.10)–(3.11) were obtained by replacing B_l and D_l with

$$\frac{\partial}{\partial z}+\frac{k_l}{k}\frac{\partial}{\partial t} \quad \text{and} \quad \frac{\partial}{\partial z}-\frac{k_l}{k}\frac{\partial}{\partial t},$$

respectively, in equations (3.4) and (3.5). The resulting expressions were integrated to give (3.10) and (3.11) with the aid of (3.12). Now (3.7) is a wave equation with a potential and it may produce bound states which either grow exponentially in time or oscillate in time with some frequency other than k. In Appendix 1 we show that trapped modes do not exist when

$$n_1^2 \geqq \max\{(k^2n^2-\pi^2)/k_1^2, n^2\}. \tag{3.13}$$

Thus the limiting amplitude principle holds [1], and $w \to e^{-ikt}v(y, z)$ as $t \to \infty$ where v satisfies (3.2)–(3.6).

We next replace B_1, D_1, B_2, and D_2 in (2.8)–(2.11) by their time dependent forms, mentioned above. For example, R_1 is now given by

$$R_1=-i\left[\frac{\partial w}{\partial z}+i\frac{k_1}{k}\frac{\partial w}{\partial t}\right]\frac{e^{ik_1z_\infty}}{(k_1-k_2)} \tag{3.14}$$

where w is evaluated at $y=\frac{1}{2}$. Note that as $t\to\infty$, $w \to v\,e^{-ikt}$ and (3.13) reduces to (2.8).

4. The difference scheme. To solve the time dependent differential equation (3.7) imposing the boundary conditions (3.10) and (3.11), we have used a standard centered difference scheme for the initial value problem of a second order hyperbolic equation with second order accuracy. Let the solution to the difference scheme be $\tilde{w}(j, m, n)$ where (j, m, n) are evaluated on a grid

$$z=-z_\infty+j\,\Delta z, \quad y=m\,\Delta y, \quad t=n\,\Delta t \tag{4.1}$$

with $0 \leqq j \leqq N$, $0 \leqq m \leqq M$. Then for an interior point

$$\tilde{w}(j, m, n+1)=T(j\pm1, j, m\pm1, m, n, n-1) \tag{4.2}$$

is determined from the values of $\tilde{w}$ at $(j\pm1, m\pm1, n)$, (j, m, n), $(j, m, n-1)$. On the boundary $z=N\,\Delta z-z_\infty=z_\infty$ we use the differenced form of (3.10)

$$\begin{aligned}\frac{\theta}{2\Delta t}\{\tilde{w}(N, m, n+1)-\tilde{w}(N, m, n-1)\}+\frac{1}{2\Delta z}\{\tilde{w}(N+1, m, n)-\tilde{w}(N-1, m, n)\}\\ =\beta\Phi(N, m, n)\end{aligned} \tag{4.3}$$

where $\Phi(N, m, n)=\int_0^{t_n}\Delta_y\tilde{w}\,dt'$ and $\Delta_y\tilde{w}$ is the difference approximation of w_{yy}. In implementing (4.3) we use the fact that

$$\Phi(N+1, m, n)=\Phi(N, m, n)+\frac{\Delta t}{2}[\Delta_y w(N)+\Delta_y w(N-1)]+\theta(\Delta t^2), \tag{4.4}$$

which is just the trapezoidal rule. The value of $\tilde{w}$ at $j=N+1, m, n$ has to be eliminated by use of the difference equation (4.2). Thus we obtain

$$\tilde{w}(N, m, n+1)=B(N, m, m\pm1, n, n-1). \tag{4.5}$$

A similar result holds for $\tilde{w}(1, m, n+1)$.

It still remains to apply the Dirichlet condition if there is an obstacle in the wave guide. We simply use the discrete version of (3.9),

$$\tilde{w} = -\sin \pi y_{m'}\, e^{-i(k_1 z_{j'} + kt_n)}, \tag{4.6}$$

where (j', m') are mesh points on the body.

In closing this section, we shall describe our numerical method for determining when $\tilde{w}$ has reached its time harmonic steady state. Recall that the reflected field $v(z > z_B)$ is given by

$$v = \tilde{w}\, e^{iwt}$$

for large time. Thus, for large time, $|\tilde{w}|$ becomes independent of time. We terminate our computations when

$$\max_{\substack{0 \leqq j \leqq N \\ 0 \leqq i \leqq M}} \|\tilde{w}(n+1, j, i) - \tilde{w}(n, j, i)\| < \varepsilon$$

for some prescribed $\varepsilon > 0$.

5. Numerical experiments. Before proceeding to describe several numerical experiments, we list here the parameters which remain constant throughout: $\Delta y = \Delta z = 0.1$, $\Delta t = 0.05$, $M = 10$ and $\varepsilon = .01$. The first problem we tried was physically trivial but was nonetheless fruitful from a numerical point of view. We placed a metallic barrier, which completely closed the guide, at $z = 0$ and set $n^2 = n_1^2 = 1$. The exact solution gives $R_1 = -1$, $T_1 = 0$, $T_i = R_i = 0$ for $i \geqq 2$. Our code reproduced these results nicely with an error of 2% for both R_1 and T_1 and an error of 0.1% for T_i and R_i when $i \geqq 2$. We found that varying z_∞ from 1.5 to 3.0 had little effect on the output except by increasing the running time. More importantly, we found that differentiating (3.10) and (3.11) and numerically implementing the second order boundary conditions gives the same numerical answer to the third decimal place. However, if one time derivative is replaced by $-ik$ to yield a first order boundary condition, then the numerical scheme becomes divergent.

The second problem we studied was more interesting both numerically and physically. A metal barrier which only partially filled the guide was placed at $z = 0$. That is, $\mathcal{R} = \{(y, z) | z = 0,\ 0 < a \leqq y \leqq 1\}$ where $1 - a$ is the barrier's height. In Fig. 2 the results for $a = \frac{1}{2}$ and $z_\infty = 2$ are shown. In this picture the numbers printed at the mesh points are ten times the total field. Thus the dark regions correspond to constructive interference, the light to destructive. Notice that the barrier casts a shadow

```
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 0 0 1 3 4 4 4 3 1 0 2 3 4 5 5 4 3 1 0 2 3 4
 4 3 3 3 3 3 3 4 3 4 4 4 4 4 4 4 3 3 2 1 0 3 7 8 9 8 6 3 1 4 7 91010 8 6 3 1 4 7 9
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 4 3 2 0 510121211 8 4 2 5 912141311 8 4 2 5 912
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 5 5 3 0 81215151310 5 2 61114161614 10 5 2 71114
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 71115171714 11 5 2 71115171714 10 5 2 71215
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 81013161716141 0 5 2 61114161614 10 5 2 61114
 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6 7 81012141514 12 9 5 1 5 912131311 8 4 2 5 912
 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 4 4 5 7 810111111 9 6 3 1 3 6 810 9 8 6 3 1 4 7 9
 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 2 3 4 4 5 6 6 5 5 3 2 0 1 3 4 5 5 4 3 1 0 2 3 4
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 |                                       |                                       |
-2                                       0                                       2
```

FIG. 2. *The total field's amplitude, $|u|$, created by the lowest incident mode with $k = 3\pi/2$, $\Delta z = \Delta y = 0.1$ and $z_\infty = 2$. The obstacle $\mathcal{R}$ is a metallic strip occupying the set of points $\{(z, y) | z = 0, \frac{1}{2} \leqq y \leqq 1\}$. The dark regions correspond to constructive interference, while the light regions correspond to destructive.*

TABLE 1
The reflection and transmission coefficients' magnitudes for $k=3\pi/2$, $\Delta z=\Delta y=0.1$, and $z_\infty=2$. The parameter $1-a$ is the length of the metal strip which partially fills the guide at $y=0$.

a	$\lvert R_1\rvert$	$\lvert T_1\rvert$
1.0	0.0	1.00
0.8	0.07	0.98
0.5	0.74	0.68
0.2	0.98	0.07
0.0	1.00	0.00

even though $k=3\pi/2$ (a moderate wave number) for this experiment. We also find that $R_1=0.74$, $T_1=0.68$, and $R_i=T_i=O(0.001)$ for $i\geqq 2$ which satisfies (1.6) to two places. We next varied a while keeping k fixed. The results of these calculations are shown in Table 1. Each run took about 23 seconds of CPU time on a CDC 7600. For each value of a we increased z_∞ to 3 and found that the results changed only in the third decimal place. As a final variation of this problem, a was fixed at 0.5 and k was increased to $5\pi/2$. At this value of k two modes can propagate in the wave guide. Since the lowest mode excites the wave guide and scatters from the barrier, we expect energy to be coupled into the second mode. Indeed we find that $|R_1|=0.50$ and $|R_2|=0.56$. In these experiments $n_1^2=n^2=1$ and (3.7) just becomes the wave equation.

The next problem we studied was quite different from a physical point of view. A dielectric block of width 2 and height 1 was placed in the guide and centered at the origin. We took $n^2=\frac{1}{3}$, $n_1^2=1$, $k=3\pi/2$, and $z_\infty=3$. This situation models a pocket of gas trapped in the wave guide. The solution follows from separation of variables and shows that the wave is evanescent in the block and that very little energy tunnels through. The results of our calculation are shown in Fig. 3 and are in agreement

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 2 2 3 3 4 5 5 6 5 4 2 0 1 3 5 6 6 5 4 2 0 1 3 5 6 6
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 2 3 4 4 5 6 8 9111110 8 4 0 3 7 9111110 7 4 0 3 7101111
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 3 4 4 5 6 8 9111315161411 6 1 4 91315161410 6 0 510131616
0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 3 3 4 5 6 8 9111315181617l3 7 1 5111518181612 7 0 611161818
0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 3 4 4 5 7 81011131619191713 8 1 5111619191713 7 0 612171919
0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 3 3 4 5 6 8 9111315181817l3 7 1 5111518181612 7 0 611161818
0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 3 4 4 5 6 8 9111315161411 6 1 4 91315161410 6 0 510131616
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 2 2 2 3 4 4 5 6 8 9111110 8 4 0 3 7 9111110 7 4 0 3 7101111
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 2 2 3 3 4 5 5 6 5 4 2 0 1 3 5 6 6 5 4 2 0 1 3 5 6 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          |                   |                   |                                       |
         -1                   0                   1                                       3
```

FIG. 3. *The total field's amplitude, $|u|$, created by the lowest incident mode with $k=3\pi/2$, $\Delta z=\Delta y=0.1$, and $z_\infty=3$. The obstacle $\mathscr{R}$ is a dielectric block of $n^2=\frac{1}{3}$. It occupies the region $\{(z,y)|0\leqq y\leqq 1, -1\leqq z\leqq 1\}$.*

with the exact answer. In particular we found that $|R_1|=0.97$, $|T_1|=0.0$, and $|T_i|=|R_i|=O(0.001)$ for $i\geqq 2$, which are in error by at most 3%. A more interesting example which cannot be solved exactly occurs when the block fills the region $\mathscr{R}=\{(y,z)|0.2\leqq y\leqq 0.8, |z|\leqq 1\}$. The results of this numerical experiment are shown in Fig. 4. For this case $|R_1|=0.93$, $|T_1|=0.36$ and $|R_i|=|T_i|=O(0.001)$ for $i\geqq 2$. In both cases the running time was about 45 seconds.

We next placed a dielectric block of width 2, height 1, and $n^2=\frac{8}{9}$ into the guide and centered it at $z=0$. Again this situation is amenable to exact solution, but now

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 2 2 2 3 3 4 5 5 6 6 5 4 2 0 2 4 5 6 6 5 3 1 0 2 4 5 6 6
0 0 0 0 0 0 0 0 0 0 1 1 1 1 2 2 2 3 3 4 5 6 6 7 910111110 7 4 0 4 710111110 7 3 0 4 7101111
0 0 0 0 0 0 0 0 1 1 1 1 2 2 2 3 3 4 5 6 7 8 910121316161410 5 0 5101416151310 5 0 610141615
0 0 0 0 0 0 0 0 1 1 1 2 2 2 3 4 4 5 6 7 8 91012141618181612 6 0 6121618181611 5 0 712161818
0 0 0 0 0 0 0 1 1 1 1 2 2 3 3 4 4 5 6 7 8 91112141719191713 6 0 7131719191712 6 0 713171919
0 0 0 0 0 0 0 0 1 1 1 2 2 2 3 4 4 5 6 7 8 91012141618181612 6 0 6121618181611 5 0 712161818
0 0 0 0 0 0 0 0 1 1 1 1 2 2 2 3 3 4 5 6 7 8 910121316161410 5 0 5101416151310 5 0 610141615
0 0 0 0 0 0 0 0 0 0 1 1 1 1 2 2 2 3 3 4 5 6 6 7 910111110 7 4 0 4 710111110 7 3 0 4 7101111
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 2 2 2 3 3 4 5 5 6 6 5 4 2 0 2 4 5 6 6 5 3 1 0 2 4 5 6 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
          |                   |                   |                                       |
         -1                   0                   1                                       3
```

FIG. 4. *The same problem and parameters as Fig.* 3 *except* $\mathcal{R}$ *now fills the smaller region* $\{(z, y)|0.2 \leqq y \leqq 0.8, -1 \leqq z \leqq 1\}$.

the block allows the wave to propagate. In fact, when $k = 3\pi/2$ there is total transmission and $R_1 = 0$. In our numerical experiment, with $n_1^2 = 2$ for $|z| \leqq 1$, $n_1^2 = 1$ for $|z| \geqq 1$, and $z_\infty = 3$, we found agreement to within 3%.

As a final experiment and variation on the previous examples, we placed a dielectric block with $n^2 = \frac{13}{9}$ in the region $\mathcal{R} = \{(y, z)|0 \leqq y \leqq 1, 0 \leqq z \leqq 1\}$ and put a metal barrier at $z = 0$ with $a = 1$. We choose $k = 3\pi/2$, $z_\infty = 3$, $n_1^2 = 2$ for $0 \leqq z \leqq 1$, and $n_1^2 = 1$ for $z > 1$. This problem is also amenable to an exact solution which gives $|R_1| = 1.0$. The results of our experiment are shown in Fig. 5 with $|R_1| = 0.97$, a mere 3% off the true value.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 4 5 5 4 1 1 3 5 6 5 3 2 0 2 4 5 6 6 5 3 1 0 2 4 5 6 5 5 3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 91110 8 3 2 6101110 7 3 0 4 710111110 7 3 0 4 7101111 9 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 712151511 4 2 914151310 5 0 5101416151310 5 0 61014151513 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 815181712 5 31116181611 6 0 7121618181611 5 0 712161818151 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 816191813 5 31117191712 6 0 7131719191712 6 0 7131719191611
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 815181712 5 31116181611 6 0 7121618181611 5 0 7121618181511
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 712151511 4 2 914151310 5 0 5101416151310 5 0 61014151513 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 5 91110 8 3 2 6101110 7 3 0 4 710111110 7 3 0 4 7101111 9 6
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 4 5 5 4 1 1 3 5 6 5 3 2 0 2 4 5 6 6 5 3 1 0 2 4 5 6 5 5 3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                              |                   |                                       |
                              0                   1                                       3
```

FIG. 5. *The total field's amplitude,* $|u|$, *created by the lowest incident mode with* $k = 3\pi/2$, $\Delta y = \Delta z = 0.1 z_\infty = 3$. *The obstacle* $\mathcal{R}$ *is a dielectric block of* $n^2 = \frac{13}{9}$ *which occupies the region* $\{(z, y)|0 \leqq y \leqq 1, 0 \leqq z \leqq 1\}$. *There is a metal strip at* $z = 0$ *which completely shunts the wave guide.*

Appendix. Consider the wave equation (3.7) without the forcing term $k^2(n^2 - 1) \cdot \sin \pi y \cdots$ and look for solutions which approach zero exponentially as $z \to \pm\infty$. These localized or "trapped" modes automatically satisfy the boundary conditions (3.10) and (3.11) in this limit. Set $w = q\, e^{\lambda t}$ where q must satisfy

$$q_{zz} + q_{yy} + k^2(n^2 - n_1^2)q = \lambda^2 n_1^2 q. \tag{A.1}$$

Multiplying this equation by q, applying the boundary conditions at $y = 0, 1$, and using Green's theorem, one finds that

$$-\int_R [|\nabla q|^2 + k^2(n_1^2 - n^2)q^2]\, dydz = \lambda^2 \int_R n_1^2 q^2\, dydz \tag{A.2}$$

where $R = \{(y, z)|0 \leqq y \leqq 1, |z| < z_\infty\}$. If n_1^2 satisfies (3.13), then (A.2) implies $\lambda^2 < 0$ and there are no growing modes. On the other setting, $w = e^{i\lambda t} q$ and applying the same argument shows that trapped modes, periodic in time, are possible. In this case set $q=\sum_{m=1}^{\infty} q_m(z)\sin m\pi y$ where q_m satisfies

$$\frac{d^2q_m}{dz^2}+[k^2(n^2-n_1^2)-m^2\pi^2]q_m+\lambda^2n_1^2q_m=0. \tag{A.3}$$

Now when $|z|>z_B$ the index of refraction $n^2=1$. Taking $n_1^2=1$ in this region and demanding exponential decay of q_m as $|z|\to\infty$ requires that $\lambda^2<m^2\pi^2$. On the other hand, when $|z|<|z_B|$ the solution may be oscillatory, which would lead to a trapped mode. This can not occur when n_1^2 satisfies (3.13). (It is interesting to note that these localized-oscillatory modes are not present in exterior problems.)

REFERENCES

[1] C. S. MORAWETZ, *The limiting amplitude principle*, Comm. Pure Appl. Math., 15 (1962), pp. 181–197.

[2] G. A. KRIEGSMANN AND C. S. MORAWETZ, *Solving the Helmholtz equation for exterior problems with a variable index of refraction*: I, this Journal, 1 (1980), pp. 371–385.

[3] A. BAYLISS AND E. TURKEL, *Boundary conditions for exterior acoustic problems*, Rep. 79-7, ICASE, NASA Langley Research Center, Hampton, VA, 1979.

[4] G. J. FIX AND S. P. MARIN, *Variational methods for underwater acoustic problems*, J. Comput. Phys., 28 (1978), pp. 253–270.

[5] D. S. JONES, *The Theory of Electromagnetism*, Pergamon Press, Oxford, 1964, pp. 261–287.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

ASPECTS OF NUMERICAL METHODS FOR ELLIPTIC SINGULAR PERTURBATION PROBLEMS*

A. SEGAL†

Abstract. Upwind difference, defect correction and central difference schemes for the solution of the convection-diffusion equation with small viscosity coefficient are compared. It is shown that central difference schemes and hence also standard Galerkin finite element methods are preferable above upwind and defect correction schemes, when Gaussian elimination is used for the solution of the resulting system of equations. When iterative solution methods are employed good results can be achieved by a defect-correction method, whereas upwind difference schemes are generally inaccurate.

Key words. iterative method, convection-diffusion equation, Reynolds number, boundary layer, upwind differencing, elliptic equation

1. Introduction. The purpose of this paper is to analyze numerical problems related with boundary layers such as occur for example in flows governed by the Navier–Stokes equations. The convection-diffusion equation

$$-\varepsilon\Delta\phi+\mathbf{u}\cdot\nabla\phi=f \tag{1.1}$$

will be used as a model problem.

Quite often in applications, $\mathbf{u}$ is a flow velocity vector and ε some diffusion or viscosity coefficient. Hence the appelation "convection-diffusion equation." Henceforth $\mathbf{u}$ will be referred to as the velocity vector and ε as the viscosity coefficient.

The convection-diffusion equation with small ε has been the subject of many papers, see for example Hemker [17], Pearson [23], [24], Il'in [18], Heinrich et al. [15], [16], Griffiths [13], Christie et al. [3], Axelsson and Gustafsson [1] and Chien [2]. It is the aim of this paper to compare the so-called upwind schemes advocated in some of these papers with central difference schemes and hence also with standard Galerkin finite element schemes, and with a predictor-corrector scheme, both for direct solution methods and for iterative solution methods. For the one-dimensional case some theory will be developed. Numerical experiments will be described for both the one-dimensional and the two-dimensional cases.

2. Boundary layers. When $\varepsilon\downarrow 0$ solutions of (1.1) exhibit boundary layer type behavior. For the purpose of a qualitative discussion we can restrict ourselves without loss of generality to the two-dimensional case, a square region and a velocity $\mathbf{u}=u\mathbf{e}_x$:

$$-\varepsilon\Delta\phi+u\frac{\partial\phi}{\partial x}=q,\qquad (x,y)\in(0,1)\times(0,1). \tag{2.1}$$

Let the following boundary conditions be given:

$$\begin{aligned}\phi(0,y)&=f_1(y),&\quad \phi(1,y)&=f_2(y),\\ \phi(x,0)&=g_1(x),&\quad \phi(x,1)&=g_2(x).\end{aligned} \tag{2.2}$$

(In what follows also a Neumann boundary condition at $x=1$ will be considered.)

For $\varepsilon=0$, (2.1) becomes

$$\frac{\partial\phi}{\partial x}=q. \tag{2.3}$$

* Received by the editors May 26, 1981, and in revised form December 3, 1981.

† Delft University of Technology, Department of Mathematics and Informatics, Postbus 356, 2600 AJ Delft, the Netherlands.

Obviously not all boundary conditions (2.2) can be satisfied. Physical intuition and mathematical theory [5] lead one to maintain the boundary condition $\phi(0, y) = f_1(y)$. In this way the so-called outer solution ϕ_0 is obtained:

$$\phi_0(x, y) = f_1(y) + \int_0^x q(\xi, y)\, d\xi. \tag{2.4}$$

Along the horizontal walls $y = 0$ and $y = 1$, in general, boundary layers occur. The equation governing these boundary layers is obtained by stretching the y-coordinate as follows: $\tilde{y} = y/\sqrt{\varepsilon}$ and by taking the limit as $\varepsilon \downarrow 0$ of (2.1):

$$-\frac{\partial^2 \phi}{\partial \tilde{y}^2} + \frac{\partial \phi}{\partial x} = q. \tag{2.5}$$

The horizontal boundary layers have thickness $O(\sqrt{\varepsilon})$ and take care of the boundary conditions along the horizontal walls. The boundary condition at $x = 1$ generally induces a vertical boundary layer. The governing equation is obtained by the following stretching: $\tilde{x} = x/\varepsilon$. The limit $\varepsilon \downarrow 0$ results in

$$-\frac{\partial^2 \phi}{\partial \tilde{x}^2} + \frac{\partial \phi}{\partial \tilde{x}} = 0. \tag{2.6}$$

Note that this is an ordinary differential equation. The thickness of the boundary layer is $O(\varepsilon)$.

For a justification of the above statements the reader is referred to [5], or to a general introduction to singular perturbation theory, such as [4].

For convenience, the horizontal boundary layers will be called parallel boundary layers and the vertical boundary layer will be called the normal boundary layer (because it is normal to the velocity vector **u**). In singularly perturbed ordinary differential equations (such as the one-dimensional case of (1.1)) only the normal boundary layer occurs. Both types will be studied in the sequel.

3. The normal boundary layer. For a study of the normal boundary layer we restrict ourselves to the one-dimensional analogue of (2.1):

$$-\varepsilon \frac{d^2 \phi}{dx^2} + u \frac{d\phi}{dx} = 0, \qquad x \in (0, 1), \quad \phi(0) = 0, \quad \phi(1) = 1. \tag{3.1}$$

with ε and u constant. The exact solution of (3.1) has been plotted in Fig. 3.1. Equation (3.1) can be discretized either by a finite difference method (FDM) or a finite element method (FEM).

Central difference schemes with constant mesh size h may give rise to oscillations (see for example Fig. 3.2) depending on the step size chosen. One can show [27] that in order to have monotone solutions for a difference scheme with constant mesh size applied to (3.1), it is necessary and sufficient that the corresponding matrix be diagonally dominant. This yields the well-known condition for the so-called grid Reynolds number uh/ε:

$$\frac{uh}{\varepsilon} \leqq 2, \tag{3.2}$$

which for practical purposes may be undesirably restrictive. In particular, it would make the computational cost dependent on ε. One of the purposes of numerical methods for singularly perturbed problems is to have the cost (and the accuracy) independent of ε.

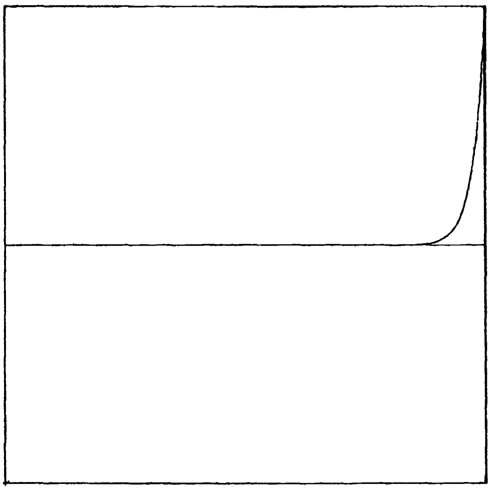


FIG. 3.1. *Exact solution of* (3.1), $\varepsilon = 0.025$, $u = 1$.

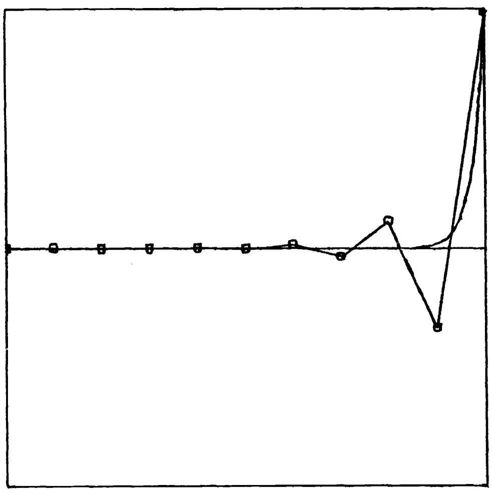


FIG. 3.2. *Numerical solution of* (3.1) *using central differences,* $\varepsilon = 0.025$, $u = 1$, $h = 0.1$, —— *exact solution,* —□— *numerical solution.*

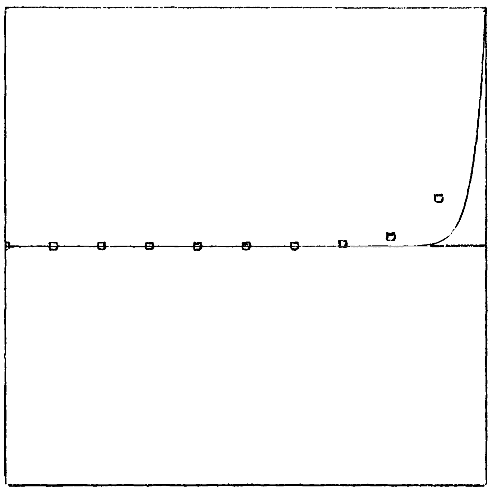


FIG. 3.3. *Solution of* (3.1) *using the backward difference scheme* (3.3). $\varepsilon = 0.025$, $u = 1$, $h = 0.1$, —— *exact solution,* □ *numerical solution.*

The simplest method to avoid oscillations is to approximate $d\phi/dx$ by backward differences if $u > 0$ and by forward differences if $u < 0$. In the literature this method is known as "upwind differencing." The solution of this method is monotone, yet far less "steep" than the exact solution (Fig. 3.3). One easily verifies that this upwind scheme is identical to the solution of the following differential equation:

$$-\left(\varepsilon + \frac{uh}{2}\right)\frac{d^2\phi}{dx^2} + u\frac{d\phi}{dx} = 0, \qquad \phi(0) = 0, \quad \phi(1) = 1, \tag{3.3}$$

with a central difference scheme. The term $-(uh/2)(d^2\phi/dx^2)$ will be called the artificial viscosity term and the introduction of this artificial viscosity term has a "smoothing" effect on the solution. In fact we solve a problem with greater viscosity.

In the literature more accurate upwind schemes have been derived, see for example [18], [1] and [26]. The most accurate scheme for equation (3.1) is the so-called Il'in scheme [18], [17], [2]. This method supplies equation (3.1) with an artificial viscosity term $-\alpha\, d^2\phi/dx^2$ such that the central difference scheme applied to

$$-(\varepsilon+\alpha)\frac{d^2\phi}{dx^2}+u\frac{d\phi}{dx}=0, \qquad \phi(0)=0, \quad \phi(1)=1, \tag{3.4}$$

yields the exact solution in the nodal points. One easily verifies that this is the case when:

$$\alpha=-\varepsilon+\frac{\varepsilon}{G}, \quad G=\frac{2}{R}\left\{1-\frac{2(e^R-1)}{e^{2R}-1}\right\}, \quad R=\frac{uh}{\varepsilon}. \tag{3.5}$$

The Il'in scheme is of course as accurate as possible for equation (3.1). However, when variable coefficients are used its accuracy decreases, although a good approximation remains possible as long as the coefficients vary not too much. Il'in's method can be extended to more dimensional rectangular regions but it will improve the accuracy only in normal boundary layers and not in parallel boundary layers. Furthermore, a generalization to nonequidistant meshes is not known.

Remark. One can show [27] that in order to get a diagonally dominant matrix it is necessary to introduce an artificial viscosity $-\alpha\, d^2u/dx^2$ with

$$\alpha \geqq \min\left(0, \frac{uh}{2}-\varepsilon\right). \tag{3.6}$$

The oscillations observed for the central difference scheme are not peculiar to the FDM. In applications of the FEM the same phenomenon occurs (see [15], [13], [16], [3], [29], [21], [20], [11], [6], [19]). This is to be expected, since the FEM can be regarded as a tool to construct finite difference equations. For example, the Galerkin method together with linear elements applied to (3.1) results in the central difference scheme. So exactly the same results can be expected. Higher order elements give oscillations also [11], [6], [19].

The literature has described how to construct finite element models that are equivalent with upwind differencing schemes. This approach is described in [15], [13], [16], [3]. The method used is the so-called Petrov–Galerkin method (see [13]), in which the test functions differ from the basis functions. These methods appear to be of the same character as the Il'in method and subject to the same restrictions.

A possible alternative is the introduction of a variable artificial viscosity term that is constructed elementwise such that the diagonal of the element matrix is nonnegative. Such an approach can be programmed simply and always gives diagonally dominant matrices. This aspect will not be pursued further in this paper.

4. Absence of normal boundary layer. The question arises whether the numerical oscillations described in the previous section are caused by the presence of the normal boundary layer, rather than by the fact that the matrix is not diagonally dominant. In order to investigate this question we study an example in which no boundary layer is present:

$$-\varepsilon\frac{d^2\phi}{dx^2}+u\frac{d\phi}{dx}=u(1-2x)+2\varepsilon, \qquad \phi(0)=\phi(1)=0. \tag{4.1}$$

The exact solution of (4.1) is given by $\phi(x)=x(1-x)$, and hence the equation is exactly solved by the central difference scheme, for every mesh size. A less trivial example is the following:

$$(4.2)\qquad -\varepsilon\frac{d^2\phi}{dx^2}+u\frac{d\phi}{dx}=\varepsilon\pi^2\sin(\pi x)+u\pi\cos(\pi x),\qquad \phi(0)=\phi(1)=0.$$

The exact solution of (4.2) is given by $\phi(x)=\sin(\pi x)$.

In Figs. 4.1 and 4.2 the solution of (4.2) has been plotted as computed with central differences and the Il'in scheme, respectively. The step size h is chosen such that the central difference matrix is not diagonally dominant. These figures show that the central difference scheme is very accurate; the Il'in method introduces some damping; in fact, it tries to generate a boundary layer. Moreover, although the central difference matrix is not diagonally dominant, no oscillations occur (see [27]).

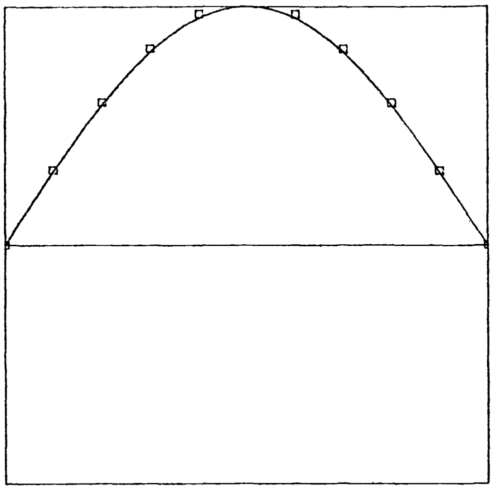


FIG. 4.1. *Central differences.*

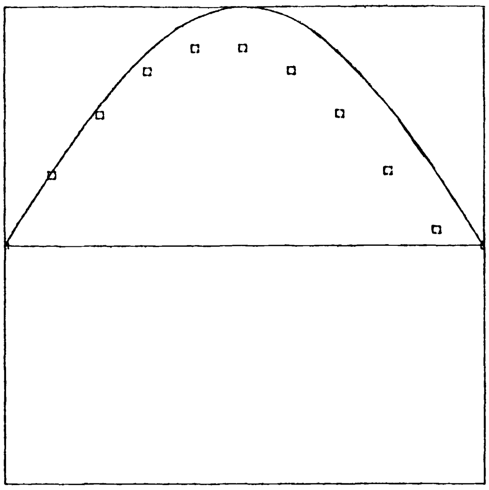


FIG. 4.2. *The Il'in scheme.*

$h=0.1$, $\varepsilon=0.025$, $u=1$. —— *exact solution,* □ *numerical solution.*

An error estimate in the case that a boundary layer is absent is given by the following theorem:

THEOREM 4.1. *Let ϕ satisfy*:

$$(4.3)\qquad L\phi=-\varepsilon\frac{d^2\phi}{dx^2}+u\frac{d\phi}{dx}=f,\qquad x\in(0,1),$$

$\phi(0)$ and $\phi(1)$ given, with ε and u constants, $u>0$. Let ϕ be ε-independent and $\phi\in C^6(0,1)$, i.e., f depends on ε. Let ϕ_h satisfy

$$(4.4)\qquad L_h\phi_h=-\tilde\varepsilon\frac{\phi_{i-1}-2\phi_i+\phi_{i+1}}{h^2}+u\frac{\phi_{i+1}-\phi_{i-1}}{2h}=f_i,\qquad i=1,2,\cdots,n,$$

with ϕ_0 and ϕ_{n+1} given. $\tilde\varepsilon$ may be ε or the result of upwind differencing, for example in (3.3) $\tilde\varepsilon=\varepsilon+uh/2$.

Then the error $\phi-\phi_h$ satisfies:

(i) *If $\tilde\varepsilon=O(h)$, $\varepsilon=o(h)$ and the matrix corresponding to L_h is diagonally dominant then*

$$\|\phi-\phi_h\|_2=O(h).$$

(ii) *If $\tilde{\varepsilon} = \varepsilon$ then*

$$\|\phi - \phi_h\|_2 = O(h^2) + O(h^4/\varepsilon).$$

(iii) *If $\tilde{\varepsilon} = O(h^2)$ and $\varepsilon = o(h^2)$, then*

$$\|\phi - \phi_h\|_2 = O(h^2).$$

For a proof of this theorem see Appendix 1.

From these error estimates we see that whenever $h = o(\varepsilon^{1/3})$, the central difference scheme is more accurate than the upwind scheme for smooth solutions. For $\varepsilon = o(h^3)$ the central difference scheme becomes inaccurate when Dirichlet boundary conditions are prescribed on the outflow boundary $x = 1$. In that case more accurate results are obtained with $\tilde{\varepsilon} = O(h^2)$. Such a scheme is not an upwind scheme in the classical sense, since the discretization matrix is not diagonally dominant; however, since it introduces artificial viscosity, we shall call it a modified upwind scheme in the sequel.

Remark 4.1. Usually one considers ε fixed and h tending to zero, thus $\varepsilon = O(1)$ in h. But since in practice one takes only a few values of ε and h, both of which are small quantities, one can consider the difference equations and their solutions as both ε and h tend to zero. It is in this sense that we write $\varepsilon = O(h^\alpha)$. Thus we are not considering the difference schemes in the usual sense of consistency with ε fixed and h tending to zero.

Remark 4.2. One can prove [27] that in the case of example (4.2) the error is given by $\|\phi - \phi_h\|_2 = O(h^2)$.

Remark 4.3. Numerical computations (see § 7) show that when Neumann boundary conditions are prescribed at $x = 1$, the accuracy of the central difference scheme is $O(h^2)$, in the case of smooth solutions. Furthermore, from the proof of Theorem 4.1 we can see that the $O(h^4/\varepsilon)$ error in the case of Dirichlet boundary conditions arises from a boundary layer due to the truncation error. Hence we may conclude that the numerical oscillations in § 3 are caused by the presence of the normal boundary layer and the fact that Dirichlet boundary conditions are given at the outflow boundary.

5. Local mesh refinement. In the preceding sections it was found that in the presence of a normal boundary layer, or a numerical boundary layer due to the truncation error, the central difference scheme gives rise to numerical oscillations. These oscillations are absent when there is not such a boundary layer. These results suggest that the oscillations may be suppressed by local mesh refinement in the boundary layer. Therefore, equation (3.1) has been solved with a graded mesh.

From the exact solution we can deduce that the boundary layer is smaller than 8ε, and therefore we divide the region into 2 subregions,

$$[0, 1-8\varepsilon], \qquad [1-8\varepsilon, 1]. \tag{5.1}$$

In each subregion, an equidistant mesh is chosen consisting of $(n+1)/2$ nodal points. In Table 5.1 the l_∞-error is presented for various values of ε and n. The difference formulas used are the trivial extensions of the classical schemes to nonuniform meshes.

In order to apply the Il'in scheme to nonuniform meshes, the following modification of the formula was necessary:

$$\begin{gathered} g_i = \frac{2}{R_i}\left\{1 - \frac{2(e^{R_i}-1)}{e^{2R_i}-1}\right\}, \qquad R_i = \frac{uh_i}{\varepsilon}, \qquad i = 1, 2, \\ (L_h\phi_h)_j = \frac{2}{h_1+h_2}\left\{\frac{\varepsilon}{g_1h_1}\phi_{j-1} - \left(\frac{\varepsilon}{g_1h_1} + \frac{\varepsilon}{g_2h_2}\right)\phi_j + \frac{\varepsilon}{g_2h_2}\phi_{j+1} + u\frac{\phi_{j+1}-\phi_{j-1}}{2}\right\}, \end{gathered} \tag{5.2}$$

with $h_1 = x_j - x_{j-1}$, $h_2 = x_{j+1} - x_j$.

TABLE 5.1
l_∞-error of some schemes for various values of ε, nonequidistant mesh (5.1), $u = 1$.

		$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
Central differences	$n = 21$	0.021	0.021	0.021	0.021	0.021	0.021
	$n = 41$	0.005	0.005	0.005	0.005	0.005	0.005
Backward differences	$n = 21$	0.108	0.109	0.109	0.109	0.109	0.109
	$n = 41$	0.064	0.064	0.064	0.064	0.064	0.064
Il'in scheme	$n = 21$	10^{-7}	10^{-15}	10^{-15}	10^{-15}	10^{-15}	10^{-15}
	$n = 41$	10^{-5}	10^{-15}	10^{-15}	10^{-15}	10^{-15}	10^{-15}

The solution of the central difference scheme oscillates in the region $[0, 1-8\varepsilon]$; however, its absolute value is very small. Table 5.1 shows that the Il'in scheme is also very accurate for the nonequidistant mesh, but that the backward differences are far less accurate than central differences when mesh refinement is applied. Because of the mesh chosen, the amount of work done and the accuracy are independent of ε. The table shows that the accuracy of the central difference scheme is $O(h^2)$, whereas that of the backward difference scheme is only $O(h)$.

In the preceding example, the solution in the outer region (see § 2) is approximately 0. In order to investigate the method in the case that the solution in the outer region is nonconstant, we consider the following example:

$$(5.3) \qquad -\varepsilon \frac{d^2\phi}{dx^2} + u\frac{d\phi}{dx} = \varepsilon\pi^2 \sin(\pi x) + u\pi \cos(\pi x), \qquad \phi(0) = 0, \quad \phi(1) = 1.$$

The exact solution of (5.3) is given by

$$(5.4) \qquad \phi(x) = \sin(\pi x) + \frac{e^{ux/\varepsilon} - 1}{e^{u/\varepsilon} - 1}.$$

l_∞-errors for the numerical computations for several values of ε are presented in Table 5.2.

TABLE 5.2
l_∞-error of some schemes for various values of ε, equation (5.3), *nonequidistant mesh* (5.1), $u = 1$.

		$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
Central differences	$n = 21$	0.022	0.021	0.021	0.021	0.021	0.021
	$n = 41$	0.005	0.005	0.005	0.005	0.005	0.005
Backward differences	$n = 21$	0.279	0.305	0.309	0.309	0.309	0.309
	$n = 41$	0.140	0.155	0.156	0.156	0.156	0.156
Il'in scheme	$n = 21$	0.212	0.299	0.308	0.309	0.309	0.309
	$n = 41$	0.081	0.149	0.156	0.156	0.156	0.156

The results for the central difference scheme are nearly the same as in the first example. The backward difference scheme gives an error that is about 2 times larger than in the homogeneous case, however, with the same order of accuracy ($O(h)$). The accuracy of the Il'in scheme decreases considerably with respect to example 1, and in fact for $\varepsilon < 10^{-3}$. Il'in and the backward difference scheme give the same results.

These examples show that when a normal boundary layer is present a central difference scheme with mesh refinement is preferable to upwind differencing. Only in the case that the outer solution is constant does the Il'in scheme appear to be very accurate.

From § 4 we may conclude that there is one exception in which the central difference scheme is inaccurate, that is, when there is no normal boundary layer and Dirichlet boundary conditions are prescribed on the outflow boundary. In general in that case no mesh refinement will be applied and hence the $O(h^4/\varepsilon)$ error may become important. This problem can be solved by the introduction of some artificial viscosity of $O(h^2)$ (compare with Theorem 4.1).

In the next section it is investigated whether it is possible to combine the advantages of central difference schemes and diagonally dominant matrices by means of a defect correction method.

6. The defect correction method.

The defect correction method has been the subject of many investigations [7], [8], [30]. The method may be used to improve the accuracy of a difference scheme by using a higher order difference scheme, or to estimate the error of such a scheme. Moreover, it may be used to approximate the solution of a higher order scheme even when the higher order scheme is unstable [14].

In this section we study the effect of the defect correction process applied to the upwind schemes using a central difference scheme as correction. We must solve equation (4.3), which will formally be denoted by

$$L\phi = f. \tag{6.1}$$

This problem can be solved by an upwind scheme denoted by

$$L_h\phi_h = f_h, \tag{6.2}$$

or alternatively by a central difference scheme,

$$L'_h\phi'_h = f'_h. \tag{6.3}$$

We define the following defect correction process:

(i) Solve $L_h\phi_h = f_h$.

(ii) Correct ϕ_n with the aid of (6.3) in order to get a new approximation $\bar{\phi}_h$:

$$\bar{\phi}_h = \phi_h - L_h^{-1}[L'_h\phi_h - f'_h]. \tag{6.4}$$

The accuracy of (6.4) may be improved by repeated iterations.

The following theorem gives an indication concerning the performance of the defect correction process as an iteration method.

THEOREM 6.1. *Let $L\phi = f$ be the one-dimensional convection diffusion equation* (4.3). *Let $L'_h\phi'_h = f'_h$ be the central difference discretization with constant mesh size, and let $L_h\phi_h = f_h$ be the upwind difference discretization* (4.4). *Then the iterated defect correction method* (6.4) *converges with a rate of convergence of approximately*

$$\frac{|\tilde{\varepsilon} - \varepsilon|}{\tilde{\varepsilon}}.$$

Theorem 6.1 is proved in Appendix 2.

Remark. When $\varepsilon/\tilde{\varepsilon} \ll 1$ the theorem states that the rate of convergence is approximately 1. Hence the defect correction method is not useful as an iteration method for $\varepsilon \ll uh$.

Although according to Theorem 6.1 defect correction is useless as an iterative method due to its slow convergence, the following theorem shows that defect correction applied once gives at least the same order of accuracy as the upwind schemes. Theorem 6.2 only applies to the case where no boundary layer is present, but this case is intuitively similar to the situation of a boundary layer and a suitably graded mesh. This intuitive idea is confirmed by numerical experience.

THEOREM 6.2. *Let ϕ be the solution of* (4.3). *Let ϕ be ε-independent and $\phi \in C^6(0, 1)$. Let $L'_h\phi'_h = f'_h$ be the central difference discretization and let $L_h\phi_h = f_h$ be the upwind discretization* (4.4), *both with constant mesh size h. Let $\bar{\phi}_h$ be the result of the defect correction method* (6.4). *Assume $\varepsilon \ll \tilde{\varepsilon}$.*

Then the error $\phi - \bar{\phi}_h$ satisfies:

(i) *If $\tilde{\varepsilon} = O(h)$ and the matrix corresponding to L_h is diagonally dominant then*

$$\phi - \bar{\phi}_h = O(h^2)$$

except for a few nodal points near $x = 1$ where

$$\phi = \bar{\phi}_h = O(h).$$

If $\tilde{\varepsilon} \approx hu/2$ then the error $O(h)$ only appears in the grid point next to $x = 1$.

(ii) *If $\tilde{\varepsilon} = O(h^2)$ then*

$$\|\phi - \bar{\phi}\|_2 = O(h^2).$$

For a proof of Theorem 6.2 see Appendix 3.

In Tables 6.1 and 6.2 results of some computations with the defect correction process are given in the case of a smooth solution.

TABLE 6.1

Accuracy of defect correction method and upwind scheme for solving the convection-diffusion equation (4.2), $\varepsilon = 10^{-10}$, $u = 1$, $\phi(x) = \sin(\pi x)$, *equidistant mesh;* ϕ_h *solution of upwind scheme* (4.4); $\bar{\phi}_h$ *solution of defect correction process* (6.4).

		$\max\lvert\phi - \phi_h\rvert$	$\max\lvert\phi - \bar{\phi}_h\rvert$ except for last point	$\phi - \bar{\phi}_h$ last point
$\tilde{\varepsilon} = hu$	$h = 0.1$	.5	.2	.2
	$h = 0.05$	.29	.096	.14
$\tilde{\varepsilon} = hu/2$	$h = 0.1$	.31	.033	.16
	$h = 0.05$	.16	.009	.08

Table 6.1 shows that when $\tilde{\varepsilon} = hu/2$, the accuracy of the defect correction process is $O(h^2)$ except for the last point, where the accuracy is of order h. Furthermore, the error in the last point is one half of the error of the upwind scheme. The results of the scheme with $\tilde{\varepsilon} = hu/2$ are considerably better than those of $\tilde{\varepsilon} = hu$.

Remark. Only two values of h are given in the table; for other values of h the same behavior was observed.

Table 6.2 shows that indeed the error of the defect correction method based on the modified upwind scheme is of order h^2; however, the accuracy of the defect correction method is considerably better than the accuracy of the modified upwind scheme.

TABLE 6.2
Accuracy of defect correction method and modified upwind scheme for solving the convection-diffusion equation (4.2), $\varepsilon = 10^{-10}$, $u = 1$, $\phi(x) = \sin(\pi x)$, *equidistant mesh.* ϕ_h *solution of modified upwind scheme* (4.4); $\bar{\phi}_h$ *solution of defect correction process* (6.4).

	$\max\|\phi - \phi_h\|$	$\max\|\phi - \bar{\phi}_h\|$
$h = .1$	.1	.03
$h = .05$	.0285	.0084

In order to investigate the performance of the defect correction method in the presence of boundary layers, the two examples of § 5 have been computed by a defect correction method using the backward difference scheme as predictor. Results of these computations are presented in Tables 6.3 and 6.4. Comparison of these tables with Tables 5.1 and 5.2 shows that, in the case of a constant solution in the outer region, the defect correction is as accurate as the central difference scheme, whereas in the case of the nonconstant solution in the outer region, the central difference scheme is about 2 times more accurate. Furthermore, due to the mesh refinement in the boundary layer, the defect correction method has an $O(h^2)$ accuracy.

TABLE 6.3
l_∞*-error of the defect correction method for solving the convection diffusion equation* (3.1) *for various values of* ε, *nonequidistant mesh* (5.1), $u = 1$. *Predictor: backward differences.*

	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
$n = 21$	.016	.017	.018	.018	.018	.018
$n = 41$	.005	.005	.005	.005	.005	.005

TABLE 6.4
l_∞*-error of the defect correction method for solving the convection diffusion equation* (5.3) *for various values of* ε, *nonequidistant mesh* (5.1), $u = 1$. *Predictor: backward differences.*

	$\varepsilon = 10^{-2}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
$n = 21$	.048	.051	.052	.052	.052	.052
$n = 41$	.012	.013	.013	.013	.013	.013

7. Two-dimensional problems. In two dimensions, two types of boundary layers occur, normal boundary layers and parallel boundary layers, as discussed in § 2. In order to study the performance of the various methods treated in the preceding sections, we consider the following four examples.

(i) Equation (2.1) with Dirichlet boundary conditions and q such that the exact solution is given by

$$\phi_1 = \frac{1}{\sqrt{x-x_0}}\{e^{-y^2/4\varepsilon(x-x_0)} + e^{-(1-y)^2 4\varepsilon(x-x_0)}\}, \qquad x_0 = -1. \tag{7.1}$$

This solution exhibits the parallel boundary layers common in flow problems. (See Fig. 7.1.) Notice that the "outer solution" as defined in § 2 is approximately zero.

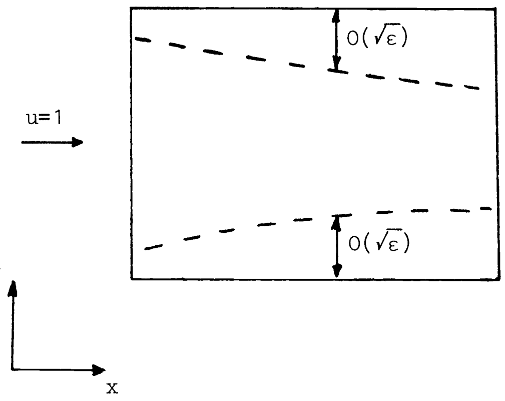


FIG. 7.1. *Parallel boundary layers.*

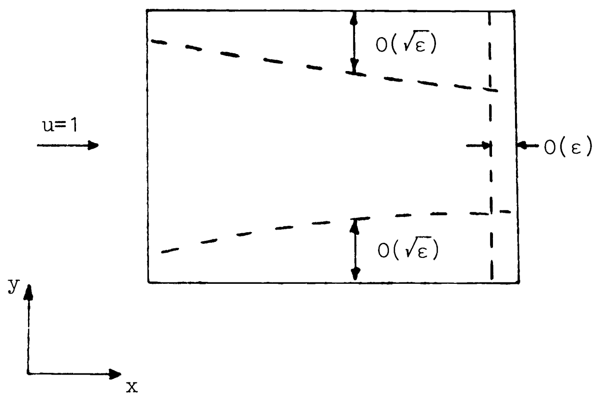


FIG. 7.2. *Parallel and normal boundary layers.*

(ii) Equation (2.1) with Dirichlet boundary conditions and q such that the exact solution is given by

$$\phi_2 = \phi_1 + \frac{e^{-1/\varepsilon} - e^{-(1-x)/\varepsilon}}{1-e^{-1/\varepsilon}}. \tag{7.2}$$

This solution contains both parallel boundary layers and the normal boundary layer of § 3.

(iii) Equation (2.1) with Dirichlet boundary conditions at $x = 0$, $y = 0$ and $y = 1$, Neumann boundary conditions at $x = 1$, and q such that the exact solution is given by

$$\phi_3 = \phi_1. \tag{7.3}$$

(iv) Equation (2.1) with Dirichlet boundary conditions at $x=0$, $y=0$ and $y=1$, Neumann boundary conditions at $x=1$, and q such that the exact solution is given by

$$\phi_4 = \sin(\pi x)\sin(\pi y). \tag{7.4}$$

Equation (7.3) is introduced in order to consider the effect of Neumann boundary conditions at $x=1$, a type of boundary condition that frequently occurs in flow problems.

In (7.4) the effect of a smooth solution is considered. For the effect of a nonconstant "outer solution," we refer to §§ 5 and 6.

All four problems have been solved by a central difference scheme, backward differences and a defect correction method consisting of a backward difference prediction and a central difference correction. The resulting systems of linear equations have been solved by Gaussian elimination. Note that upwinding only takes place in the x-direction and hence does not affect the parallel boundary layer.

(i) *Parallel boundary layer, Dirichlet boundary conditions.* In order to represent the boundary layer adequately the mesh in y-direction was divided into 3 parts:

$$P_1: [0, 8\sqrt{\varepsilon}], \quad P_2: [8\sqrt{\varepsilon}, 1-8\sqrt{\varepsilon}], \quad P_3: [1-8\sqrt{\varepsilon}, 1].$$

In each part an equidistant mesh size was chosen with P_1 and P_2 divided into n meshes and P_2 into 10 meshes. In the x-direction a constant mesh size consisting of 10 meshes was chosen.

In Table 7.1 results of the computations for $n=10$ and $n=20$ and various values of ε are given.

TABLE 7.1
l_∞-error for $(2n+10)\times 10$ grid.

		$\varepsilon=10^{-3}$	$\varepsilon=10^{-4}$	$\varepsilon=10^{-5}$	$\varepsilon=10^{-6}$	$\varepsilon=10^{-7}$
Central differences	$n=10$	.0134	.0205	.0303	.0955	.3009
	$n=20$	.0037	.0062	.0218	.0847	.2891
Upwind differences	$n=10$	.0090	.0090	.0090	.0090	.0090
	$n=20$	.0054	.0054	.0054	.0054	.0054
Defect correction	$n=10$	.0093	.0094	.0094	.0094	.0094
	$n=20$	.0040	.0041	.0041	.0041	.0041

This table shows that the upwind scheme and the defect correction method give comparable results, with a slightly better accuracy for the defect correction. The error is mainly caused by the error in the y-direction. For the central difference scheme the $O(h^4/\varepsilon)$ error in the x-direction becomes important for $\varepsilon < 10^{-3}$. This is due to the Dirichlet boundary condition at $x=1$. (Remember, $h=0.1$ in the x-direction.)

(ii) *Parallel boundary layer in the y-direction and normal boundary layer in the x-direction. Dirichlet boundary conditions.* The mesh in the y-direction was chosen as in Example (i). In order to obtain a good mesh in the x-direction, the interval [0, 1] was divided into 2 parts:

$$Q_1: [0, 1-8\varepsilon], \qquad Q_2: [1-8\varepsilon, 1].$$

In each part an equidistant mesh size has been chosen with Q_1 divided into 10 and Q_2 into n meshes.

In Table 7.2 results of the computations for $n = 10$ and $n = 20$ and various values of ε are given. The table shows that for this example the central difference scheme and the defect correction method appear to be the most accurate. Furthermore, due to the mesh refinement in the normal boundary layer, the error of the defect correction method is about $O(h^\alpha)$, $\frac{3}{2} \leqq \alpha \leqq 2$ instead of $O(h)$ as in Theorem 6.2. Hence the defect correction method is significantly better than the upwind scheme.

TABLE 7.2
l_∞-error for Example (ii), $(2n+10) \times (n+10)$ *grid.*

		$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
Central differences	$n = 10$	.024	.024	.024	.024	.024
	$n = 20$	.006	.006	.006	.006	.006
Upwind differences	$n = 10$	.112	.112	.112	.112	.112
	$n = 20$	.068	.068	.068	.068	.068
Defect correction	$n = 10$	.019	.019	.019	.019	.019
	$n = 20$	.006	.006	.006	.006	.006

(iii) *Parallel boundary layer in the y-direction, Neumann boundary conditions at* $x = 1$. The same mesh as in Example (i) has been used. Results are given in Table 7.3.

TABLE 7.3
l_∞-error for Example (iii), $(2n+10) \times 10$ *grid.*

		$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
Central differences	$n = 10$	.0052	.0052	.0052	.0052	.0052
	$n = 20$	.0015	.0015	.0015	.0015	.0015
Upwind differences	$n = 10$	.0110	.0110	.0110	.0110	.0110
	$n = 20$	.0048	.0048	.0048	.0048	.0048
Defect correction	$n = 10$	.0056	.0056	.0056	.0056	.0056
	$n = 20$	.0026	.0026	.0026	.0026	.0026

This example shows that when a Neumann boundary condition is given at the end of the flow, which is in many cases a natural thing to do, the accuracy of the central difference and the defect correction method is better than the accuracy of the upwind difference method.

(iv) *Smooth solution, Neumann boundary conditions at* $x = 1$. In both the x- and y-directions, $n + 1$ equidistant mesh points were chosen. Results are given in Table 7.4. Table 7.4 shows that the central difference scheme is far more accurate ($O(h^2)$) than the upwind scheme and the defect correction method. Since the largest errors of the upwind scheme appears in the neighborhood of the boundary $x = 1$ (compare with § 4), a better accuracy can be expected when mesh refinement in the x-direction near this boundary is applied. This is certainly the case for the defect correction method (see § 6).

TABLE 7.4
l_∞-error of the computations of Example (iv), ($n \times n$) *net.*

		$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-4}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-6}$	$\varepsilon = 10^{-7}$
Central difference	$n = 10$	.017	.017	.017	.017	.017
	$n = 20$	.004	.004	.004	.004	.004
Upwind difference	$n = 10$	.306	.309	.309	.309	.309
	$n = 20$	.154	.156	.156	.156	.156
Defect correction	$n = 10$	.080	.081	.081	.081	.081
	$n = 20$	.039	.040	.040	.040	.040

Our final conclusion is that for the general case, where boundary layers are present, the "outer solution" is not constant, a Neumann condition is given at the outflow boundary and mesh refinement is used in the horizontal boundary, central differencing is much more accurate than upwind differencing, whereas defect correction results in a method that is less accurate than the central difference method but more accurate than the upwind difference method. The accuracy of the defect correction method can be improved considerably by using a mesh refinement near the outflow boundary even when the solution is smooth. If the "outer solution" is constant, the upwind difference scheme also appears to be accurate when mesh refinement in the parallel boundary layers is applied. In that case, defect correction does not need mesh refinement near the outflow boundary. Furthermore, for all three methods, work and accuracy are uniform in ε if the grading of the mesh is ε-dependent in a suitable way.

The defect correction method shares with the upwind difference method the advantage that the matrix of the system to be solved is diagonally dominant. In the next section the computational cost and accuracy of the direct and iterative methods are studied.

8. Some experiments with Gaussian elimination and iterative methods for the solution of the discretized equations. One can show (see Appendix 1) that the condition of the matrix corresponding to the central difference scheme is of order $(1/\varepsilon)$. Hence for small values of ε this may cause problems. In order to investigate this matter we computed numerical solutions for the two-dimensional convection-diffusion equation (2.1), with Dirichlet boundary conditions and q chosen such that the exact solution is given by:

$$\phi(x, y) = \sin(\pi x)\sin(\pi y). \tag{8.1}$$

Remark. The absence of a boundary layer does not influence the condition of the difference matrix.

Equation (2.1) was solved by a central difference scheme with constant mesh size in both directions ($h = \Delta x = \Delta y$). One can prove [28] that the error of the central difference scheme in this specially chosen example (see also the remarks in § 4), is of order h^2 independently of the value of ε. In Table 8.1 results of Gaussian elimination are given for various values of ε and h. The computations were performed on an IBM 370/158 computer with an accuracy of 16 decimal places.

The table shows that for $\varepsilon \geqq 10^{-8}$ the condition of the matrix does not influence the results, whereas for $\varepsilon \leqq 10^{-9}$ it does. Moreover, when h decreases the condition

TABLE 8.1
l_∞-error of the solution of (2.1), (8.1) *with a central difference scheme.*

$h = \Delta x = \Delta y$	$\varepsilon = 10^{-7}$	$\varepsilon = 10^{-8}$	$\varepsilon = 10^{-9}$	$\varepsilon = 10^{-10}$	$\varepsilon = 10^{-11}$	$\varepsilon = 10^{-12}$
$h = .1$	$.84_{10}\,{}^{3}$	$.84_{10}\,{}^{3}$	$.80_{10}\,{}^{3}$	$.24_{10}\,{}^{2}$	$.44$	18_{10^1}
$h = .05$	$.71_{10}\,{}^{4}$	$.71_{10}\,{}^{4}$	$.63_{10}\,{}^{4}$	$.89_{10}\,{}^{4}$	$.12_{10}\,{}^{1}$	$.79_{10}\,{}^{1}$

number of the matrix decreases, since then the matrix becomes "more diagonally dominant." In fact, many practical problems will have $\varepsilon > 10^{-8}$.

In Table 8.2 results of a special iterative method (a version of a nonsymmetric conjugate gradient method (IDR) developed by Sonneveld [32]) and in Table 8.3 results for a preconditioned variant of this method (PIDR) (see [32]) have been given for various values of ε and h.

TABLE 8.2
Number of iterations to solve (2.1) *by two iterative methods.*

	h	$\varepsilon = 1$	$\varepsilon = 0.1$	$\varepsilon = 0.01$	$\varepsilon = 0.001$
IDR	0.1	14	24	56	-
	0.05	32	47	78	-
PIDR	0.1	8	8	7	-
	0.05	15	16	10	86

central difference scheme ($h = \Delta x = \Delta y$); required accuracy: IDR 10^{-3}, PIDR 10^{-4}

The number of unknowns is equal to 81 for $h = 0.1$ and to 261 for $h = 0.05$. So, from a practical point of view, a number of iterations of 56 for $h = 0.1$ means no convergence. The condition for diagonal dominancy is $\Delta x \leqq 2\varepsilon/u$, so Table 8.2 indicates that the IDR method converges as long as the matrix is diagonally dominant, whereas for the PIDR method, this condition may be violated a little.

In Table 8.3 the number of iterations of the PIDR method has been given for various values of h and ε. This table shows that the PIDR method converges rapidly as long as $\varepsilon \geqq 0.04\ \Delta x$; for smaller values of ε no convergence can be guaranteed.

TABLE 8.3
Number of iterations to solve (2.1) *by the* PIDR *method and central differences.*

	$\varepsilon = 0.1h$	$\varepsilon = 0.08h$	$\varepsilon = 0.06h$	$\varepsilon = 0.04h$	$\varepsilon = 0.02h$	$\varepsilon = 0.01h$
$h = 0.1$	8	8	11	15	31	-
$h = 0.05$	10	11	14	26	133	-

required accuracy: 10^{-4}; $h = \Delta x = \Delta y$

These examples show that in order to use iterative methods it is necessary to use either upwind schemes or defect correction methods. Since the defect correction methods are generally more accurate (see § 7) they should be preferred. Furthermore, since the PIDR method allows for a little violation of the diagonal dominancy condition, and since the defect correction method becomes more accurate when $\tilde{\varepsilon} \to O(h^2)$ one may expect good results of a combination of these methods.

Remark. In many practical problems $0.04h = O(h^2)$.

Note that for (1.1) in more than one dimension, advanced iterative methods, such as preconditioned conjugate gradient methods (e.g., PIDR) or multigrid methods are much cheaper than Gaussian elimination [32]. Recently, a new overrelaxation method has been constructed that is able to solve the central difference scheme iteratively [31].

9. Conclusions. In this paper the convection-diffusion equation has been investigated for small values of the viscosity coefficient. This equation may be considered as a simple model for various more complicated equations such as the Navier–Stokes equations at high Reynolds number. In problems of this type two kinds of boundary layers appear: parallel and normal boundary layers.

The accuracy and computational cost of three kinds of numerical schemes have been studied: central differences, a defect correction method, and upwind differences such as the Il'in scheme.

In more than one dimension the central difference method is more expensive than the other two. The reason is that for small ε the matrix of the system that has to be solved is not diagonally dominant, whereas for the other two it is. This precludes for small ε the use of fast iterative methods for the central difference method. In more than one dimension these are appreciably cheaper than Gaussian elimination for large problems.

In the one-dimensional case with homogeneous right-hand side, the Il'in scheme is the most accurate. In more general one-dimensional problems the central difference scheme is the most accurate, provided it is combined with a suitable ε-dependent mesh refinement in the boundary layer.

In the general two-dimensional case with suitably ε-dependent mesh refinement in the boundary layers, the central difference scheme is the most accurate, except when there is no normal boundary layer and Dirichlet boundary conditions are prescribed at the outflow boundary. The defect correction method based on one-step upwind differences, corrected with one-step central differences, is always more accurate than the upwind difference method; however, in order to get an $O(h^2)$ accuracy in the case of a smooth "outer solution" with Neumann boundary conditions at the outflow boundary, it is necessary to use mesh refinement in the neighborhood of that boundary. A good alternative in that case may be the introduction of an artificial viscosity of $O(h^2)$ instead of $O(h)$, and to use defect correction based on such a scheme. In the special case where the "outer solution" is constant, the accuracy of the upwind difference method is better, but still about a factor of 2 worse than for the other methods.

Appendix 1. Proof of Theorem 4.1. Let $\varepsilon \ll 1$. In order to estimate the error of scheme (4.4) it is necessary to have an estimate of $\|L_h^{-1}\|_2$. $\|L_h^{-1}\|_2 = 1/\sigma_1$ with σ_1^2 the smallest eigenvalue of $L_h^T L_h$, hence

$$\sigma_1^2 = \min_x \frac{x^T L_h^T L_h x}{x^T x} \quad \text{or} \quad \sigma_1 = \min_x \frac{\|L_h x\|_2}{\|x\|_2}.$$

The following lemmas and proofs are due to J. van Kan (private communication). With $y \geqq 0$ we mean $y_i \geqq 0$, $i = 1, 2, \cdots, n$.

LEMMA 1. *Let A be a diagonally dominant matrix with positive diagonal and negative offdiagonals. Then the minimum value of $\|Ax\|_2/\|x\|_2$ will be reached for a vector x_0 satisfying $x_0 \geqq 0$ and $Ax_0 \geqq 0$.*

Proof. Since A is diagonally dominant with positive diagonal and negative offdiagonals it satisfies the following maximum principle:

$$Aw = g \text{ with } g \geqq 0 \text{ implies } w \geqq 0.$$

Let $Ax = f$ and $Ay = |f|$, $(|f|_i = |f_i|)$. From $|f| \geqq 0$ we deduce $y \geqq 0$.

$$A(y-x) = |f| - f \geqq 0, \quad \text{hence} \quad y - x \geqq 0,$$

$$A(y+x) = |f| + f \geqq 0, \quad \text{hence} \quad y + x \geqq 0,$$

$$(y-x, y+x) = \|y\|_2^2 - \|x\|_2^2 \geqq 0.$$

Therefore

$$\frac{\|Ay\|_2}{\|y\|_2} \leqq \frac{\|Ax\|_2}{\|x\|_2}, \qquad \text{Q.E.D.}$$

LEMMA 2. *Let E be the $n \times n$ matrix*

$$\begin{bmatrix} 0 & & & \\ 1 & 0 & & 0 \\ & 1 & & \\ 0 & & 1 & 0 \end{bmatrix},$$

and let $A = -\alpha E + I - \beta E^T$ with I the identity matrix. Let $\alpha + \beta = 1$, $\alpha, \beta \geqq 0$ and $\alpha > \beta$. Then

$$\|A^{-1}\|_2 \leqq \frac{1}{\alpha - \beta} O(n).$$

Proof. We have to estimate $\sigma = \min \|Ax\|_2/\|x\|_2$. According to Lemma 1, $\sigma = \|Ax_0\|_2/\|x_0\|_2$ for some x_0 with $Ax_0 \geqq 0$ and $x_0 \geqq 0$. Since $E^T E = \operatorname{diag}(1, 1, \cdots, 1, 0)$,

$$(\alpha - \beta E^T)(I - E)x_0 = (\alpha I - \beta E^T - \alpha E + \beta E^T E)x_0 \leqq Ax_0.$$

Since $\|E^T y\|_2 \leqq \|y\|_2$, for all y,

$$\|(\alpha I - \beta E^T)y\|_2 \geqq |\alpha\|y\| - \beta\|E^T y\|_2| \geqq (\alpha - \beta)\|y\|_2 \quad \forall y.$$

Hence $\|Ax_0\|_2 \geqq (\alpha - \beta)\|(I - E)x_0\|_2$. The matrix $(I - E^T)(I - E) = I - E - E^T + E^T E$ has eigenvalues

$$\lambda_k = 2 - 2\cos\frac{2k+1}{2n+1}, \qquad k = 1, 2, \cdots, n.$$

Hence $\|(I-E)x_0\|_2 \geqq \sqrt{\lambda_1}\|x_0\|_2 = O(n^{-1})\|x_0\|_2$. Q.E.D.

LEMMA 3. *Let A be the $n \times n$ matrix*

$$A = -\alpha E + I - \beta E^T$$

with $\alpha + \beta = 1$ and E and I as in Lemma 2. Then

$$\|A^{-1}\|_2 \leqq O(n^2).$$

Proof. $J[u]=\|u\|_2$ is a convex functional, hence $J[u]-J[v]\geqq J'[v](u-v)$, *so*

$$\|Ax\|_2-\|x\|_2=\|\alpha(x-Ex)+\beta(x-E^Tx)\|_2-\|x\|_2\geqq-\frac{(x,\alpha Ex+\beta E^Tx)}{\|x\|_2}.$$

Moreover, $(x, Ex)=(Ex, x)=(x, E^Tx)$, and

$$(x, \alpha Ex+\beta E^Tx)=(x, Ex)=\tfrac{1}{2}(x, (E+E^T)x).$$

The matrix $\frac{1}{2}(E+E^T)$ has eigenvalues $\lambda_k=\cos(k\pi/(n+1))$, $k=1, 2, \cdots, n$; hence

$$\tfrac{1}{2}(x, (E+E^T)x)\leqq\cos\left(\frac{\pi}{n+1}\right)\|x\|_2^2=(1-O(n^{-2}))\|x\|_2^2.$$

Therefore

$$\frac{\|Ax\|_2}{\|x\|_2}\geqq O(n^{-2}).$$ Q.E.D.

We now proceed to prove Theorem 4.1. Application of Lemmas 2 and 3 to the matrix L_h gives the following results:

$$L_h=\frac{2\tilde{\varepsilon}}{h^2}\{-\alpha E+I-\beta E^T\}$$

with

$$\alpha=\frac{h^2}{2\tilde{\varepsilon}}\left(\frac{\tilde{\varepsilon}}{h^2}+\frac{u}{2h}\right),\qquad \beta=\frac{h^2}{2\tilde{\varepsilon}}\left(\frac{\tilde{\varepsilon}}{h^2}-\frac{u}{2h}\right).$$

(i) From Lemma 3, $\|L_h^{-1}\|_2\leqq O(1/\tilde{\varepsilon})$.

(ii) When the matrix L_h is diagonally dominant, and $\tilde{\varepsilon}=O(h)$,

$$\alpha-\beta=\frac{h^2}{2\tilde{\varepsilon}}\frac{u}{h}=\frac{hu}{2\tilde{\varepsilon}}=O(1).$$

From Lemma 2 we obtain $\|L_h^{-1}\|_2=O(1)$.

Next consider the truncation error e_T:

(1) $$e_T=L_h\phi-f_h$$

with f_h the vector with components f_i. Then

(2) $$(e_T)_k=(\varepsilon-\tilde{\varepsilon})\frac{d^2\phi}{dx^2}(x_k)+h^2\left\{-\frac{\tilde{\varepsilon}}{12}\frac{d^4\phi}{dx^4}(x_k)+\frac{u}{6}\frac{d^3\phi}{dx^3}(x_k)\right\}+O(h^4).$$

Since $L_h\phi_h=f_h$ and $L_h\phi=f_h+e_T$, we have $L_h(\phi-\phi_h)=e_T$. Hence $\phi-\phi_h=L_h^{-1}e_T$ and

$$(\phi-\phi_h)_0=(\phi-\phi_h)_{n+1}=0.$$

We distinguish three possibilities:

(a) Equation (4.4) is a classical upwind scheme; hence

$$\|L_h^{-1}\|_2=O(1),\qquad \tilde{\varepsilon}=O(h).$$

We suppose that $\varepsilon<\tilde{\varepsilon}$; then

$$\|\phi-\phi_h\|_2\leqq\|L_h^{-1}\|_2\|e_T\|_2=O(h).$$

(b) Equation (4.4) is a central difference scheme; hence $\tilde{\varepsilon} = \varepsilon$. When $\varepsilon = O(h)$, then $\|\phi - \phi_h\|_2 = O(h^2)$; otherwise, when $(\varepsilon \leqq O(h^2))$,

(3) $$(e_T)_k = h^2 g_k + O(h^4), \qquad k = 1, 2, \cdots, n,$$

with

$$g_k = -\frac{\varepsilon}{12}\frac{d^2\phi}{dx^4}(x_k) + \frac{u}{6}\frac{d^3\phi}{dx^3}(x_k).$$

Define

$$\psi(x) = \frac{h^2}{6}\frac{d^2\phi}{dx^2} + \frac{h^2\varepsilon}{12u}\frac{d^3\phi}{dx^3};$$

then

$$(L_h\psi + v)_k = -\frac{\varepsilon h^2}{12}\frac{d^4\phi}{dx^4}(x_k) + \frac{uh^2}{6}\frac{d^2\phi}{dx^3}(x_k) + O(h^2\varepsilon^2) + O(h^4), \quad k = 1, 2, \cdots, n,$$

with v the vector with components v_k satisfying

$$v_k = 0, \qquad k = 2, 3, \cdots, n-1,$$

$$v_1 = \psi(x_0)\left(-\frac{\varepsilon}{h^2} - \frac{u}{2h}\right), \qquad v_n = \psi(x_{n+1})\left(-\frac{\varepsilon}{h^2} + \frac{u}{2h}\right).$$

Hence:

$$L_h(\phi - \phi_h) = L_h\psi + v + O(h^4),$$

$$\|\phi - \phi_h\|_2 \|\psi\| + \|L_h^{-1}v\| + \|L_h^{-1}\| O(h^4) = O(h^2) + O\left(\frac{h^4}{\varepsilon}\right) + \|L_h^{-1}v\|.$$

To estimate the term $\|L_h^{-1}v\|$ we consider $w = L_h^{-1}v$. One easily verifies that the exact solution is given by $w_i = Ar^i + B$, with

$$A = -\frac{\psi_0 - \psi_{n+1}}{r^{n+1} - 1}, \qquad B = -\frac{\psi_{n+1} - \psi_0 r^{n+1}}{r^{n+1} - 1},$$

$$r = \frac{\alpha + 1}{\alpha - 1}, \qquad \alpha = \frac{2\varepsilon}{uh} \ll 1.$$

Hence,

$$r = -(1 + 2\alpha) + O(\varepsilon^2), \qquad r^i = (-1)^i(1 + 2i\alpha) + O(\varepsilon^2).$$

For n even we have $r^{n+1} = -1 + O(\varepsilon)$, and therefore $w = O(h^2)$. For n odd we have

$$r^{n+1} = 1 + \frac{4(n+1)}{uh}\varepsilon + O(\varepsilon^2) = 1 + \frac{4\varepsilon}{uh^2} + O(\varepsilon^2),$$

and therefore $w = O(h^4/\varepsilon)$. Since the worst case may appear, we conclude

$$\|\phi - \phi\|_2 = O(h^2) + O\left(\frac{h^4}{\varepsilon}\right).$$

(c) Equation (4.4) is a modified upwind scheme ($\tilde{\varepsilon} = h^2$). In the same way as for a central difference scheme we can prove that

$$\|\phi - \phi_h\|_2 = O(h^2). \qquad \text{Q.E.D.}$$

Appendix 2. Proof of Theorem 6.1. The iterated defect correction process is defined by

(1) $$\phi_h^i = \phi_h^{i-1} - L_h^{-1}[L_h'\phi_h^{i-1} - f_h'], \qquad i = 1, 2, \cdots,$$

with ϕ_h^0 defined by

(2) $$L_h\phi_h^0 = f_h.$$

From $L_h'\phi_h' = f_h'$ we obtain:

(3) $$\phi_h^i - \phi_h' = (I - L_h^{-1}L_h')(\phi_h^{i-1} - \phi_h').$$

Hence we have to analyse the spectral radius $\rho(I - L_h^{-1}L_h')$.

Let E be defined as in Appendix 1, Lemma 2. Then

(4) $$L_h = \frac{1}{h^2}\left\{-\left(\tilde{\varepsilon} + \frac{uh}{2}\right)E + 2\tilde{\varepsilon}I - \left(\varepsilon - \frac{uh}{2}\right)E^T\right\}$$

and

(5) $$L_h' = \frac{1}{h^2}\left\{-\left(\varepsilon + \frac{uh}{2}\right)E + 2\varepsilon I - \left(\varepsilon - \frac{uh}{2}\right)E^T\right\}.$$

If λ is an eigenvalue, then

(6) $$(I - L_h^{-1}L_h')\mathbf{u} = \lambda\mathbf{u},$$

or $(L_h(1-\lambda) - L_h')\mathbf{u} = \mathbf{0}$, i.e.,

(7) $$\{(-\Delta\varepsilon + \lambda\delta_1)E + (2\Delta\varepsilon - 2\lambda\tilde{\varepsilon})I + (-\Delta\varepsilon + \lambda\delta_2)E^T\}\mathbf{u} = 0$$

with $\Delta\varepsilon = \tilde{\varepsilon} - \varepsilon$, $\delta_1 = \tilde{\varepsilon} + uh/2$, $\delta_2 = \tilde{\varepsilon} - uh/2$. Substituting $u_k = r^k$ gives

(8) $$r_{1,2} = \frac{-\Delta\varepsilon + \lambda\tilde{\varepsilon} \pm \sqrt{\mu}}{-\Delta\varepsilon + \lambda\delta_2}$$

with $\mu = (\Delta\varepsilon - \lambda\tilde{\varepsilon})^2 - (-\Delta\varepsilon + \lambda\delta_1)(-\Delta\varepsilon + \lambda\delta_2) = \lambda^2(uh/2)^2$.

The general solution of (7) is given by

$$u_k = a_1r_1^k + a_2r_2^k, \qquad u_0 = 0, \qquad u_{n+1} = 0.$$

In order for us to have a nontrivial solution, it is necessary that

(9) $$r_1 = r_2\, e^{2\pi ik(n+1)}, \qquad k = 1, 2, \cdots, n.$$

From (8) and (9) we obtain

(10) $$(-\Delta\varepsilon + \lambda\tilde{\varepsilon})(1 - e^{2\pi ik/(n+1)}) = \lambda\frac{uh}{2}(1 + e^{2\pi ik/(n+1)}).$$

Hence

$$(-\Delta\varepsilon + \lambda\tilde{\varepsilon})\left(-i\tan\left(\frac{\pi k}{n+1}\right)\right) = \lambda\frac{uh}{2},$$

$$\lambda_k = \frac{\Delta\varepsilon}{\tilde{\varepsilon} - \alpha_k}, \qquad \alpha_k = i\frac{uh}{2\tan\left(\frac{\pi k}{n+1}\right)},$$

$$|\lambda_k| = \frac{\frac{|\Delta\varepsilon|}{\tilde{\varepsilon}}}{1+\beta_k^2}\sqrt{1+\beta_k^2},$$

with

$$\beta_k = \frac{uh}{2\tilde{\varepsilon}\tan\left(\frac{\pi k}{n+1}\right)}, \qquad k=1,2,\cdots,n.$$

So

$$|\lambda_k| = \frac{|\Delta\varepsilon|}{\tilde{\varepsilon}\sqrt{1+\beta_k^2}} \leqq \frac{|\Delta\varepsilon|}{\tilde{\varepsilon}} < 1.$$

The minimum value of β_k will be small due to the term $\tan(\pi k/n+1)$, hence

$$|\lambda_{\max}| \approx \frac{|\Delta\varepsilon|}{\tilde{\varepsilon}}. \qquad \text{Q.E.D.}$$

Appendix 3. Proof of Theorem 6.2. We shall prove Theorem 6.2 only for the case that L_h corresponds to a classical upwind scheme, i.e., L_h is diagonally dominant and $\tilde{\varepsilon} = O(h)$. The other part of the theorem can be proved by similar arguments. Let

(1) $$L_h\phi_h = f_h$$

be the upwind scheme, and

(2) $$L'_h\phi'_h = f'_h$$

be the central difference scheme. The defect correction scheme is given by

(3) $$\bar{\phi}_h = \phi_h - L_h^{-1}[L'_h\phi_h - f'_h].$$

From (3) we obtain

(4) $$\bar{\phi}_h - \phi = L_h^{-1}[L_h - L'_h](\phi_h - \phi) - L_h^{-1}[L'_h\phi - f'_h].$$

We consider the terms on the right-hand side separately.

(i) $L_h^{-1}[L'_h\phi - f'_h]$. According to Appendix 1 (1), (2), (3), $L'_h\phi - f'_h = O(h^2)$ and $\|L_h^{-1}\|_2 = O(1)$. Hence $\|L_h^{-1}[L'_h\phi - f'_h]\|_2 = O(h^2)$.

(ii) $L_h^{-1}[L_h - L'_h](\phi_h - \phi)$. The truncation error $e_T = L_h\phi - f_h$ is given by $(e_T)_k = g_k + O(h^4)$ with

$$g_k = (\varepsilon - \tilde{\varepsilon})\frac{d^2\phi}{dx^2}(x_k) + h^2\left\{-\frac{\tilde{\varepsilon}}{12}\frac{d^4\phi}{dx^4}(x_k) + \frac{u}{6}\frac{d^3\phi}{dx^3}(x_k)\right\}.$$

Define

$$\psi = \frac{\varepsilon - \tilde{\varepsilon}}{u}\frac{d\phi}{dx} + \left\{\frac{h^2}{6} + \frac{\tilde{\varepsilon}(\varepsilon - \tilde{\varepsilon})}{u^2}\right\}\frac{d^2\phi}{dx^2} - \frac{h^2\tilde{\varepsilon}}{12u}\frac{d^3\phi}{dx^3}.$$

Then $(L_h\psi + v)_k = g_k + O(h^3)$ with

$$v_k = 0, \qquad k = 2, 3, \cdots, n-1,$$

$$v_1 = -\psi_0\left(\frac{\tilde{\varepsilon}}{h^2} + \frac{u}{2h}\right), \qquad v_n = -\psi_{n+1}\left(\frac{\tilde{\varepsilon}}{h^2} - \frac{u}{2h}\right).$$

Hence $\phi - \phi_h = \psi + L_h^{-1}v + O(h^3)$. Since

$$\|L_h^{-1}\|_2 = O(1), \quad \|L_h - L'_h\|_2 = O(h^{-1}), \quad (L_h - L'_h)\psi = \omega_1 + \omega_2,$$

with

$$(\omega_1)_k = \frac{\tilde{\varepsilon}-\varepsilon}{h^2}(-\psi_{k-1}+2\psi_k-\psi_{k+1}) = (\tilde{\varepsilon}-\varepsilon)\frac{d^2\psi}{dx^2}(x_k)+O(h^2), \qquad k=1,2,\cdots,n,$$

$$(\omega_2)_k = 0, \qquad k=2,3,\cdots,n-1,$$

$$(\omega_2)_0 = \frac{\tilde{\varepsilon}-\varepsilon}{h^2}\psi_0, \qquad (\omega_2)_n = \frac{\tilde{\varepsilon}-\varepsilon}{h^2}\psi_{n+1},$$

we have

$$L_h^{-1}[L_h-L_h'](\phi_h-\phi) = L_h^{-1}[L_h-L_h']L_h^{-1}v + L_h^{-1}\omega_2 + O(h^2).$$

One easily verifies that $(L_h^{-1}v)_k = Ar^k+B$, with

$$A = \frac{\psi_0-\psi_{n+1}}{1-r^{n+1}}, \quad B = \frac{\psi_{n+1}-\psi_0 r^{n+1}}{1-r^{n+1}}, \quad r = \frac{\alpha+1}{\alpha-1}, \quad \alpha = \frac{2\tilde{\varepsilon}}{uh} > 1.$$

Hence $[L_h-L_h']L_h^{-1}v = q-\omega_2$ with $q_k = \zeta r^k$, $k=1,2,\cdots,n$, and

$$\zeta = \frac{2(\tilde{\varepsilon}-\varepsilon)}{h^2}\frac{2A}{\alpha^2-1} < \frac{2A\alpha}{\alpha^2-1}\frac{u}{h} = \frac{2\alpha}{\alpha^2-1}\frac{u}{h}\frac{\psi_0-\psi_{n+1}}{1-r^{n+1}}.$$

Finally:

$$\bar{\phi}_h-\phi = L_h^{-1}q+O(h^2) = \frac{\zeta}{u}\left[\left(-kh+\frac{r^{n+1}}{r^{n+1}-1}\right)r^k+\frac{r^{n+1}}{1-r^{n+1}}\right]+O(h^2).$$

Without difficulty one can show that in the range of interest $1 \leqq \alpha \leqq 3$ we have:

$$(L_h^{-1}q)_k \leqq \tfrac{1}{2}(\psi_0-\psi_{n+1}).$$

Furthermore, $(L_h^{-1}q)_k$ is small except for the last few points before $k=n+1$. When $\alpha \to 1$,

$$(L_h^{-1}q)_k \to 0, \qquad k=1,2,\cdots,n-1, \qquad (L_h^{-1}q)_n \to \tfrac{1}{2}(\psi_0-\psi_{n+1}).$$

Since $\psi = O(h)$, the theorem has been proved. The case $\tilde{\varepsilon} = O(h^2)$ can be proved in exactly the same way. Q.E.D.

REFERENCES

[1] O. Axelsson and I. Gustafsson, *A modified upwind scheme for convective transport equations, and the use of a conjugate gradient method for the solution of nonsymmetric systems of equations*, J. Inst. Math. Appl., 23 (1979), pp. 321–337.

[2] J. C. Chien, *A general finite difference formulation with application to Navier-Stokes equations*, Computers and Fluids, 5 (1977), pp. 15–31.

[3] I. Christie, D. F. Griffiths, A. R. Mitchell and O. C. Zienkiewicz, *Finite element methods for second order equations with significant first derivatives*, Int. J. Num. Meth. Engng, 10 (1976), pp. 1389–1396.

[4] W. Eckhaus, *Matched Asymptotic Expansions and Singular Perturbations*, North-Holland, Amsterdam, 1973.

[5] W. Eckhaus and E. M. de Jager, *Asymptotic solutions of singular perturbation problems for linear differential equations of elliptic type*, Arch. Rational Mech. Anal., 23 (1966), pp. 26–86.

[6] C. Ehlig, *Comparison of numerical methods for solution of the diffusion-convection equation in one and two dimensions*, in [12], pp. 1.91–1.102.

[7] R. Frank, *The method of iterated defect-correction and its application to two-point boundary value problems, Part* I, Numer. Math., 25 (1976), pp. 409–419; *Part* II, Numer. Math., 27 (1977), pp. 407–420.

[8] R. Frank and C. W. Ueberhuber, *Iterated defect correction for differential equations. Part* I: *theoretical results*, Computing, 20 (1978), pp. 207–228.

[9] R. H. Gallagher et al. eds., *Finite Elements in Fluids*, vol. 3, John Wiley, London, 1978.

[10] D. K. Gartling, *Some comments on the paper by Heinrich, Huyakorn, Zienkiewicz and Michell.* Int. J. Num. Meth. Engng, 12 (1978), pp. 187–190.

[11] M. Th. van Genuchten, *On the accuracy and efficiency of several numerical schemes for solving the convective-dispersion equation*, in [12], pp. 1.71–1.90.

[12] W. G. Gray and G. F. Pinder, eds., *Finite Elements in Water Resources*, First conference, Princeton Univ., July 1976, Pentech Press, London, 1976.

[13] D. F. Griffiths and J. J. Lorenz, *An analysis of the Petrov-Galerkin finite element method applied to a model problem*, Research Paper 334, Dept. Mathematics and Statistics, Univ. Calgary, Alberta, Canada, February 1977.

[14] W. Hackbusch, *Bemerkungen zur iterierten Defectkorrectur und zu ihrer kombination mit Mehrgitterverfahren*, Report 79-13, Universität zu Köln, Mathematisches Institut, September 1979.

[15] J. C. Heinrich, P. S. Huyakorn, O. C. Zienkiewicz and A. R. Mitchell, *An upwind finite element scheme for two-dimensional convective transport equations*, Int. J. Num. Meth. Engng, 11 (1977), pp. 131–143.

[16] J. C. Heinrich and O. C. Zienkiewicz, *Quadratic finite element schemes for two-dimensional convective transport problems*, Int. J. Num. Meth. Engng, 11 (1977), pp. 1831–1844.

[17] P. W. Hemker, *A numerical study of stiff two-point boundary problems*, Thesis, Mathematisch Centrum, Amsterdam, 1977.

[18] A. M. Il'in, *Differencing scheme for a differential equation with a small parameter affecting the highest derivative*, Math. Notes Acad. Sc. USSR, 6 (1969), pp. 596–602.

[19] D. C. L. Lam, *Comparison of finite element and finite difference methods for nearshore advection-diffusion transport models*, in [12], pp. 1.115–1.130.

[20] P. Melli, *An application of the Galerkin method to the Eulerian–Lagrangian treatment of time dependent advection and diffusion of air pollutants*, in [12], pp. 1.59–1.70.

[21] J. W. Mercer and C. R. Faust, *The application of finite element techniques to immiscible flow in porous media*, in [12], pp. 1.21–1.58.

[22] A. R. Mitchell and R. Wait, *The Finite Element Method in Partial Differential Equations*, John Wiley, Chichester, 1977.

[23] C. E. Pearson, *On a differential equation of boundary layer type*, J. Math. Phys., 47 (1968), pp. 134–154.

[24] ———, *On nonlinear ordinary differential equations of boundary layer type*, J. Math. Phys., 47 (1968), pp. 351–358.

[25] P. J. Roache, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1972.

[26] A. A. Samarskii, *Monotonic difference schemes for elliptic and parabolic equations in the case of a non-selfadjoint elliptic operator*, Ž. Vyčisl. Mat. i Mat. Fiz., 5 (1965), pp. 548–581. (In Russian.)

[27] A. Segal, *On the need for upwind differencing for elliptic singular perturbation problems, Part* 1, Report NA-27, Technical University Delft, 1980.

[28] ———, *On the need for upwind differencing for elliptic singular perturbation problems, Part* 2, Report NA-35, Technical University Delft, 1980.

[29] I. A. Smith, *Integration in time of diffusion and diffusion-convection equations*, in [12], pp. 1.3–1.20.

[30] H. J. Stetter, *The defect correction principle and discretization methods*, Numer. Math., 29 (1978), pp. 425–443.

[31] J. Strikwerda, *Iterative methods for the numerical solution of second order elliptic equations with large first order terms*, this Journal, 1 (1980), pp. 119–130.

[32] P. Wesseling and P. Sonneveld, *Numerical experiments with a multiple grid and a preconditioned Lanczos type method*, Approximation Methods for Navier–Stokes Equations, R. Rautmann, ed., Lecture notes in Mathematics 771, Springer, Berlin, 1980, pp. 543–562.

[33] O. C. Zienkiewicz and J. C. Heinrich, *The finite element method and convection problems in fluid mechanics*, in [9], pp. 1–22.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

A VECTORIZABLE VARIANT OF SOME ICCG METHODS*

HENK A. VAN DER VORST†

Abstract. The preconditioned conjugate gradient method can be a useful tool in solving certain very large sparse linear systems. If this is done on a vector machine like the CRAY-1, then it appears that some of the most effective preconditionings are difficult to vectorize. In this paper it is shown how a class of preconditionings can be modified in such a way that they become highly vectorizable while still easy to program. Numerical experiments that show the improvement in performance of the preconditioned conjugate gradient algorithm have been included.

Key words. preconditioned conjugate gradients, ICCG methods, incomplete Choleski, vector/parallel computers

0. Introduction. The ICCG methods arise when the conjugate gradient algorithm (CG) is applied to the preconditioned system $K^{-1}Ax = K^{-1}b$ in order to solve the linear system $Ax = b$, where K is an incomplete Choleski (IC) decomposition of the n by n symmetric positive real matrix A [6], [9], [10]. Since most of the CG algorithm is vectorizable (assuming that fast code is available for inner products) and the matrix vector product Ax is also vectorizable, the main difficulty arises in the computation of $K^{-1}y$ for a given vector y on a vector processor.

This is due to the fact that in the IC decomposition we have $K = LL^T$, where L is lower triangular, and computation of $L^{-1}z$ as well as $L^{-T}z$ requires in many relevant cases recurrence relations in the components of z and these are not vectorizable directly. This has been recognized by many authors who propose techniques to overcome this problem. Significant improvement in efficiency can be made if one uses cyclic reduction techniques [3], [5], [7], [11]. Using these techniques rather fast code can be written for vector/parallel processors such as the CRAY-1, but this requires a permutation of the unknowns and has often to be done in assembler code in order to achieve optimal performance. Dubois, Greenbaum and Rodrigue [1] propose to approximate the inverse of A by a truncated Neumann expansion and to use this as a preconditioning. Although their preconditioning is completely vectorizable, they report a significant increase in the number of *cg* iterations as compared to the ICCG methods. Johnson and Paul [4] claim that the number of iterations can be reduced to about the same level as for the ICCG methods, by introducing parameters that depend on extreme eigenvalues of the matrix A.

In this paper we return to the incomplete Choleski decomposition and describe a useful variant that can be vectorized completely very easily. The variant arises if one applies similar techniques as proposed in [1] in an intermediate step of the preconditioning process. We will show that for this variant the number of iterations will be about the same as for the standard ICCG algorithms. No information is required about eigenvalues of A and neither are other parameters introduced. The resulting preconditioned CG method has been tested on a CRAY-1 computer and numerical examples indicating the efficiency of the new variants have been included.

* Received by the editors September 9, 1981, and in revised form December 17, 1981. The research reported in this paper was supported in part by the European Research Office, London, through grant DAJA 37-80-C-0243.

† Academisch Computer Centrum Utrecht, Budapestlaan 6, Utrecht, the Netherlands.

1. The vectorizable ICCG algorithm. The ICCG algorithm is defined by

$$\alpha_i=\frac{(r_i,K^{-1}r_i)}{(p_i,Ap_i)},\quad x_{i+1}=x_i+\alpha_ip_i,\quad r_{i+1}=r_i-\alpha_iAp_i,$$

(1.1)

$$\beta_i=\frac{(r_{i+1},K^{-1}r_{i+1})}{(r_i,K^{-1}r_i)},\qquad p_{i+1}=K^{-1}r_{i+1}+\beta_ip_i.$$

The preconditioning matrix K is an incomplete Choleski factorization of the matrix A and will be written as $K=(L+D)D^{-1}(D+L)^T$. L is a strict lower triangular matrix and D is a diagonal matrix.

We will now assume that A has been scaled in such a way that $D=I$, which simplifies most of the coming formulas and makes actual computation more efficient. In most situations the computation of Ap_i is completely vectorizable and we therefore focus attention to the computation of $K^{-1}r_i$, which causes the bottleneck on vector computers since it requires back substitution. In our further analysis we will restrict ourselves to matrices A that come from 5-point finite difference approximations of elliptic p.d.e.'s over rectangular regions. From the presentation it will be clear for which other matrices the ideas presented in this paper can be applied.

In our case the matrix A has the block structure shown in Fig. 1.

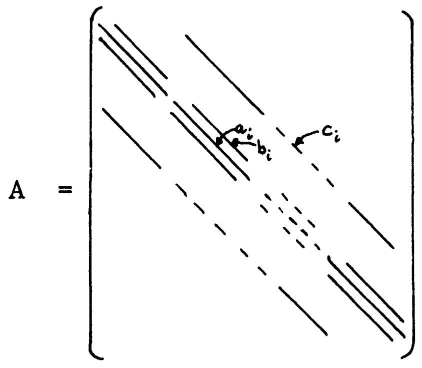


FIG. 1

We now consider for K the ICCG(1, 1) variant [10], which means that the factor L of K has the same sparsity structure as the corresponding part of A. It follows that the elements of L are identical to those of A.

We write the decomposition as

$$A=K+R=(-F-E+I)(I-E-F)^T+R,$$

where E^T is the matrix consisting of only the upper diagonal elements b_i and F^T contains the upper diagonal elements c_i. The matrix R represents the approximation error of K with respect to A.

A typical step in the determination of $K^{-1}r_i$ is the solution of z from

$$(I-E-F)z=y$$

or

$$(I-E_j)z_j=y_j+F_jz_{j-1},$$

where the index j refers to the jth block of the matrix and where z and y have been partitioned accordingly.

Since the elements b_i of E_j in general are small compared to 1.0, the idea is now to compute z_j from

$$z_j=(I-E_j)^{-1}(y_j+F_jz_{j-1})=(I+E_j+E_j^2+E_j^3+\cdots)(y_j+F_jz_{j-1}),$$

and to truncate the power series after some term, the mth term say. We observe that the approximate solution

$$(I+E_j+\cdots+E_j^m)(y_j+F_jz_{j-1}) \tag{1.2}$$

has now been written in fully vectorizable form. Our main concern is to investigate whether a value of m exists such that the computation of (1.2) will not be too expensive and also that this truncation will not lead to a much higher number of iterations in the resulting ICCG algorithm.

2. The effect of truncation on the preconditioning matrix.

We now try to determine where to truncate the power series for $(I-E)^{-1}$ in such a way that the error due to this truncation is small compared to the approximation error $R=A-K$. We use the relation

$$(I+E+\cdots+E^m)^{-1}=(I-E)(I-E^{m+1})^{-1}, \tag{2.1}$$

and recall that

$$A=(I-E-F)(I-E-F)^T+R=K+R. \tag{2.2}$$

If we approximate the factor $(I-E)^{-1}$ in the back substitution by $I+E+E^2+\cdots+E^m$, then we have effectively

$$A=\tilde{K}+S+R, \tag{2.3}$$

where $\tilde{K}$ is the matrix which describes the truncated back substitution and S is the error matrix due to the truncation.

It follows from (2.1) that

$$\tilde{K}=((I-E)(I-E^{m+1})^{-1}-F)((I-E)(I-E^{m+1})^{-1}-F)^T. \tag{2.4}$$

The matrix S_1 is defined by

$$\begin{aligned}(I-E)(I-E^{m+1})^{-1}&=(I-E)(I+E^{m+1}+E^{2(m+1)}+\cdots)\\&=I-E+(I-E)E^{m+1}(I-E^{m+1})^{-1}=I-E+S_1.\end{aligned} \tag{2.5}$$

If (2.5) is inserted in (2.4), then it follows from (2.3) that

$$S=-S_1(I-E-F)^T-(I-E-F)S_1^T-S_1S_1^T. \tag{2.6}$$

The elements of E are just the b_i and those of F are given by c_i. Therefore we can derive bounds on S in the following way:

$$\|I-E-F\|_\infty \leqq 1+b+c, \tag{2.7}$$

where $b=\max b_i$, $c=\max c_i$. Thus

$$\|S_1\|_\infty \leqq (1+b)b^{m+1}(1-b^{m+1})^{-1}\simeq b^{m+1}(1+b). \tag{2.8}$$

If we neglect also the higher order term $S_1S_1^T$, then from (2.6), (2.7) and (2.8) we have

$$\|S\|_\infty \leqq 2(1+b)(1+b+c)b^{m+1}. \tag{2.9}$$

The matrix R has the structure in Fig. 2, where $r_i = b_i c_{i-1}$. Thus

$$\|R\|_\infty = 2 \max |b_i c_{i-1}|. \tag{2.10}$$

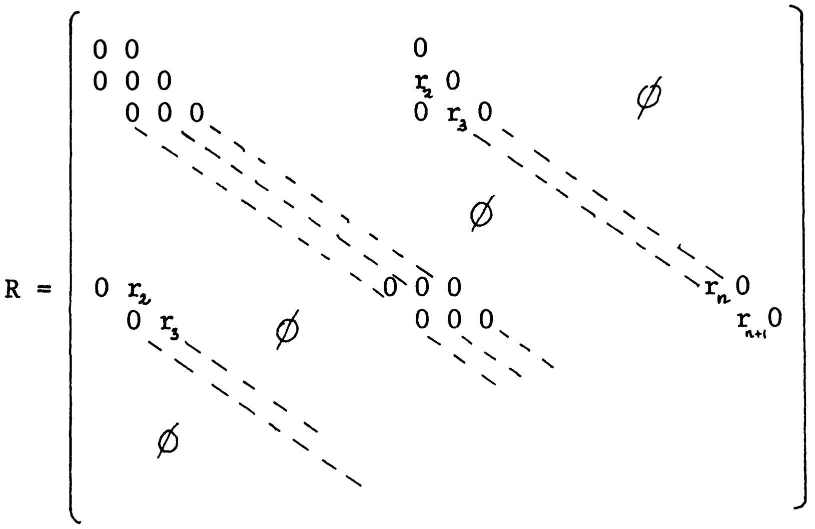


FIG. 2

If we choose as a model problem the set of equations that arises from 5-point finite difference approximation to the Poisson equation, with Dirichlet boundary conditions, then it follows that the elements of the matrices E and F are approximately $b_i \simeq 0.3$ and $c_i \simeq 0.3$. In this case we have $\|R\|_\infty \simeq 0.18$ and for $\|S\|_\infty$ we find the upper bounds in Table 1, depending on the level of truncation.

TABLE 1
Upper bounds for $\|S\|_\infty$.

m	1	2	3
$\|S\|_\infty$	0.4	0.11	0.034

We see from this table that for $m = 2$ the truncation error S is comparable to the approximation error R, while for $m = 3$ the approximation error R dominates.

Since $(I+E+E^2+E^3)y$ can be computed as $(I+E^2)(I+E)y$ which can be computed almost at the cost of $(I+E+E^2)y$, we prefer truncation after 3 terms over truncation after 2 terms.

3. Numerical results. In this section we give the numbers of iterations for both the "standard" ICCG(1, 1) algorithm and the truncated ICCG(1, 1) algorithm, as well as the CPU times required for both algorithms on a CRAY-1 computer. The truncation idea can also be applied similarly to the more complicated ICCG algorithms like ICCG(1, 2) and ICCG(1, 3) [10]. In the latter variant 2 extra nonzero diagonals are allowed in the factors of K, situated near the F-diagonals. Since the approximation error matrix R for the ICCG(1, 3) decomposition is in general much smaller than for the ICCG(1, 1) case, it may be expected that in this case the number of iterations will be increased when we apply truncation. The results for this decomposition have also been included. All the algorithms have been coded in standard Fortran except for the inner products which were available in assembler code. In the truncated ICCG algorithms the vector $(I-E)^{-1}y$ has been approximated by $(I+E^2)(I+E)y$, where E^2 was computed once and stored in memory. Note that E^2 is a matrix that has only one nonzero diagonal. All the matrices occurring in the algorithms have been represented in diagonal form, this seems to be the most effective way on a vector/parallel processor [8]. It should be stressed here that the CPU times listed for the standard ICCG algorithm may give the reader a too optimistic impression with respect to their behavior on vector machines. Their CPU times are relatively low due to the fact that A has only 5 nonzero diagonals in our examples, the decompositions have been scaled such that their main diagonal elements are equal to 1.0, Eisenstat's efficient implementation has been applied [2], and they have been vectorized also as far as possible (Eisenstat's ideas have not been applied to the new variants). Further on the size of the matrices is comparatively small so that any initial scalar arithmetic influences the results.

The computer time required for the construction of the incomplete decompositions as well as for the scaling of the matrix has been included in the listed CPU times.

In order to be better able to judge the results for the vectorizable variants we have also included the results for the vectorizable truncated Neumann expansion preconditioning as proposed in [1]. The value of p in this case refers to the number of terms after which the Neumann expansion has been truncated.

Although more complicated problems, e.g., 9-point finite difference operators or finite element problems, would give a higher improvement ratio for the vectorizable algorithms, we have chosen simple examples in order to facilitate for the reader the reconstruction of these problems for testing purposes.

In both the examples 20 different starting vectors, with entries chosen at random in $(0, 1)$, have been taken and the average results over these 20 iteration processes have been listed. The iteration was terminated as soon as the residual vector in 2-norm was less than 10^{-10}. For the 5-point finite difference discretisation of the p.d.d.'s see [13].

Example 1. $-(Au'_x)'_x-(Au'_y)'_y=B$. This equation was solved over the unit square (see Fig. 3), where A and B are given by

region 1: $A=1, \quad B=0,$

region 2: $A=100, \quad B=100.$

On the boundary defined by $y=0$ we have $u=0$, along the other three boundaries we have the Neumann condition $\partial u/\partial n=0$. A uniform mesh was chosen with mesh spacing 1/32, resulting in a linear system with 1056 unknowns. Table 2 shows the iteration results.

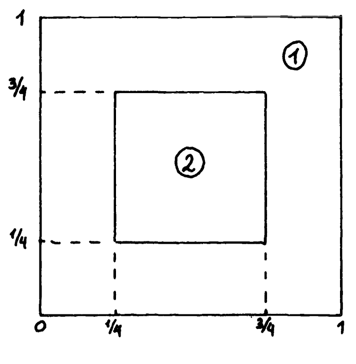


FIG. 3

TABLE 2
Iteration results for Example 1.

algorithm	number of iterations	CPU time on CRAY-1
standard ICCG(1, 1)	62	0.125
truncated ICCG(1, 1)	62	0.078
standard ICCG(1, 3)	31	0.086
truncated ICCG(1, 3)	33	0.051
truncated Neumann, $p=2$	103	0.079
truncated Neumann, $p=4$	73	0.087

Example 2. $-(Au'_x)'_x-(Au'_y)'_y+Bu=B$. (See Fig. 4.) The coefficients in this equation are defined as follows:

region 1: $A=3.0$, $B=0.05$,
region 2: $A=2.0$, $B=0.03$,
region 3: $A=1.0$, $B=0.02$.

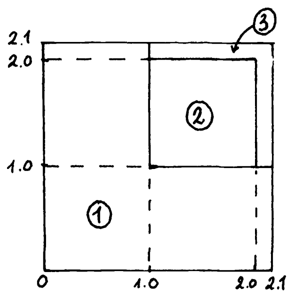


FIG. 4

Along the boundaries the Neumann condition $\partial u/\partial n=0$ is imposed. The grid spacing of the uniform mesh was chosen to be 1/42; thus the resulting linear system is of the order $N=1849$. See Table 3.

TABLE 3
Iteration results for Example 2.

algorithm	number of iterations	CPU-time on CRAY-1
standard ICCG(1, 1)	87	0.304
truncated ICCG(1, 1)	89	0.181
standard ICCG(1, 3)	44	0.206
truncated ICCG(1, 3)	47	0.116
truncated Neumann, $p=2$	149	0.194
truncated Neumann, $p=4$	105	0.215

4. Conclusions. We see that the vectorizable ICCG variants give a considerable improvement in CPU time on the CRAY-1. The variants are also more efficient than the truncated Neumann expansion preconditioning as suggested in [1]. It seems that they are very near to the equivalent incomplete Choleski preconditioning the existence of which has been questioned by Dubois, Greenbaum and Rodrigue [1, see Conclusions]. Since the programming effort is only slightly increased as compared to the standard ICCG algorithms, we believe that it is advisable to prefer the truncated algorithms for computers like the CRAY-1. From numerical experiments it follows that truncation after 2 terms increases the number of iterations by a small percentage. Since the computational effort to compute $(I+E+E^2)y$ is almost the same as the effort for $(I+E+E^2+E^3)y$, truncation after 3 terms pays off most. This truncation idea can be easily applied on more complicated ICCG algorithms and their nonsymmetric equivalents [12]. As soon as there are more nonzero diagonals near the main diagonal then the computer time required for evaluation of the truncated power series (applied on a given vector) increases accordingly, making this idea possibly less attractive in those cases. It is an open question whether and to what extent the introduction of parameters in the truncated power series similar to those as described by Johnson and Paul [4], could improve the performance of the truncated ICCG methods.

Acknowledgments. Discussions with Thomas Jordan (Los Alamos), Gene Golub and Rob Schreiber (both at Stanford) proved to be quite helpful and stimulating. Both referees draw my attention to several useful references for which I am very much indebted. They also contributed to the presentation of this paper.

REFERENCES

[1] P. F. DUBOIS, A. GREENBAUM AND G. H. RODRIGUE, *Approximating the inverse of a matrix for use in iterative algorithms on vector processors*, Computing, 22 (1979), pp. 257–268.

[2] S. C. EISENSTAT, *Efficient implementation of a class of preconditioned conjugate gradient methods*, this Journal, 2 (1981), pp. 1–4.

[3] A. GREENBAUM AND G. RODRIGUE, *The incomplete Choleski conjugate gradient method for the STAR (5-point operator)*, Res. Rep. UCID-17574, Lawrence Livermore Lab., Livermore, CA, 1977.

[4] O. G. JOHNSON AND G. PAUL, *Optimal parameterized incomplete inverse preconditioning for conjugate gradient calculations*, Res. Rep. RC8644, IBM-Research Center, Yorktown Heights, NY, 1981.

[5] T. L. JORDAN, *A guide to parallel computation and some CRAY-1 experiences*, LANL Report LA-UR-81-247, Los Alamos National Laboratory, Los Alamos, NM, 1981.

[6] D. S. KERSHAW, *The ICCG method for the iterative solution of systems of linear equations*, J. Comp. Phys., 26 (1978), pp. 43–65.

[7] ———, *The solution of single linear tridiagonal systems and vectorization of the ICCG algorithm on the CRAY-1*, Res. Rep., UCID-19085, Lawrence Livermore Lab., Livermore, CA, 1981.

[8] N. K. MADSEN, G. H. RODRIGUE AND J. I. KARUSH, *Matrix multiplication by diagonals on a vector/parallel processor*, Inform. Proc. Letters, 5 (1976), pp. 41–45.

[9] J. A. MEYERINK 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.

[10] ———, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems*, J. Comp. Phys., 44 (1981), pp. 134–155.

[11] G. RODRIGUE AND D. WOLITZER, *Incomplete block cyclic reduction*, 2nd IMACS International Symposium on Parallel Computation, Newark, Delaware, 1981.

[12] H. A. VAN DER VORST, *Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems*, J. Comp. Phys., 44 (1981), pp. 1–19.

[13] R. VARGA, *Matrix Iterative Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1962.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

AN IMPROVED METHOD FOR NUMERICAL INVERSION OF LAPLACE TRANSFORMS*

F. R. DE HOOG†, J. H. KNIGHT† AND A. N. STOKES†

Abstract. An improved procedure for numerical inversion of Laplace transforms is proposed based on accelerating the convergence of the Fourier series obtained from the inversion integral using the trapezoidal rule. When the full complex series is used, at each time-value the epsilon-algorithm computes a (trigonometric) Padé approximation which gives better results than existing acceleration methods. The quotient-difference algorithm is used to compute the coefficients of the corresponding continued fraction, which is evaluated at each time-value, greatly improving efficiency. The convergence of the continued fraction can in turn be accelerated, leading to a further improvement in accuracy.

Key words. Laplace transforms, Fourier series, acceleration, ε-algorithm, continued fractions, quotient-difference algorithm.

1. Introduction.

The Laplace transform of a function $f(t)$, $t \geqq 0$, is defined as

$$F(p) = \int_0^\infty \exp(-pt) f(t)\, dt, \tag{1}$$

and the inverse transform is given by

$$f(t) = \frac{1}{2\pi i} \int_{\gamma - i\infty}^{\gamma + i\infty} \exp(pt) F(p)\, dp, \tag{2}$$

where γ is such that the contour of integration is to the right of any singularities of $F(p)$.

There are many problems for which the Laplace transform of the solution is readily found, but the transform cannot be easily inverted analytically. For such cases a numerical method of inversion must be used. Unfortunately, different numerical methods give the most accurate answers for various classes of functions and there is no "best" method. Recently, Davies and Martin [4] tested a wide variety of methods on a representative set of sixteen transforms with known inverses and concluded that one of the more successful methods was based on accelerating the convergence of a Fourier series.

It therefore seems worthwhile attempting to improve an already quite good method, and this is the aim of the present paper. In the next section, we describe the Fourier series method and the need to accelerate the convergence of the series. Our improved algorithm is given in § 3, and some numerical comparisons are presented in § 4.

2. Fourier series methods.

Since f is a real-valued function for real t, three mathematically equivalent forms can be obtained by manipulating the real and imaginary parts of (2). They are

$$f(t) = \frac{2}{\pi} \exp(\gamma t) \int_0^\infty \operatorname{Re}\{F(p)\} \cos(\omega t)\, d\omega \tag{3}$$

$$= -\frac{2}{\pi} \exp(\gamma t) \int_0^\infty \operatorname{Im}\{F(p)\} \sin(\omega t)\, d\omega \tag{4}$$

$$= \frac{1}{\pi} \exp(\gamma t) \int_0^\infty \operatorname{Re}\{F(p) \exp(i\omega t)\}\, d\omega, \tag{5}$$

* Received by the editors April 21, 1980, and in revised form December 7, 1981.

† Division of Mathematics and Statistics, Commonwealth Scientific and Industrial Research Organization, Canberra, ACT 2601, Australia.

where $p=\gamma+i\omega$. If we now discretize, using the trapezoidal rule with step size π/T, we obtain the approximations

$$f_1(t)=\frac{2}{T}\exp(\gamma t)\left[\frac{F(\gamma)}{2}+\sum_{k=1}^{\infty}\operatorname{Re}\left\{F\left(\gamma+\frac{ik\pi}{T}\right)\right\}\cos\left(\frac{k\pi t}{T}\right)\right], \tag{6}$$

$$f_2(t)=-\frac{2}{T}\exp(\gamma t)\sum_{k=1}^{\infty}\operatorname{Im}\left\{F\left(\gamma+\frac{ik\pi}{T}\right)\right\}\sin\left(\frac{k\pi t}{T}\right), \tag{7}$$

$$f_3(t)=\frac{1}{T}\exp(\gamma t)\left[\frac{F(\gamma)}{2}+\sum_{k=1}^{\infty}\operatorname{Re}\left\{F\left(\gamma+\frac{ik\pi}{T}\right)\exp\left(\frac{ik\pi t}{T}\right)\right\}\right] \tag{8}$$

with various derivations; (6)–(8) are the basis of Fourier series methods examined and used by Dubner and Abate [5], Cooley, Lewis and Welch [2], Silverberg [11], Durbin [6], Crump [3] and others.

Although (3) and (4) are mathematically equivalent, the discretized forms (6)–(8) are not. In fact, it can be shown (see Durbin [6]) that for $0\leqq t\leqq 2T$

$$f_1(t)=f(t)+\sum_{k=1}^{\infty}\exp(-2\gamma kT)[f(2kT+t)+\exp(2\gamma t)f(2kT-t)], \tag{9}$$

$$f_2(t)=f(t)+\sum_{k=1}^{\infty}\exp(-2\gamma kT)[f(2kT+t)-\exp(2\gamma t)f(2kT-t)], \tag{10}$$

$$f_3(t)=f(t)+\sum_{k=1}^{\infty}\exp(-2\gamma kT)f(2kT+t). \tag{11}$$

Thus, if γT is large, the discretization error (incurred when the integral is approximated by the Fourier series) is small for f_1 and f_2 when $0\leqq t\leqq T$ and for f_3 when $0\leqq t\leqq 2T$. Furthermore, (9)–(10) indicate that f_3 (which is the average of f_1 and f_2) may be the most useful approximation in practice because it does not contain the exponentially increasing term $\exp(2\gamma t)$ in the discretization error. However, the choice of the most useful discretization depends heavily on how the sums of the infinite series in (6)–(8) are calculated.

To illustrate this, let us consider the evaluation of $f_3(t)$ on an equally spaced partition $t_j=2jT/J$, $j=0,1,\cdots,J-1$. A formal manipulation of $f_3(t_j)$ yields

$$T\exp(\gamma t_j)f_3(t_j)=\operatorname{Re}\left\{\sum_{k=0}^{J-1}a_k\exp\left(\frac{2\pi ikj}{J}\right)\right\}, \tag{12}$$

where

$$a_0=\frac{F(\gamma)}{2}+\sum_{l=1}^{\infty}F\left(\gamma+\frac{ilJ\pi}{T}\right) \tag{13}$$

and

$$a_k=\sum_{l=0}^{\infty}F\left(\gamma+\frac{i(k+lJ)\pi}{T}\right),\qquad k=1,\cdots,J-1.$$

If J is large and a_k, $k=0,\cdots,J-1$, are known, (12) can be evaluated efficiently using a fast Fourier transform. Unfortunately, the formal manipulation used to derive (12) is not valid unless the series defining f_3 converges uniformly. For example, if $F(p)=1/p$, the sums in (13) will diverge.

On the other hand, if f_1 is evaluated on the grid $s_j = jT/J$, $j = 0, \cdots, J$, we obtain

$$\frac{T}{2}\exp(\gamma s_j) f_1(s_j) = \sum_{k=0}^{J} b_k \cos\left(\frac{\pi k j}{J}\right), \qquad j = 0, \cdots, J \tag{14}$$

where

$$b_0 = \frac{F(\gamma)}{2} + \sum_{l=1}^{\infty} \operatorname{Re}\left\{F\left(\gamma + \frac{2\pi i l J}{T}\right)\right\}$$

for $k = 1, \cdots, J-1$,

$$b_k = \sum_{l=0}^{\infty} \operatorname{Re}\left\{F\left(\gamma + \frac{i(2lJ+k)\pi}{T}\right) + F\left(\gamma + \frac{i(2lJ+J-k)\pi}{T}\right)\right\} \tag{15}$$

and

$$b_J = \sum_{l=0}^{\infty} \operatorname{Re}\left\{F\left(\gamma + \frac{i(2l+1)J\pi}{T}\right)\right\}.$$

Cooley, Lewis and Welch [2] have noted that (14) can also be evaluated using a fast cosine transform and that the sums defining $b_0, \cdots, b_j$ will usually converge. They, therefore, choose (15) and (16) as the basis of their inversion algorithm even though a comparison of (9) and (11) shows that the discretization error of f_3 will usually be smaller than that of f_1. A method for choosing an appropriate value of γ (assuming that the b_k, $k = 0, \cdots, J$ are known to machine precision) is also given by these authors. However, accurate calculation of the b_k is by no means trivial. For example, if $F(p) = 1/p$, then

$$b_0 = \frac{1}{2\gamma} + \gamma T^2 \sum_{l=1}^{\infty} \frac{1}{\gamma^2 T^2 + 4\pi^2 l^2 J^2},$$

which converges very slowly. In general, a very large number of evaluations of $F(p)$ may be required for an accurate inversion, and it is clear that this part of the calculation can be the major source of rounding error and require the most computer time, even when $F(p)$ has a simple form.

Of course, slow convergence, with its attendant problems of loss of accuracy and excessive computation time, can be expected for any algorithm which evaluates (6)–(8) directly. If a Fourier scheme is to be useful, it would seem to be necessary to accelerate the convergence of the appropriate infinite series. Durbin [6] reported that he had tried various convergence acceleration methods, apparently without major improvement. Crump [3] used the epsilon-algorithm of Wynn [12] to accelerate the convergence of the sum in (5) with good results.

To accelerate the convergence of a sequence of partial sums using the epsilon-algorithm

$$S_0 = \frac{F(\gamma)}{2}, \qquad S_k = S_{k-1} + \operatorname{Re}\left\{F\left(\gamma + \frac{ik\pi}{T}\right)\exp\left(\frac{ik\pi t}{T}\right)\right\}, \qquad k = 1, \cdots, 2M,$$

we define $\varepsilon_{-1}^{(m)} = 0$, $\varepsilon_0^{(m)} = S_m$, $m = 0, 1, \cdots, 2M$, and then put

$$\varepsilon_{p+1}^{(m)} = \varepsilon_{p-1}^{(m+1)} + [\varepsilon_p^{(m+1)} - \varepsilon_p^{(m)}]^{-1}. \tag{16}$$

The sequence $\varepsilon_0^{(0)}, \varepsilon_2^{(0)}, \varepsilon_4^{(0)}, \cdots, \varepsilon_{2M}^{(0)}$ gives successive approximations to the sum of the series. Crump's accelerated estimate of $f(t)$ we denote by

$$\hat{f}_1(t, \gamma, T, M) = \frac{1}{T} \exp(\gamma t) \varepsilon_{2M}^{(0)}.$$

For comparison, we denote the unaccelerated estimate by

$$\hat{f}_0(t, \gamma, T, M) = \frac{1}{T} \exp(\gamma t)\left[\frac{F(\gamma)}{2} + \sum_{k=1}^{2M} \operatorname{Re}\left\{F\left(\gamma + \frac{ik\pi}{T}\right) \exp\left(\frac{ik\pi t}{T}\right)\right\}\right].$$

3. Improved acceleration. A convergence acceleration technique can be applied to any arbitrary sequence of numbers, with uncertain results, but most such techniques assume that the sequence has some particular structure, and they work best for sequences close to the assumed form. As emphasized by Wynn [13], applying the epsilon-algorithm to the partial sums of a power series in a variable z is equivalent to constructing successive rational approximations to the power series. These rational approximations have special properties being particular Padé approximations to the power series. (For their properties see Baker [1] or Henrici [8].) The sequence $\{\varepsilon_{2m}^{(0)}\}$, $m = 0, 1, \cdots, M$, gives diagonal elements of the Padé table

$$\varepsilon_{2m}^{(0)} = \left[\sum_{n=0}^{m} b_n z^n\right] \Big/ \left[\sum_{n=0}^{m} c_n z^n\right], \qquad c_0 = 1$$

with

$$\sum_{n=0}^{2m} a_n z^n - \varepsilon_{2m}^{(0)} = O(z^{2m+1}).$$

The Padé approximations are often better than the original power series at representing a function and valid in a larger domain. Accordingly, the greater improvement in convergence can be expected when the algorithm is applied to a sequence which happens to be the partial sums of some power series. For example, given a cosine series $\sum a_n \cos(n\theta)$ to accelerate, Wynn [13] appends the conjugate sine series $\sum a_n i \sin(n\theta)$, applies the epsilon-algorithm to the complex power series in the variable $z = \exp(i\theta)$ and takes the real part of the result. Because the algorithm is nonlinear, this is not the same as accelerating the cosine series directly.

For our problem of Laplace transform inversion, the natural way to obtain a power series is to retain the original complex form

$$g(t) = \frac{F(\gamma)}{2} + \sum_{k=1}^{\infty} F\left(\gamma + \frac{ik\pi}{T}\right) \exp\left(\frac{ik\pi t}{T}\right) = \sum_{k=0}^{\infty} a_k z^k \tag{17}$$

with $a_0 = \frac{1}{2}F(\gamma)$, $a_k = F(\gamma + ik\pi/T)$, $k = 1, 2, \cdots$ and $z = \exp(i\pi t/T)$.

We propose to apply the epsilon-algorithm to the partial sums of the series (17) to give an improved estimate

$$\hat{f}_2(t, \gamma, T, M) = \frac{1}{T} \exp(\gamma t) \operatorname{Re}\{\varepsilon_{2M}^{(0)}\}. \tag{18}$$

If the epsilon table is written out in the form

$$\begin{array}{cccc}
 & \varepsilon_0^{(0)} & & \\
\varepsilon_{-1}^{(1)} & & \varepsilon_1^{(0)} & \\
 & \varepsilon_0^{(1)} & & \varepsilon_2^{(0)} \\
\varepsilon_{-1}^{(2)} & & \varepsilon_1^{(1)} & \\
 & \varepsilon_0^{(2)} & & \\
\varepsilon_{-1}^{(3)} & & &
\end{array}$$

for each J, a new diagonal $\varepsilon_j^{(J-j)}$, $j=-1, 0, \cdots, J$, can be calculated using the "rhombus rule" in the form

$$(19) \qquad \varepsilon_j^{(J-j)} = \varepsilon_{j-2}^{(J+1-j)} + [\varepsilon_{j-1}^{(J+1-j)} - \varepsilon_{j-1}^{(J-j)}]^{-1}, \qquad j = 1, \cdots, J$$

with the initial values

$$\varepsilon_{-1}^{(J+1)} = 0, \qquad \varepsilon_0^{(J)} = \sum_{k=0}^{J} a_k z^k.$$

Every second such diagonal yields an improved estimate $\varepsilon_{2m}^{(0)}$, $m = 1, 2, \cdots, M$.

When applied to the partial sums of a power series in this way, it can be seen that the procedure calculates $\varepsilon_1^{(J-1)} = [a_J z^J]^{-1}$, for example, by subtracting successive partial sums to retrieve the Jth term of the power series. Table entries can be numerically large and the rhombus rule often involves the subtraction of two large quantities to find a relatively small difference. It is not surprising that calculation of table elements is numerically unstable and can be seriously affected by roundoff error for large J. For each new value of z, all the entries in the table must be recalculated, and $M(2M-1)$ entries are needed to give the estimate $\varepsilon_{2M}^{(0)}$ although no more than two diagonals need to be stored at any stage.

Given that our proposed acceleration procedure is in fact computing a diagonal Padé approximation corresponding to the power series (17), it is profitable to investigate other methods which may be more efficient. Wynn [13] suggests that, while it is more economical to use the epsilon-algorithm for an exploratory survey with a limited number of sample values of z, if results are required for many values it is better to use a method such as the quotient-difference algorithm of Rutishauser [10]. This makes the rational approximation available in the form of a continued fraction for substitution of different values of z. Therefore, for the inversion of Laplace transforms, we propose to use the quotient-difference algorithm to calculate a (trigonometric) rational approximation to the series (17) in the form of a continued fraction, and to evaluate it by recursion at any value of the time.

In our outline of the quotient-difference algorithm, we follow Henrici [7], [8], [9]. Given the power series (17), we wish to calculate the corresponding continued fraction

$$v(z) = d_0/(1 + d_1 z/(1 + d_2 z/(1 + \cdots)))$$

with the same formal power series development. Given

$$u(z, M) = \sum_{k=0}^{2M} a_k z^k, \qquad v(z, M) = d_0/(1 + d_1 z/(1 + \cdots + d_{2M} z)),$$

$$u(z, M) - v(z, M) = O(z^{2M+1})$$

so $v(z, M)$ is the same diagonal Padé approximation as $\varepsilon_{2M}^{(0)}$. The coefficients d_k, $k = 0, 1, \cdots, 2M$ can be calculated using the quotient-difference algorithm as follows. We set $e_0^{(i)} = 0$ for $i = 0, \cdots, 2M$ and $q_1^{(i)} = a_{i+1}/a_i$ for $i = 0, \cdots, 2M-1$. Then successive columns of the array are formed according to the rules for

$$(20) \qquad r = 1, \cdots, M, \qquad e_r^{(i)} = q_r^{(i+1)} - q_r^{(i)} + e_{r-1}^{(i+1)}, \qquad i = 0, \cdots, 2M-2r,$$

for

$$r = 2, \cdots, M, \qquad q_r^{(i)} = q_{r-1}^{(i+1)} e_{r-1}^{(i+1)} / e_{r-1}^{(i)}, \qquad i = 0, \cdots, 2M-2r-1.$$

The array is written out in the form

$$\begin{array}{ccccc}
 & q_1^{(0)} & & & \\
e_0^{(1)} & & e_1^{(0)} & & \\
 & q_1^{(1)} & & q_2^{(0)} & \\
e_0^{(2)} & & e_1^{(1)} & & e_2^{(0)}, \\
 & q_1^{(2)} & & q_2^{(1)} & \\
e_0^{(3)} & & e_1^{(2)} & & \\
 & q_1^{(3)} & & & \\
e_0^{(4)} & & & &
\end{array}$$

and it can be seen that successive diagonals can be built up in the same way the epsilon table is constructed using the relations (20).

The continued fraction coefficients d_k, $k = 0, \cdots, 2M$ are given by

$$d_0 = a_0, \quad d_{2m-1} = -q_m^{(0)}, \quad d_{2m} = -e_m^{(0)}, \qquad m = 1, \cdots, M.$$

Calculating these using the $q-d$ algorithm involves about the same effort as calculating the epsilon table but does not have to be redone for a new value of z. For any z the successive convergents of the continued fraction can be evaluated using the recurrence relations

$$(21) \qquad \begin{aligned} A_n &= A_{n-1} + d_n z A_{n-2}, \\ B_n &= B_{n-1} + d_n z B_{n-2}, \end{aligned} \qquad n = 1, \cdots, 2M$$

with the initial values $A_{-1} = 0$, $B_{-1} = 1$, $A_0 = d_0$, $B_0 = 1$, then $v(z, M) = A_{2M}/B_{2M}$.

To apply the procedure to the series (17), as before we put $z = \exp(i\pi t/T)$, $a_0 = \frac{1}{2}F(\gamma)$, $a_k = F(\gamma + ik\pi/T)$, $k = 1, \cdots, 2M$ and the estimate of $f(t)$ is

$$(22) \qquad \hat{f}_3(t, \gamma, T, M) = \frac{1}{T} \exp(\gamma t) \operatorname{Re}\left\{\frac{A_{2M}}{B_{2M}}\right\}.$$

If the numerical calculations were exact, $\hat{f}_2$ and $\hat{f}_3$ would be identical since the underlying rational approximations are the same. Calculating the elements of the $q-d$ array is also numerically unstable, but it is to be expected that roundoff error can affect $\hat{f}_2$ and $\hat{f}_3$ in different ways.

The continued fraction has been obtained as the result of applying an acceleration procedure to the power series (17), and now we can consider applying an acceleration procedure to the continued fraction itself. Writing $v(z)$ as a terminating fraction with remainder,

$$v(z)=d_0(1+d_1z/(1+\cdots+d_nz/(1+r_{n+1})\cdots)),$$

where $r_{n+1}(z)$ is the remainder. In the usual method of evaluation, the nth convergent A_n/B_n is obtained with $r_{n+1}(z)$ taken as zero, but better estimates of the remainder give more accurate approximations for $v(z)$. The simplest improvement is achieved by assuming that $d_{n+m}=d_{n+1}$ for all positive m, which leads to an estimate of the remainder satisfying

$$r_{n+1}=d_{n+1}z/(1+r_{n+1}).$$

However, in many continued fractions, the coefficients form a pattern which repeats in pairs so a better and more general assumption is that for all nonnegative m

$$d_{n+2m}=d_n \quad \text{and} \quad d_{n+2m+1}=d_{n+1}.$$

This leads to an estimate $R_{n+1}(z)$ of the remainder satisfying

$$R_{n+1}=d_{n+1}z/(1+d_nz/(1+R_{n+1}))$$

or

$$R^2_{n+1}+[1+(d_n-d_{n+1})z]R_{n+1}-d_{n+1}z=0.$$

For convergence, we want the root of smaller magnitude which is

$$R_{n+1}(z)=-h_{n+1}[1-(1+d_{n+1}z/h^2_{n+1})^{1/2}],$$

where h_{n+1} denotes $\frac{1}{2}[1+(d_n-d_{n+1})z]$, and the complex square root has argument $\leqq\pi/2$.

This further form of acceleration can be done in conjunction with the iterative method of forward evaluation used above, if on the last evaluation of the recurrence relations $d_{2M}z$ is replaced by $R_{2M}(z)$,

(23) $$R_{2M}(z)=-h_{2M}[1-(1+d_{2M}z/h^2_{2M})^{1/2}], \qquad h_{2M}=\tfrac{1}{2}[1+(d_{2M-1}-d_{2M})z]$$

giving

(24) $$A'_{2M}=A_{2M-1}+R_{2M}A_{2M-2}, \qquad B'_{2M}=B_{2M-1}+R_{2M}B_{2M-2}.$$

A doubly accelerated estimate for $f(t)$ is given by

(25) $$\hat{f}_4(t,\gamma,T,M)=\frac{1}{T}\exp(\gamma t)\operatorname{Re}\left\{\frac{A'_{2M}}{B'_{2M}}\right\}.$$

4. Numerical results. The three main sources of error in approximating $f(t)$ by the sum of a Fourier series instead of by the integral (5) are (i) discretization error, (ii) truncation error, caused by taking only a finite sum of N terms, and (iii) roundoff error. The discretization error is governed by the values of the parameters γ and T and the roundoff error by the properties of the machine used. It is effectively the truncation error that we are trying to improve, so for a given transform we will use identical values of γ and T for all methods and compute in double precision where necessary to prevent significant roundoff error. Of course, for fixed word length, stability to roundoff error is an important attribute on which we will comment later.

Numerical results were obtained on a CDC Cyber 76 computer with about 14 decimal digits in single precision and about 28 in double precision. Two transforms with known inverses were used to evaluate the improvements to the accelerated

TABLE 1
Errors in estimates of $g_1(t)$, $T = 7.5$, $\gamma = -0.5 + 0.4 \ln (10.0)$.

estimate: precision:	$\hat{f}_1(t, \gamma, T, 9)$ double	$\hat{f}_2(t, \gamma, T, 9)$ double	$\hat{f}_3(t, \gamma, T, 9)$ single	$\hat{f}_4(t, \gamma, T, 9)$ single	$\hat{f}_4(t, \gamma, T, 14)$ single
t					
0.0	7.2×10^{-3}	8.2×10^{-3}	8.2×10^{-3}	5.9×10^{-3}	1.5×10^{-3}
0.5	-1.8×10^{-3}	2.9×10^{-5}	2.9×10^{-5}	4.2×10^{-6}	6.8×10^{-7}
1.0	3.8×10^{-4}	1.5×10^{-6}	1.5×10^{-6}	7.8×10^{-7}	6.7×10^{-7}
2.0	4.5×10^{-5}	3.5×10^{-7}	3.5×10^{-7}	3.5×10^{-7}	3.5×10^{-7}
3.0	8.9×10^{-6}	3.1×10^{-8}	3.1×10^{-8}	3.1×10^{-8}	3.1×10^{-8}
4.0	-1.6×10^{-6}	-1.1×10^{-7}	-1.1×10^{-7}	-1.1×10^{-7}	-1.1×10^{-7}
5.0	-6.2×10^{-8}	-9.5×10^{-8}	-9.5×10^{-8}	-9.5×10^{-8}	-9.5×10^{-8}
6.0	-3.6×10^{-8}	-3.5×10^{-8}	-3.5×10^{-8}	-3.5×10^{-8}	-3.5×10^{-8}
7.0	7.0×10^{-9}	7.0×10^{-9}	7.0×10^{-9}	7.0×10^{-9}	7.0×10^{-9}
8.0	1.9×10^{-8}	1.9×10^{-8}	1.9×10^{-8}	1.9×10^{-8}	1.9×10^{-8}
9.0	1.2×10^{-8}	1.2×10^{-8}	1.2×10^{-8}	1.2×10^{-8}	1.2×10^{-8}
10.0	8.8×10^{-9}	2.6×10^{-9}	2.6×10^{-9}	2.6×10^{-9}	2.6×10^{-9}
11.0	-6.5×10^{-9}	-2.4×10^{-9}	-2.4×10^{-9}	-2.4×10^{-9}	-2.4×10^{-9}
12.0	-1.9×10^{-5}	-3.8×10^{-9}	-3.8×10^{-9}	-2.9×10^{-9}	-2.8×10^{-9}
13.0	2.1×10^{-3}	-4.5×10^{-8}	-4.5×10^{-8}	-5.9×10^{-9}	-1.3×10^{-9}
13.5	2.9×10^{-2}	4.5×10^{-7}	4.5×10^{-7}	4.7×10^{-8}	-5.9×10^{-10}
14.0	-1.8×10^{-1}	-8.1×10^{-7}	-8.1×10^{-7}	-2.9×10^{-7}	-3.5×10^{-11}
14.5	-3.0×10^{-1}	-1.1×10^{-3}	-1.1×10^{-3}	-1.4×10^{-4}	-2.7×10^{-8}
15.0	4.0	4.5	4.5	3.3	8.5×10^{-1}

Fourier series method of Crump [3]. The first is $G_1(p) = (p^2 + p + 1)^{-1}$ which is the transform of the damped sinusoid $g_1(t) = (2/\sqrt{3}) \exp(-t/2) \sin(t\sqrt{3}/2)$ used as an example by Dubner and Abate [5] and by Crump. Most numerical methods can be expected to perform well on this transform of a continuous function. The other is $G_2(p) = 1/p$ the transform of the heaviside unit function $g_2(t) = H(t)$ which is zero for negative t, unity for positive t and takes the value $\frac{1}{2}$ for t zero. The jump discontinuity at zero makes this a difficult example for all Fourier series inversion methods. In practice, if the location and magnitude of a discontinuity are known, it can often be treated by analytical methods.

To invert the transform $G_1(p)$, the parameter values of Crump were used, namely $T = 7.5$, $N = 2M + 1 = 19$ and $\gamma = 0.5 + 0.4 \ln (10.0) \sim 0.421$. Double precision-arithmetic was used if the single precision calculations were found to be affected by roundoff. The Crump estimate $\hat{f}_1$ was compared with our improved estimates $\hat{f}_2$ obtained with the complex epsilon-algorithm, $\hat{f}_3$ obtained with the quotient-difference algorithm and $\hat{f}_4$ obtained with accelerated evaluation of the continued fraction. The numerical differences between these and the true values of $g_1(t)$ for the four methods are shown in Table 1 for values of the time up to 15.0. Single precision results are given for $\hat{f}_3$ and $\hat{f}_4$; as to the accuracy shown, the errors were unchanged for these estimates by computing in double precision.

It can be seen that between about 5.0 and 10.0, the truncation error is negligible for all four estimates, the remaining error being due to discretization and not decreased by further increasing M. The convergence of the Fourier series may be affected by a Gibbs phenomenon near $t = 0$ and $t = 2T = 15.0$, and near these points our methods are much more accurate than that of Crump. The single precision estimate $\hat{f}_3$ gives identical errors to the double precision estimate $\hat{f}_2$ showing that the continued fraction procedure is less susceptible to roundoff error as well as computationally much more

TABLE 2
Errors in estimates of $g_2(t)$, $\gamma = 1.0$, $T = 12.0$.

estimate: precision:	$\hat{f}_0(t, \gamma, T, 500)$ single	$\hat{f}_1(t, \gamma, T, 17)$ double	$\hat{f}_2(t, \gamma, T, 17)$ double	$\hat{f}_3(t, \gamma, T, 17)$ single	$\hat{f}_4(t, \gamma, T, 17)$ single
t					
0.00	-1.22×10^{-3}	-3.57×10^{-3}	-6.84×10^{-2}	-6.84×10^{-2}	8.25×10^{-2}
0.05	-2.27×10^{-2}	-3.86×10^{-1}	1.09×10^{-4}	1.09×10^{-4}	5.78×10^{-6}
0.10	-6.94×10^{-3}	2.20×10^{-3}	-7.43×10^{-4}	-7.43×10^{-4}	-4.17×10^{-5}
0.15	-1.82×10^{-5}	4.58×10^{-2}	3.04×10^{-4}	3.04×10^{-4}	1.07×10^{-5}
0.20	3.78×10^{-3}	1.16×10^{-1}	-5.99×10^{-6}	-5.99×10^{-6}	3.97×10^{-7}
0.30	5.47×10^{-3}	-2.73×10^{-3}	-1.60×10^{-6}	-1.60×10^{-6}	-1.01×10^{-7}
0.40	2.08×10^{-3}	5.08×10^{-3}	1.02×10^{-6}	1.02×10^{-6}	2.49×10^{-8}
0.50	-2.22×10^{-3}	9.98×10^{-3}	-2.44×10^{-7}	-2.44×10^{-7}	-2.98×10^{-9}
0.75	3.32×10^{-4}	-1.66×10^{-3}	3.20×10^{-9}	3.20×10^{-9}	1.41×10^{-10}
1.00	1.27×10^{-3}	1.10×10^{-4}	2.36×10^{-10}	2.36×10^{-10}	3.56×10^{-11}
1.50	3.58×10^{-3}	-1.97×10^{-5}	4.16×10^{-11}	4.16×10^{-11}	3.77×10^{-11}
2.00	3.22×10^{-3}	-4.00×10^{-6}	3.78×10^{-11}	3.78×10^{-11}	3.78×10^{-11}
4.00	-5.13×10^{-5}	-2.37×10^{-8}	3.78×10^{-11}	3.78×10^{-11}	3.78×10^{-11}
6.00	-6.40×10^{-2}	1.63×10^{-11}	3.78×10^{-11}	3.78×10^{-11}	3.78×10^{-11}

efficient. The accelerated continued fraction estimate $\hat{f}_4(t, \gamma, T, 9)$ yields a further increase in accuracy near zero and 15.0. Also, $N = 2M + 1 = 29$ terms were used to calculate the estimate $\hat{f}_4(t, \gamma, T, 14)$ and a further improvement was obtained.

A similar comparison was made for the transform $G_2(p)$ of the unit step function using the parameter values $\gamma = 1.0$ and $T = 12.0$ and various values of M. The first column of Table 2 shows the errors in the simple sum $\hat{f}_0(t, \gamma, T, 500)$ using $N = 2M + 1 = 1001$ terms in the sum. The second column shows the errors in the Crump estimate $\hat{f}_1(t, \gamma, T, 17)$ using only $N = 2M + 1 = 35$ terms clearly showing the great improvement resulting from accelerating the convergence. For the same value of M, the errors given by our improved acceleration methods are shown in the remaining columns of Table 2, and it can be seen that for this discontinuous function $g_2(t)$, the improvement over the Crump method is more dramatic. Right at the discontinuity, our methods are no better, but the improvement increases rapidly for increasing t. As in the first example, the doubly accelerated continued fraction estimate $\hat{f}_4$ gives the best results. Calculations (results not shown) were also done with $N = 2M + 1 = 41$ terms. However, owing to the inherent instability of the quotient-difference algorithm, inaccuracies in the single precision calculation of the coefficients d_n became apparent with $M = 20$. These did not greatly affect the final estimate $\hat{f}_3$ but affected the estimates $R_{2M}(z)$ of the remainder, causing $\hat{f}_4$ to be less of an improvement over $\hat{f}_3$ for this value of M.

5. Conclusions. The numerical results in the previous section show that for the accelerated Fourier series methods using the full complex series to calculate a trigonometric rational approximation with either the epsilon or the quotient-difference algorithm results in a significant improvement in accuracy over the method of Crump [3]. Using the $q-d$ algorithm to explicitly calculate the coefficients of the corresponding continued fraction is highly efficient when the inverse is required at many time-values. Evaluating the continued fraction is more stable than applying the epsilon-algorithm to the partial sums and can itself be accelerated to give a further improvement in accuracy.

Acknowledgments. The authors would like to thank Drs. B. Davies and B. L. Martin for a preprint of their survey article and Dr. Davies and Mr. J. E. Drummond for supplying other references.

REFERENCES

[1] G. A. BAKER, *Essentials of Padé Approximants*, Academic Press, New York, 1975.

[2] J. W. COOLEY, P. A. W. LEWIS AND P. D. WELCH, *The fast Fourier transform algorithm: Programming considerations in the calculation of sine, cosine and Laplace transforms*, J. Sound Vib., 12 (1970), pp. 315–337.

[3] K. S. CRUMP, *Numerical inversion of Laplace transforms using a Fourier series approximation*, J. Assoc. Comp. Mach., 23 (1976), pp. 89–96.

[4] B. DAVIES AND B. L. MARTIN, *Numerical inversion of Laplace transforms: A critical evaluation and review of methods*, J. Comp. Phys., 33 (1979), pp. 1–32.

[5] H. DUBNER AND J. ABATE, *Numerical inversion of Laplace transforms by relating them to the finite Fourier cosine transform*, J. Assoc. Comp. Mach., 15 (1968), pp. 115–123.

[6] F. DURBIN, *Numerical inversion of Laplace transforms: An efficient improvement to Dubner and Abate's method*, Comput. J., 17 (1974), pp. 371–376.

[7] P. HENRICI, *Quotient-difference algorithms*, Mathematical Methods for Digital Computers, vol. 2, A. Ralston and H. S. Wilf, eds., John Wiley, New York, pp. 37–62, 1967.

[8] ———, *Applied and Computational Complex Analysis*, vol. 1, John Wiley, New York, 1974.

[9] ———, *Applied and Computational Complex Analysis*, vol. 2, John Wiley, New York, 1977.

[10] H. RUTISHAUSER, *Der Quotienten-Differenzen-Algorithmus*, Birkhauser Verlag, Basel, 1957.

[11] M. SILVERBERG, *Improving the efficiency of Laplace-transform inversion for network analysis*, Electronics Letts., 6 (1970), pp. 105–106.

[12] P. WYNN, *On a device for computing the $e_m(S_n)$ transformation*, MTAC, 10 (1956), pp. 91–96.

[13] ———, *Transformations to accelerate the convergence of Fourier series*, Blanch Anniversary Volume, B. Mond, ed., U.S. Air Force, Washington, DC, 1967, pp. 339–379.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

DIFFERENTIAL/ALGEBRAIC EQUATIONS ARE NOT ODE'S*

LINDA PETZOLD†

Abstract. This paper outlines a number of difficulties which can arise when numerical methods are used to solve systems of differential/algebraic equations of the form $\mathbf{F}(t, \mathbf{y}, \mathbf{y}') = \mathbf{0}$. Problems which can be written in this general form include standard ODE systems as well as problems which are substantially different from standard ODE's. Some of the differential/algebraic systems can be solved using numerical methods which are commonly used for solving stiff systems of ordinary differential equations. Other problems can be solved using codes based on the stiff methods, but only after extensive modifications to the error estimates and other strategies in the code. A further class of problems cannot be solved at all with such codes, because changing the stepsize causes large errors in the solution. We describe in detail the causes of these difficulties and indicate solutions in some cases.

Key words. ordinary differential equations, singular differential systems, stiff, numerical methods, error estimates, backward differentiation formulas

1. Introduction. A number of difficulties can arise when numerical methods are used to solve systems of differential/algebraic equations (DAE) of the form $\mathbf{F}(t, \mathbf{y}, \mathbf{y}') = \mathbf{0}$. These problems look much like standard ordinary differential equation (ODE) systems of the form $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$ (and of course include these systems as a special case), and many of the DAE systems can be solved using numerical methods which are commonly used for solving stiff systems of ODE's. However, the class of DAE systems also includes problems with properties that are very different from those of standard ODE's. Some of these problems cannot be solved using variable-stepsize stiff methods such as backward differentiation formulas (BDF). Others can be solved using such methods but only after substantial modifications to the strategies usually used in codes implementing those methods. In this paper we explore the causes of the difficulties and describe modifications which enable codes based on BDF to solve a wider class of problems than were previously possible. Additionally, we suggest strategies for detecting the problems which cannot be solved with this technique.

Several authors [1], [2], [3], [4], [5], [6], [7] have written codes designed to deal with either DAE systems of the form

$$\mathbf{F}(t, \mathbf{y}, \mathbf{y}') = \mathbf{0} \tag{1.1}$$

or special cases of this general problem. These codes are based on a technique which was introduced by Gear [1]. The idea of this technique is that the derivative $y'(t)$ can be approximated by a linear combination of the solution $y(t)$ at the current mesh point and at several previous mesh points. For example, $y'(t)$ may be approximated by BDF. The simplest method for solving differential/algebraic systems is the first order BDF, or backward Euler method. In this method the derivative $y'(t_{n+1})$ at time t_{n+1} is approximated by a backward difference of $y(t)$, and the resulting system of nonlinear equations is solved for $\mathbf{y}_{n+1}$,

$$\mathbf{F}(t_{n+1}, \mathbf{y}_{n+1}, (\mathbf{y}_{n+1} - \mathbf{y}_n)/(t_{n+1} - t_n)) = 0. \tag{1.2}$$

In this way, the solution is advanced from time t_n to time t_{n+1}. In this report we will assume that $\mathbf{y}(t_0)$ is known.

We investigate the behavior of the backward Euler method for solving systems of the form (1.1) in detail because it is the simplest member of several classes of methods

* Received by the editors April 6, 1981, and in revised form October 23, 1981.

† Applied Mathematics Division, Sandia National Laboratories, Livermore, California 94550.

which could conceivably be used for solving systems of the form (1.1). Even this simple method exhibits several serious difficulties when used in attempting to solve certain types of differential/algebraic problems.

All of the difficulties that we describe occur in solving simple linear problems. These problems are much more easily understood than nonlinear problems, and we hope that an understanding of the linear models will provide a plan for action in the nonlinear case. It is likely that difficulties in solving nonlinear problems are at least as great as for related linear problems. Thus, in the first few sections of this report we will be concerned only with linear problems. Later sections examine how the difficulties which occur in solving certain linear systems might also affect strategies for solving the general nonlinear problem (1.1).

We summarize in § 2 results of Sincovec et al. [2] on the decomposition of linear differential/algebraic systems of the form

$$Ey' = Ay + \mathbf{g}(t) \tag{1.3}$$

into canonical subsystems and on the properties of solutions of these subsystems. The structure of linear DAE systems can be characterized by a parameter m called the nilpotency of the system. This is important because the type of numerical difficulties which can be expected depends on the nilpotency of the system to be solved. Standard ODE systems have nilpotency $m = 0$.

In § 3 we describe the difficulties which arise in solving some of these canonical subsystems using the backward Euler method with varying stepsizes. We find that problems of nilpotency $m \leqq 2$ can be solved by codes based on variable-stepsize BDF; however, the usual error estimates which are proportional to the difference between the predictor and the corrector are grossly inaccurate. Although the error in the solution tends to zero as the current stepsize is reduced, these error estimates tend to a positive nonzero limit. This causes codes using these estimates to fail unnecessarily on many ($m = 2$) systems. Unfortunately, the situation is much less hopeful for systems of nilpotency $m \geqq 3$, where varying the stepsize can lead to totally incorrect answers. Worse yet, error estimates which have been proposed [2] or used [3] in some codes would allow wrong answers to be computed for these problems, with absolutely no warning. We know of no techniques for handling these ($m \geqq 3$) problems which do not destroy the structure and sparseness of systems written in the form (1.3).

The remainder of the paper is devoted to techniques for solving problems of nilpotency $m \leqq 2$. New error estimates are derived in § 4 which enable codes to solve this extended ($m \leqq 2$) class of problems reliably. In § 5 we take up some practical issues which are of importance in codes for solving nonlinear DAE systems. In particular, strategies for deciding when the Newton iteration has converged and for detecting problems of nilpotency $m \geqq 3$ (i.e., those which cannot be solved by variable-step BDF) are discussed. We make some recommendations in those areas, but there is still much work to be done.

2. Linear differential/algebraic systems.

This section reviews the structure of linear differential/algebraic systems and the properties of solutions of these systems. The results discussed in this section are derived and explained in greater detail by Sincovec et al. [2]. We summarize the main points here because they are necessary background for the understanding of the remainder of this report.

The system we consider in this and the next section is

$$Ey' = Ay + \mathbf{g}(t), \qquad y(t_0) = \mathbf{y}_0. \tag{2.1}$$

Gantmacher [8] has given a complete analysis of the general matrix pencil, $A-\lambda E$, where A and E are $N\times N$ matrices and A or E or both can be singular. The key result in [8] and [2] is that there exist nonsingular matrices P and Q which reduce the matrix pencil $A-\lambda E$ to a canonical form. When P and Q are applied to (2.1), we obtain

$$PEQQ^{-1}\mathbf{y}' = PAQQ^{-1}\mathbf{y}+P\mathbf{g}(t). \tag{2.2}$$

System (2.2) is composed of five types of uncoupled canonical subsystems. Three of the types correspond to cases where no solutions exist or infinitely many solutions exist. It is not even reasonable to try to solve these problems numerically. Fortunately, codes based on BDF reject these problems almost automatically, because the iteration matrix $E-h\beta A$, (where h is the stepsize and β is a scalar which depends on the method and recent stepsize history) is singular for all values of $h\beta$. The remaining two types of canonical subsystems correspond to the case where $A-\lambda E$ is a regular matrix pencil, that is, $\det(A-\lambda E)$ is not identically zero. In [2], these systems are called "solvable" because solutions to the differential/algebraic equation exist and two solutions which share the same initial value must be identical. In what follows, we will deal only with systems where the matrix pencil is regular.

For solvable systems (2.2) is equivalent to

$$\mathbf{y}_1'(t) = E_1\mathbf{y}_1(t)+\mathbf{g}_1(t), \qquad \mathbf{y}_1(t_0)=\mathbf{y}_{1,0}, \tag{2.3a}$$

$$E_2\mathbf{y}_2'(t) = \mathbf{y}_2(t)+\mathbf{g}_2(t), \qquad \mathbf{y}_2(t_0)=\mathbf{y}_{2,0}, \tag{2.3b}$$

where

$$Q^{-1}\mathbf{y}(t)=\begin{bmatrix}\mathbf{y}_1(t)\\ \mathbf{y}_2(t)\end{bmatrix}, \qquad P\mathbf{g}(t)=\begin{bmatrix}\mathbf{g}_1(t)\\ \mathbf{g}_2(t)\end{bmatrix}$$

and E_2 has the property that there exists an integer m such that $E_2^m=0$, $E_2^{m-1}\neq 0$. The value of m is defined to be the nilpotency of the system. The matrix E_2 is always composed of Jordan blocks of the form

$$\begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ & & & & 0 \end{bmatrix}, \tag{2.4}$$

and m is the size of the largest of these blocks.

The behavior of numerical methods for solving standard ODE systems of the form (2.3a) is well understood, and will not be elaborated upon here. Since the subsystems are completely uncoupled and the methods we are interested in are linear, it suffices for understanding (2.1) to study the action of numerical methods on subsystems of the form (2.3b), where E_2 is a single block of form (2.4). When E_2 is a matrix of form (2.4) and size n, the system (2.3b) will be referred to as a canonical nonstate $(m=n)$ subsystem.

Let us now take a closer look at one of these canonical nonstate subsystems. For example, the simplest $(m=2)$ system is

$$y_2'(t)=y_1(t)+g_1(t), \qquad 0=y_2(t)+g_2(t). \tag{2.5}$$

This system has the solution

$$y_2(t) = -g_2(t), \qquad y_1(t) = -g_1(t) - g_2'(t). \tag{2.6}$$

We will deal with this example repeatedly in this report. This system differs from conventional ODE systems in several significant ways. First, note that the solution depends on g (and its derivatives) at the current time only. That is, it does not depend on the initial value or on the past history of g. Also, observe that the solution to (2.5) depends on the derivative of $g_2(t)$. Thus, if $g_2(t)$ is differentiable but not continuously differentiable, then $y_1(t)$ is discontinuous. Numerical methods have a great deal of difficulty in dealing with this situation. In view of the obvious differences between nonstate systems and state systems (standard ODE systems), it is not surprising that methods designed to deal with standard ODE systems should experience so many problems with the nonstate systems. In fact, what is surprising is that these methods actually can solve some of the nonstate systems, as we will show later. For the general nonstate canonical subsystem (2.3b) of nilpotency m, the solution is given by

$$\mathbf{y}_2(t) = -\sum_{i=0}^{m-1} E_2^i \mathbf{g}_2^{(i)}(t), \tag{2.7}$$

and it is easy to see that this system shares the properties described above.

One other point to be made before we move on to numerical methods is that differential/algebraic systems are very similar to stiff systems. For example, if $\varepsilon > 0$ small, then

$$(E - \varepsilon A)\mathbf{z}'(t) = A\mathbf{z}(t) + \mathbf{g}(t) \tag{2.8}$$

is a stiff system near to (2.1). Thus, we expect that if the underlying differential/algebraic system ($\varepsilon = 0$) has nilpotency $m \geqq 2$ many of the difficulties which are described in this report should occur in problems like (2.8). Of course, the stiff system can be solved using a small enough stepsize, but this may be very inefficient. We do not know whether stiff problems with this structure occur very often in practice but when they do most codes will have trouble solving them.

3. Discontinuities, errors and error estimates. This section describes several problems which are likely to arise in solving certain linear differential/algebraic systems. First we look at what happens when codes based on BDF with the usual error control strategy are faced with problems of nilpotency $m = 2$. Codes behave rather strangely in these circumstances and may even fail because the error estimates do not reflect the true behavior of the error. Later in this section we examine the difficulties inherent in solving systems of nilpotency $m \geqq 3$ and indicate why these problems are not solvable by codes based on variable-stepsize BDF. For simplicity, all of our examples use the backward Euler method and low order error estimates. However, it is easy to see that the same types of difficulties occur for higher order methods and higher order error estimates.

Our first example is a problem involving a discontinuity in the dependent variable. This problem is not "solvable" since the solution is not defined at the point of the discontinuity (though it exists and is unique everywhere else). However, it is easy to generate such a problem without realizing it even with a differentiable input function. Hence, we feel that a code should at least fail leaving some indication of the cause of the difficulty on this type of problem. Furthermore, we find later that the difficulties in solving this example problem occur in exactly the same way for continuous, "solvable" systems.

Consider solving the system

$$E\mathbf{y}' = A\mathbf{y} + \mathbf{g}(t). \tag{3.1}$$

Several unpleasant problems may occur if some derivative of $\mathbf{g}(t)$ is discontinuous. First of all, recall from (2.7) that a discontinuity in a derivative of $\mathbf{g}(t)$ may lead to a discontinuity in some component of $\mathbf{y}$. If we were solving such a problem numerically, we would hope that the code could find the discontinuity and pass over it or at least would fail at the discontinuity. Instead, a code based on BDF with the usual error control mechanism (error estimates proportional to the difference between the predictor and the corrector) is likely to fail *not* on the step which spans the discontinuity but on the subsequent step. This leaves us with little indication that the discontinuity was the source of the difficulties. These problems occur even when using a one-step method, which normally (for standard form ODE systems) we think of as having no memory.

How can this happen? As an example, consider the problem

$$\begin{gathered} y_2'(t) = y_1(t), \qquad 0 = y_2(t) - g(t), \\ \text{where } g(t) \text{ is given by } g(t) = \begin{cases} 0, & t \leqq 0 \\ ct, & t > 0 \end{cases}. \end{gathered} \tag{3.2}$$

Suppose $y_1 \equiv y_2 \equiv 0$ for $t < 0$ and that the error estimate (which we will use for accepting or rejecting the step and for choosing the next stepsize) is proportional to $\|\mathbf{y}_{n+1} - \mathbf{y}_n\|$. Now, take one step with the backward Euler method, assuming that t_{n-1} is the time corresponding to the last accepted step to the left of zero (that is, assume $t_{n-1} < 0$, $t_n > 0$). Then, at time t_n we obtain

$$y_{2,n} = g(t_n) = ct_n, \qquad y_{1,n} = \frac{g(t_n) - g(t_{n-1})}{h_n} = \frac{ct_n}{h_n},$$

where $h_n = t_n - t_{n-1}$. Now, as t_n approaches zero, h_n is bounded away from zero as long as $t_{n-1} < 0$ is fixed. So the error estimate,

$$\left\| \begin{matrix} y_{1,n} - y_{1,n-1} \\ y_{2,n} - y_{2,n-1} \end{matrix} \right\| = \left\| \begin{matrix} ct_n/h_n - 0 \\ g(t_n) - g(t_{n-1}) \end{matrix} \right\|$$

approaches zero as $t_n \to 0$ and for some $t_n > 0$ the step is accepted.

There is already a problem at this point as $y_{1,n}$ is in error by $y_{1,n} - y_1(t_n) = ct_n/h_n - c = c(1 - t_n/h_n)$ so that unless either t_{n-1} or t_n is exactly zero, we can construct problems (by choosing c large enough) for which the error in $Y_{1,n}$ is arbitrarily large but the step is accepted. However, this is not as bad as it may at first seem because for $t_n \to 0_+$ we get $\mathbf{y}_n \to 0$, which is the correct solution for a nearby time (a time less than zero).

Since the step to t_n is accepted for some $t_n > 0$, $t_{n-1} < 0$, the code will continue, taking another step to t_{n+1},

$$y_{2,n+1} = g(t_{n+1}) = ct_{n+1}, \qquad y_{1,n+1} = \frac{g(t_{n+1}) - g(t_n)}{h_{n+1}}.$$

These solutions are exactly correct (we can expect this to happen only for linear systems with nilpotency m equal to 2), but the error estimate is

$$\left\| \begin{matrix} c(1 - t_n/h_n) \\ ch_{n+1} \end{matrix} \right\|.$$

The first component of this error estimate is independent of h_{n+1} so that we cannot make the estimate as small as we want by reducing h_{n+1}. Thus, for large enough c, the code will fail on this step—even though the solution is exactly correct.

We can get a better understanding of the cause of this problem with the error estimates by observing what happens graphically. Figure 3.1 shows $g(t)$.

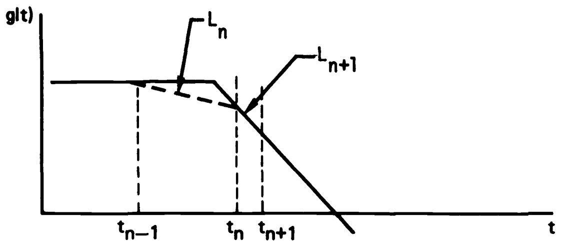


FIG. 3.1

Let L_n be the line joining $g(t_{n-1})$ and $g(t_n)$. The numerical solution $y_{1,n}$ is the slope of the secant line L_n. At the time when we are trying to compute $\mathbf{y}_{n+1}$, this line is fixed because the step from t_{n-1} to t_n has been accepted. The solution for $y_{1,n+1}$ is the slope of the line L_{n+1} between $g(t_n)$ and $g(t_{n+1})$. But as $t_{n+1} \to t_n$ (when the code is reducing the stepsize h_{n+1} to try to obtain a smaller error estimate), this line (L_{n+1}) approaches the tangent of g at t_n. Unless g is linear, the slope of the tangent at t_n is not the same as the slope of the secant from t_{n-1} to t_n. If the difference between these two slopes is bigger than the error estimate, then the code will fail on this step. In any case, error estimates based on the difference between the predictor and the corrector fail to reflect the true magnitude of the error for this problem. Recall that the predictor is just a polynomial extrapolating through past values of the solution, so that the predictor gets arbitrarily close to the most recent solution value as $h_{n+1} \to 0$, and these estimates behave qualitatively in the same way as the simpler estimates we have been considering.

It is easily seen that these difficulties with the error estimate are not limited to problems whose solutions are discontinuous. For example, consider solving the same problem as before except with $g(t)$ given in Fig. 3.2.

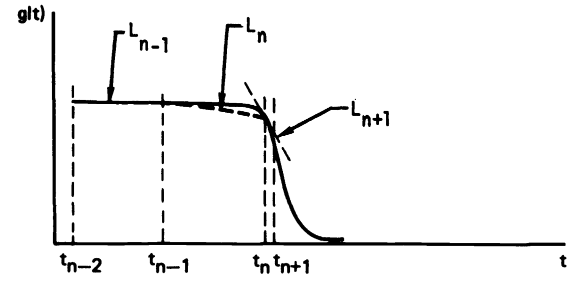


FIG. 3.2

This problem has a steep gradient. Since the slope of L_{n-1} is not much different from the slope of L_n, the step to t_n is accepted. But in the next step, the slope is much different, and we cannot close the gap because the new slope (of L_{n+1}) just gets nearer and nearer to the slope of the tangent at t_n. Since the slope of the tangent is far from the slope of L_n, the code fails on this step.

We conclude that 1) in general, for systems of the form (3.1), the difference between the predictor and the corrector does not approach zero as the current stepsize approaches zero and 2) codes using error estimates based on this difference may fail on problems with steep gradients, and these error estimates seem to bear little resemblance to the errors which are actually incurred for some DAE systems.

Are the problems described above due to poor error estimates or is there some fundamental problem with using backward Euler with varying stepsizes for solving linear DAE systems with nilpotency $m \geqq 2$? (To save writing these will be called ($m \geqq 2$) systems.) To gain a better understanding of the source of this problem, we take a closer look at the errors at each step compared with the error estimates.

Suppose we are starting with initial values at t_n equal to the exact solution. Then what is the error after one step in solving (3.1) with backward Euler? Taking one step, we obtain

$$E\left(\frac{\mathbf{y}_{n+1}-\mathbf{y}_n}{h_{n+1}}\right)=A\mathbf{y}_{n+1}+\mathbf{g}(t_{n+1}). \tag{3.3}$$

If $\mathbf{y}(t_{n+1})$ is the true solution to (3.1) at time t_{n+1}, then we have

$$\mathbf{y}(t_{n+1})=\mathbf{y}(t_n)+h_{n+1}\mathbf{y}'(t_{n+1})+\frac{h_{n+1}^2}{2}\mathbf{y}''(\xi), \tag{3.4}$$

where $t_n \leqq \xi \leqq t_{n+1}$. Substituting (3.4) for $\mathbf{y}'(t_{n+1})$ in (3.1), we obtain

$$E\left(\frac{\mathbf{y}(t_{n+1})-\mathbf{y}(t_n)-\frac{h_{n+1}^2}{2}\mathbf{y}''(\xi)}{h_{n+1}}\right)=A\mathbf{y}(t_{n+1})+\mathbf{g}(t_{n+1}). \tag{3.5}$$

Now, subtracting (3.5) from (3.3), if $\mathbf{e}_{n+1}=\mathbf{y}_{n+1}-\mathbf{y}(t_{n+1})$,

$$E\left(\frac{\mathbf{e}_{n+1}-\mathbf{e}_n+\frac{h_{n+1}^2}{2}\mathbf{y}''(\xi)}{h_{n+1}}\right)=A\mathbf{e}_{n+1}, \tag{3.6}$$

or, rewriting,

$$\mathbf{e}_{n+1}=(E-h_{n+1}A)^{-1}E\mathbf{e}_n-(E-h_{n+1}A)^{-1}E\left(\frac{h_{n-1}^2}{2}\right)\mathbf{y}''(\xi). \tag{3.7}$$

Now, (3.7) says that the error after one step, starting from exact initial values (so that $\mathbf{e}_0=0$) is

$$\mathbf{e}_1=-(E-h_{n+1}A)^{-1}E\left(\frac{h_{n+1}^2}{2}\right)\mathbf{y}''(\xi). \tag{3.8}$$

There are several consequences of this simple expression. For example, for the ($m=2$) problem (3.2), we have

$$\mathbf{y}''(\xi)=\begin{bmatrix} g'''(\xi) \\ g''(\xi) \end{bmatrix}$$

and

$$(E-h_{n+1}A)^{-1}E=\begin{bmatrix} 0 & -1/h_{n+1} \\ 0 & 0 \end{bmatrix},$$

so that the error after one step is

$$\mathbf{e}_1=\begin{bmatrix} h_{n+1}g''(\xi)/2 \\ 0 \end{bmatrix}. \tag{3.9}$$

We observe from (3.9) that the error is $O(h_{n+1})$, not $O(h_{n+1}^2)$, and that it depends on $y_2''(t)$ and not $y_1''(t)$. Thus, if we had a good estimate for $\mathbf{y}''$, then the usual error estimate based on $(h_{n+1}^2/2)\mathbf{y}''$ would be asymptotically a gross underestimate. The term $(E-h_{n+1}A)^{-1}E(h_{n+1}^2/2)\mathbf{y}''(\xi)$ is in a sense the local contribution to the global error. Notice that we cannot really define a local error in the usual way as $\mathbf{y}_{n+1}-\mathbf{u}(x_{n+1})$, where $\mathbf{u}(x_n)=\mathbf{y}_n$, because there may not be a solution passing through $\mathbf{y}_n$.

The situation for $(m=2)$ nonstate subsystems is obviously enough to wreak havoc with any stepsize selection algorithm which assumes that errors are $O(h^{k+1})$, where k is the order of the method. Furthermore, the usual error estimates can cause a code to do very strange things when $g(t)$ has a steep gradient. Despite all of this, the situation is not at all without hope. The error in y_{n+1} at any step can be reduced by reducing h_{n+1}, (recall, however, that the difference between the predictor and corrector is not reduced) so that in principle, if we knew how to adjust the stepsize, the error in y_{n+1} could be controlled by locally adjusting h_{n+1}. In the next section, we will find error estimates which can accomplish this task.

Unfortunately, there are several even more severe problems in solving systems with nilpotency $m \geqq 3$. For the $(m=3)$ system

$$y_2'(t)=y_1(t), \quad y_3'(t)=y_2(t), \quad 0=y_3(t)-g(t), \tag{3.10}$$

we have

$$(E-h_{n+1}A)^{-1}E=\begin{bmatrix} 0 & -1/h_{n+1} & -1/h_{n+1}^2 \\ 0 & 0 & -1/h_{n+1} \\ 0 & 0 & 0 \end{bmatrix}$$

and

$$\frac{h_{n+1}^2}{2}\mathbf{y}''(\xi)=\frac{h_{n+1}^2}{2}\begin{bmatrix} g^{iv}(\xi) \\ g'''(\xi) \\ g''(\xi) \end{bmatrix},$$

so that the error in $\mathbf{y}$ after one step starting from the exact solution is

$$\mathbf{e}_1=\begin{pmatrix} \dfrac{g''(\xi)}{2}+\dfrac{h_{n+1}}{2}g'''(\xi) \\ \dfrac{h_{n+1}}{2}g''(\xi) \\ 0 \end{pmatrix}. \tag{3.11}$$

Restating this, we cannot choose h_{n+1} small enough so that the error in the solution after one step starting from exact initial values is small. This poses a very difficult problem in finding initial values for $m \geqq 3$ systems, if even the exact solution will not do.

This observation seems to conflict with a theorem proved in [2]. Of course, it is not really a conflict but only appears that way because these results are not nearly as strong as what we are accustomed to for standard ODE's. The theorem states that the backward Euler method and k-step BDF converge to the analytic solution of the system (3.1). Furthermore, it is stated that the error in solving nonstate subsystems with the k-step method is $O(h^k)$, where h is the stepsize. How can the method converge when the error after the first step is independent of h? We must be very careful about interpreting the qualifications on this theorem. The convergence, at least for systems where $m \geqq 3$, starting from the exact solution, only applies to the solution at the endpoint of some fixed interval of integration. This is because the first $m-2$ solution values contain errors which do not become arbitrarily small when h is decreased. The results at later steps depend only on the function $g(t)$ at past steps and not on the initial values so that the solution converges in any interval bounded away from the initial time. The theorem is only true for linear systems, and for these systems, this behavior may be tolerable if we have anticipated it. However, this is a very serious problem if what we are really trying to solve is a nonlinear system. In that case, large errors in the first few steps may persist throughout the entire interval.

The other qualification on the theorem is that it is only true for constant stepsizes. We can see this by applying backward Euler to the $m=3$ system (3.9),

$$\begin{aligned} &y_{3,n+1}=g(t_{n+1}),\\ (3.12)\qquad &y_{2,n+1}=(g(t_{n+1})-g(t_n))/h_{n+1},\\ &y_{1,n+1}=\left(\left(\frac{g(t_{n+1})-g(t_n)}{h_{n+1}}\right)-\left(\frac{g(t_n)-g(t_{n-1})}{h_n}\right)\right)\Big/h_{n+1}.\end{aligned}$$

Now if $h_{n+1}=h_n=h$, then $y_{1,n+1}$ is given by $(\nabla^2 g/h^2)$ which converges to g'' as $h\to 0$ as the theorem predicts. When $h_{n+1}\neq h_n$ we might hope that $y_{1,n+1}$ would be given by the divided difference $2[g_{n+1}, g_n, g_{n-1}]$, where

$$(3.13)\qquad 2[g_{n+1}, g_n, g_{n-1}]=\frac{\dfrac{g_{n+1}-g_n}{h_{n+1}}-\dfrac{g_n-g_{n-1}}{h_n}}{\dfrac{h_{n+1}+h_n}{2}},$$

but the method fails to pick up these differences correctly. The error in the approximation to $y_{1,n+1}$ which is caused by changing the stepsize has the form $\frac{1}{2}(1-h_n/h_{n+1})g''(\xi)+O(h_{n+1})$. This term becomes arbitrarily large as $h_{n+1}\to O$ (h_n fixed). If we were very careful to select stepsizes so that $h_{n+1}=h_n(1+O(h_n))$, then this problem could be avoided. However, this seems to be very difficult to accomplish in a practical code and sequences of stepsizes would have to satisfy even more stringent restrictions for systems with $m>3$.

We have tried to illustrate in this section that while the BDF can, in principle, solve (3.1) (via the theorem [2]), there are a great many qualifications to this statement which can cause serious difficulties in any kind of a practical code. At present, we know of no way to adequately handle (3.1) when $m\geqq 3$ subsystems are present. The most serious problems seem to be in starting the code and in changing stepsize. The remainder of this paper will be concerned with handling ($m\leqq 2$) systems and related nonlinear problems and detecting the other problems that codes based on methods such as BDF cannot solve.

4. Error estimation. In this section we examine several potential candidates for error estimates for differential/algebraic systems. Our aim is to find an estimate which accurately reflects the behavior of the error for linear ($m\leqq 2$) systems. Additionally, we hope to detect ($m\geqq 3$) systems which cannot be solved accurately using BDF with varying stepsizes and to give an accurate indication of why the code fails to solve these problems.

Several different approaches to error estimation for DAE systems have been reported in the literature. Gear and Brown [3] solve systems of the form

$$\mathbf{f}(t, \mathbf{y}, \mathbf{y}') + P\mathbf{v} = 0, \tag{4.1}$$

where $\mathbf{y}$ and $\mathbf{y}'$ are of length p, $\mathbf{v}$ is of length $q-p$, P is a $qx(q-p)$ matrix and $\mathbf{f}$ is a vector function of length q. A close look at their code [3] reveals that there is no attempt to estimate errors in $\mathbf{v}$. In some respects this makes sense as $\mathbf{v}$ is completely determined by $\mathbf{y}$ on every step so that errors in $\mathbf{v}$ do not cause errors in $\mathbf{y}$. If for some reason we are actually interested in the values of $\mathbf{v}$, we must understand that they may contain very large errors. For example, the $(m=3)$ system (3.9) may be rewritten in the form (4.1), with $v = y_1$, $z_1 = y_2$, $z_2 = y_3$, as

$$z_1' - v = 0, \quad z_2' - z_1 = 0, \quad z_2 - g(t) = 0,$$

and error control is not attempted on v (on y_1). But this is exactly the component that exhibits arbitrarily large errors after a change of stepsize. Also, if we should fail to realize that y_1 occurs linearly, then the behavior of the code is very different.

Recently, a different approach to error control was proposed by Sincovec et al. [2] for linear DAE systems. They observed that errors in nonstate components (subsystem (2.3b)) have a different asymptotic behavior than errors in state components (subsystems (2.3a)) so that stepsize selection schemes which assume a certain asymptotic behavior of the error would have difficulty in controlling the errors in these components. In addition, errors in nonstate components affect the solution only locally, and are not propagated globally to state components. As a consequence, they proposed to "filter out" the part of the error estimate that corresponds to the nonstate solution components. This is accomplished by monitoring a projection of the usual error estimate, where the new estimate is given by

$$\varepsilon_n^* = \|Mc_{n,k}(\mathbf{y}_n^c - \mathbf{y}_n^p)\|. \tag{4.2}$$

Here $\mathbf{y}_n^c$ is the final corrected value of $\mathbf{y}$ at the end of a step, $\mathbf{y}_n^p$ is the predicted value of $\mathbf{y}$ and $c_{n,k}$ is a constant depending on the method and recent stepsize history of the integration. (Note that $\|c_{n,k}(\mathbf{y}_n^c - \mathbf{y}_n^p)\|$ is the error estimate which is generally used in ODE codes.) M is called the canonical state variable projection matrix and is given by

$$M = \lim_{h\to 0} M(h, j), \tag{4.3}$$

where $M(h, j) = ((E - hA)^{-1}E)^j$, where j is greater than or equal to the nilpotency of the system.

It is shown in [2] that this error estimate has the effect of filtering out that part of $(\mathbf{y}^c - \mathbf{y}^p)$ which is associated with nonstate variables (if the system were transformed to canonical form), without the expensive operations of transforming the system into canonical form. It is convenient because LU decompositions of the matrices $(E - hA)$ are always available (for h the current stepsize) because that is the iteration matrix for the Newton iteration. It is somewhat inconvenient in that it is hard to find out what the nilpotency of the system is, and we wonder what to do about systems which are "almost" nilpotent.

Unfortunately, there is one very bad defect in this filtered error estimation scheme and, indeed, in nearly any scheme which fails to control the errors in certain components of the solution. With this particular scheme we can, for example, "solve" any

($m \geqq 3$) system, the stepsize being chosen to control errors in the state components of the system. However, as we have already seen, this can be very misleading because large errors are introduced into some components of the solution whenever the stepsize is changed.

The examples that we have given show that it is a very dangerous practice not to control errors in some components of the solution. The only possible circumstances under which we feel this could be done safely are if 1) we are not interested in the value of that component, *and* we are sure that errors in that component cannot be propagated into any other component later in the integration, or 2) if we are sure that no errors are being made in that component. For example, if in solving (4.1) we are not interested in the value of $\mathbf{v}$, then it is safe to omit those variables from the error control.

We have already noted the similarities between stiff systems and differential/algebraic systems. It is natural to look at error estimation schemes proposed for stiff equations. Curtis [7] noted that for a single linear ODE, $y' = -\lambda(y - f(t)) + f'(t)$ end-step errors are smaller than the usual error estimate by a factor $1/\lambda h$, where $-\lambda$ is the eigenvalue of the Jacobian matrix and h the step size. When λ is large, this problem is very nearly the same as the ($m = 1$) algebraic system $y = f(t)$, which we solve exactly on every step. On the other hand, we saw in § 3 that for some problems, the usual error estimate can severely underestimate the error.

Sacks-Davis [9] noted that for stiff problems, the usual error estimate based on the difference between the predictor and the corrector overestimates the true error. He suggests error estimates for second derivative methods which are asymptotically correct as $h \to 0$, and are reliable and efficient for very stiff problems. The estimates have the form

$$\varepsilon_n^{**} = \|W_n^{-1} c_{n,k}(\mathbf{y}_n^c - \mathbf{y}_n^p)\|, \tag{4.4}$$

where W_n is the iteration matrix for the second derivative method.

If we examine an estimate similar to (4.4) for BDF, where W_n is the iteration matrix for the kth order BDF, then it is easy to see that ε_n^{**} from (4.4) is the same as the estimate (4.2), if $m = 1$ and $M(h_{n+1}, 1)$ is used in place of M and if the matrix E in (3.1) is nonsingular. We also note from (3.7) that the local contribution to the global error for the backward Euler method is

$$s(E - hA)^{-1} E \frac{h^2}{2} \mathbf{y}''(\xi).$$

Because of these observations, we are led to try the estimate

$$\varepsilon_n = \|(E - \beta_{n,k} h_n A)^{-1} E c_{n,k}(\mathbf{y}_n^c - \mathbf{y}_n^p)\|, \tag{4.5}$$

where $\beta_{n,k}$ and $c_{n,k}$ are constants depending on the method used and possibly on the recent stepsize history. For a standard-form ODE, (4.5) is asymptotically equivalent to the usual estimate, $\|c_{n,k}(\mathbf{y}_n^c - \mathbf{y}_n^p)\|$, so that the question we must answer is how well the estimate performs on the nonstate and/or stiff components of a system.

On first glance the estimate (4.5) might seem to contradict statements that were made earlier, as this estimate does not "control the error" in algebraic ($m = 1$) subsystems. These subsystems have the form $y(t) = f(t)$, and they are solved by the method exactly (apart from errors due to terminating the Newton iteration, which are discussed in the next section), so this strategy is not unreasonable. A problem with (4.5) is that if a code interpolates to find the solution at user-specified output points then this estimate does not control the error in the interpolation. These errors seem to go to zero as the stepsizes get smaller and smaller, but we know of no way to explicitly control them without incurring at the same time all of the problems mentioned in § 3.

How accurately does the estimate (4.5) reflect the error for $(m=2)$ nonstate subsystems? From (3.7), we can see that for backward Euler (4.5) accurately estimates the local contribution to the global error as long as $\mathbf{y}''(\xi)$ can be found accurately. Actually, for the canonical $(m=2)$ nonstate subsystems (3.2), we have

$$(E-hA)^{-1}E=\begin{bmatrix}0 & -1/h\\ 0 & 0\end{bmatrix},$$

so that only $y_2''(\xi)$ is involved in the estimate. This is the second derivative of the input function $g(t)$, and it can be estimated reasonably well by a divided difference of $g(t)$, so that the estimate (4.5) works. It is fortunate that the estimate does not make use of $y_1''(\xi)$, because difficulties in obtaining this term were in part responsible for the problems discussed in § 3. One other point we make is that for a $(m=2)$ subsystem the local contribution to the global error is the global error (neglecting roundoff and errors due to terminating the Newton iteration early). This happens because, from (2.3b), the nonstate subsystems can be written in the form

$$E_2\mathbf{y}'=\mathbf{y}+g(t),$$

where $E_2^2=0$ and then, from (3.7),

$$\mathbf{e}_{n+1}=(E_2-h_{n+1}I)^{-1}E_2\mathbf{e}_n+(E_2-h_{n+1}I)^{-1}E_2\boldsymbol{\tau}_{n+1},$$

where $\boldsymbol{\tau}_{n+1}=(h_{n+1}^2/2)\mathbf{y}''(\xi)$. Rewriting this, we obtain

$$\mathbf{e}_{n+1}=(E_2-h_{n+1}I)^{-1}E_2(E_2-h_nI)^{-1}E_2[\mathbf{e}_{n-1}+\boldsymbol{\tau}_n]+(E_2-h_{n+1}I)^{-1}E_2\boldsymbol{\tau}_{n+1},$$

but the matrix $(E_2-h_{n+1}I)^{-1}E_2(E_2-h_nI)^{-1}E_2$ is identically zero for this case so that

$$\mathbf{e}_{n+1}=(E_2-h_{n+1}I)^{-1}E_2\boldsymbol{\tau}_{n+1}.$$

It is easy to verify that the estimate (4.5) also accurately reflects the behavior of the error for all of the BDF for $(m\leqq 2)$ systems. In general, the kth order BDF approximates $h_{n+1}\mathbf{y}'_{n+1}$ by $\sum_0^k \alpha_{i,n+1}\mathbf{y}_{n+1,i}$, where $\alpha_{i,n+1}$ depend on the order and recent stepsize history, so that $\mathbf{y}_{n+1}$ satisfies

$$E\left(\sum_0^k \alpha_{i,n+1}\mathbf{y}_{n+1-i}\right)=h_{n+1}A\mathbf{y}_{n+1}+h_{n+1}\mathbf{g}(t_{n+1}), \tag{4.6}$$

or, rewriting,

$$(\alpha_{0,n+1}E-h_{n+1}A)\mathbf{y}_{n+1}=-E\sum_1^k \alpha_{i,n+1}\mathbf{y}_{n+1-i}+h_{n+1}E\mathbf{g}_{n+1}. \tag{4.7}$$

Now, if $\boldsymbol{\tau}_{n+1}$ is defined by

$$\boldsymbol{\tau}_{n+1}=h_{n+1}\mathbf{y}'(t_{n+1})-\sum_0^k \alpha_{i,n+1}\mathbf{y}(t_{n+1-i}) \tag{4.8}$$

($\boldsymbol{\tau}_{n+1}$ is usually called the local truncation error of the method), then

$$(\alpha_{0,n+1}E-h_{n+1}A)\mathbf{y}(t_{n+1})=-E\sum_1^k \alpha_{1,n+1}\mathbf{y}(t_{n+1-i})+h_{n+1}Eg_{n+1}-E\boldsymbol{\tau}_{n+1}. \tag{4.9}$$

Subtracting (4.9) from (4.7), we have, if $\mathbf{e}_n=\mathbf{y}_n-\mathbf{y}(t_n)$,

$$\mathbf{e}_{n+1}=(\alpha_{0,n+1}E-h_{n+1}A)^{-1}E\sum_1^k \alpha_{i,n+1}\mathbf{e}_{n+1-i}+(\alpha_{0,n+1}E-h_{n+1}A)^{-1}E\boldsymbol{\tau}_{n+1} \tag{4.10}$$

so that $(\alpha_{0,n+1}E - h_{n+1}A)^{-1}E\boldsymbol{\tau}_{n+1}$ is the local contribution to the global error for variable-stepsize BDF, and this has the same form as the estimate (4.5).

Since $\boldsymbol{\tau}_{n+1}$ is the local truncation error of a variable-stepsize BDF, it depends on the recent stepsize history of the computation. For example, for a variable-step 2nd order BDF,

$$\|\boldsymbol{\tau}_{n+1}\| \leqq h_{n+1}^2(h_{n+1}+h_n)\|\mathbf{y}^{(k+1)}(\xi)\|. \tag{4.11}$$

Thus, the error as $h_{n+1} \to O$ (h_n fixed) has the form $O(h_{n+1}^2)$, not $O(h_{n+1}^3)$. It is important when using this error estimate to get this dependence correctly. If we instead assume (as is done in some methods which change the stepsize via interpolation [3], [5], [6]) that the error depends on the current stepsize as $O(h_{n+1}^3)$, this can cause the code to severely underestimate the error when the solution is just beginning to change rapidly. This is a particular problem for ($m \geqq 3$) nonstate subsystems, where the error does not go to zero as $h_{n+1} \to 0$ (h_n fixed), and the error estimate is given by (for $m = 3$)

$$\boldsymbol{\varepsilon}_{n+1} = \begin{bmatrix} 0 & -1/h_{n+1} & -1/h_{n+1}^2 \\ 0 & 0 & -1/h_{n+1} \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{\tau}_{n+1}. \tag{4.12}$$

If we had taken $\boldsymbol{\tau}_{n+1} = O(h_{n+1}^3)$, then the error estimate would become smaller as h_{n+1} decreases and we would be led to believe that errors were under control, when in reality there is a large error in the first component. It is especially important for DAE systems and the error estimate (4.5) to use the actual (variable stepsize) principal term of the local truncation error in the error estimate rather than an approximation which assumes that the last k steps were taken with a constant stepsize.

In summary, the error estimate (4.5) seems to be a useful alternative to the usual estimates based on a difference between the predictor and the corrector. It reflects the behavior of the error more accurately than the usual error estimate and overcomes many of the difficulties mentioned in § 3 for systems containing ($m = 2$) nonstate subsystems. Use of this estimate enables codes based on BDF to solve a wider class of problems. The estimate is easily generalized to nonlinear problems. While we have no theory to support this generalization, it seems to have worked well in our experience.

5. Practical issues. It is evident from the problems mentioned in § 3 that DAE systems are in many respects very different from ODE systems. With this in mind, it is not unreasonable to expect that codes for solving the two types of problems must be different in some respects. In this section we question several strategies used in ODE codes which one might be tempted to carry over to DAE codes to discover whether or not they are applicable to these more general problems.

Codes for solving stiff and differential/algebraic systems generally use a modified Newton iteration to solve an implicit equation for $\mathbf{y}_{n+1}$ at each step. For example, in solving

$$E\mathbf{y}' = A\mathbf{y} + \mathbf{g}(t) \tag{5.1}$$

with backward Euler, there is an implicit equation

$$E\left(\frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{h_{n+1}}\right) = A\mathbf{y}_{n+1} + \mathbf{g}(t_{n+1}) \tag{5.2}$$

which is solved for $\mathbf{y}_{n+1}$ at each step.

We will use the linear problem (5.1) as a model, though it is really the more general nonlinear problem (1.1) that we wish to gain some intuition about. Linear problems are, of course, much easier to solve because Newton's method converges in one iteration if the exact iteration matrix is used. We consider situations where the iteration matrix is not exact to learn more about how to handle nonlinear problems because these problems are much harder to understand.

Now suppose (5.2) is solved by a modified Newton iteration,

$$\mathbf{y}_{n+1}^{(k+1)}=\mathbf{y}_{n+1}^{(k)}-J^{-1}\left(E\frac{(\mathbf{y}_{n+1}^{(k)}-\mathbf{y}_n)}{h_{n+1}}-A\mathbf{y}_{n+1}^{(k)}-\mathbf{g}(t_{n+1})\right), \tag{5.3}$$

where J is an approximation to the matrix $(E/h_{n+1}-A)$ (J may have been computed in some previous step). Several decisions must be made in implementing (5.3). For example, how should we decide when the iteration has converged? If the iteration is not converging, what action should be taken? If it appears that we cannot make the iteration converge, can we discover the problem? We take up these questions here.

In deciding when to terminate the corrector iteration, our main consideration is that the error introduced by terminating this iteration early is small relative to errors due to the difference approximation to $\mathbf{y}'$ and that these errors do not affect the error estimates appreciably. Suppose then that $\boldsymbol{\delta}_{n+1}$ is the error in $\mathbf{y}_{n+1}$ due to terminating the corrector iteration early. That is, if $\tilde{\mathbf{y}}_{n+1}$ is the exact solution to the difference equation and $\mathbf{y}_{n+1}=\mathbf{y}_{n+1}^{(M)}$ is the solution accepted after M iterations, then $\tilde{\mathbf{y}}_{n+1}=\mathbf{y}_{n+1}+\boldsymbol{\delta}_{n+1}$. Then from (5.2),

$$E\left(\frac{\mathbf{y}_{n+1}-\mathbf{y}_n+\boldsymbol{\delta}_{n+1}}{h_{n+1}}\right)=A(\mathbf{y}_{n+1}+\boldsymbol{\delta}_{n+1})+\mathbf{g}(t_{n+1}). \tag{5.4}$$

The exact solution $\mathbf{y}(t)$ satisfies

$$E\left(\frac{\mathbf{y}(t_{n+1})-\mathbf{y}(t_n)+\boldsymbol{\tau}_{n+1}}{h_{n+1}}\right)=A\mathbf{y}(t_{n+1})+\mathbf{g}(t_{n+1}). \tag{5.5}$$

Subtracting (5.5) from (5.4), the global error $\mathbf{e}_n=\mathbf{y}_n-\mathbf{y}(t_n)$ satisfies

$$\mathbf{e}_{n+1}=(E-h_{n+1}A)^{-1}E\mathbf{e}_n+(E-h_{n+1}A)^{-1}E\boldsymbol{\tau}_{n+1}-\boldsymbol{\delta}_{n+1} \tag{5.6}$$

and, rewriting,

$$\begin{aligned}\mathbf{e}_{n+1}&=(E-h_{n+1}A)^{-1}E(E-h_nA)^{-1}E(\mathbf{e}_{n-1}+\boldsymbol{\tau}_n)\\&\quad-(E-h_{n+1}A)^{-1}E\boldsymbol{\delta}_n+(E-h_{n+1}A)^{-1}E\boldsymbol{\tau}_{n+1}-\boldsymbol{\delta}_{n+1}.\end{aligned}$$

We know from ODE theory that for the state variables in the system (2.3a) errors are not amplified greatly from step to step by the BDF and it is sufficient to control $\boldsymbol{\delta}_{n+1}$. Our main concern is what happens to this error for nonstate ($m=1$ and $m=2$) subsystems.

For ($m=1$) subsystems it is logical to control the error in $\mathbf{y}_{n+1}$ (that is, to control $\boldsymbol{\delta}_{n+1}$), as $E=0$ in this case, so that $\mathbf{e}_{n+1}=\boldsymbol{\delta}_{n+1}$ and $\boldsymbol{\delta}_{n+1}$ is the only error. For a ($m=2$) nonstate subsystem, the situation is not quite so clear. Let us look at the ($m=2$) canonical subsystem (2.5). In this case, the matrix $(E-h_{n+1}A)^{-1}E(E-h_nA)^{-1}E$ is identically zero so that the contribution to the global error due to terminating the iteration early is given by

$$-(E-h_{n+1}A)^{-1}E\boldsymbol{\delta}_n-\boldsymbol{\delta}_{n+1},\quad \text{where } (E-h_{n+1}A)^{-1}E=\begin{bmatrix}0 & -1/h_{n+1}\\ 0 & 0\end{bmatrix}.$$

Since $y_1=y_2'$ is approximated by a backward difference, $y_{1,n+1}\cong(y_{2,n+1}-y_{2,n})/h_{n+1}$, then an error in $y_{2,n}$ is amplified by $1/(h_{n+1})$ in $y_{1,n+1}$.

If we are only interested in iteration error as it relates to the error estimates of § 4, then this does not matter as $y_{1,n+1}$ does not enter into the error estimate so we should control $\boldsymbol{\delta}_{n+1}$. That is, terminate the iteration when

$$\|\boldsymbol{\delta}_{n+1}\| \leqq \varepsilon, \tag{5.9}$$

where ε is a constant depending on the error tolerances requested. On the other hand, for the purposes of controlling the error in y_1, we should control $\boldsymbol{\delta}_{n+1}$ so that

$$\|\boldsymbol{\delta}_{n+1}\| \leqq h_{n+1}\varepsilon \tag{5.10}$$

because the error may be amplified by $1/h_{n+1}$ on the next step. (Though we do not know in advance what the stepsize for the next step will be, we assume it will be the same order of magnitude as the current stepsize.) Starner [6] uses this test in his code though apparently for different reasons. The test (5.10) has the apparent disadvantage that it is not independent of scaling of the independent variable in the system; however, note also that any scaling of this type also scales $\mathbf{y}$. For these reasons, it is somewhat unclear whether to use (5.9) or (5.10), but there seem to be several good reasons for choosing (5.10). Shampine [10] discusses ways to bound $\|\boldsymbol{\delta}_{n+1}\|$ based on the differences between iterates, so that finding a bound for this number is not a big problem.

What if the corrector iteration fails to converge (assuming that the iteration matrix is current)? There are several reasons why our answer to this question might be different for a DAE code than for a code aimed at ODE's in standard form. Generally, the strategy taken in ODE codes is to reduce the stepsize when the iteration fails to converge. There are two reasons why this works. First, as we reduce the stepsize, the prediction gets to be a better and better initial guess for the Newton iteration. Second, the iteration matrix for a standard-form ODE $y' = f(t, y)$ is $(I - h\beta\partial F/\partial y)$, and this matrix tends towards the identity as the stepsize h is reduced. This is fortunate, because the identity matrix is very well conditioned so that errors in $\partial F/\partial y$ and in solving the linear systems affect the solution less and less as h is reduced.

The situation with differential/algebraic systems is not quite as desirable. As we have seen, the difference between the predictor and the corrector does not tend to zero as $h_{n+1} \to 0$. Thus, the initial guess may not get better with decreasing h_{n+1}. For these systems the iteration matrix $(\partial F/\partial y' - h\beta\partial F/\partial y)$ looks like $\partial F/\partial y'$ as $h \to 0$. In general, $\partial F/\partial y'$ may be singular, so that as $h_{n+1} \to 0$ the iteration matrix becomes more and more poorly conditioned. This is very troublesome for some systems, where for small enough h roundoff from solving a poorly conditioned linear system causes the corrector iteration to diverge. We are not yet certain about the proper way to handle these problems, but the following alternatives to the usual strategy seem to be worth considering. Instead of decreasing h_{n+1} when the iteration fails to converge, we could instead use a more robust iteration procedure (for example, damped Newton) or, because for at least some component a better initial guess could be obtained by reducing h_{n+1}, we could try reducing h_{n+1}, and if this doesn't help (if after reducing h_{n+1} several times, the initial guess is not much better), then switch to a more robust iteration scheme.

It is possible that the corrector iteration fails to converge despite all of our actions to try to help it. There are a great many more possibilities for the cause of this problem with a DAE system than there are for an ODE system. For an ODE system, we generally assume that this problem is caused by a very poor approximation to the Jacobian matrix. For a DAE system, it could be that we could not get a good initial guess or, maybe, the system contains a $(m \geqq 3)$ nonstate subsystem, and the initial guesses are actually diverging from the true solution to the corrector equation. The iteration may be diverging because the iteration matrix is very poorly conditioned, or the iteration matrix may be a very poor approximation to the Jacobian matrix because of errors in numerical differencing.

Given that all of the above situations are possible, our hope is to be able to diagnose at least some of these possibilities. Let us consider, for example, what happens when a code encounters a system with a ($m \geqq 3$) nonstate subsystem. For a linear or nearly linear problem where corrector convergence presents minimal difficulties, the usual response is for the stepsize to be reduced several times to try to satisfy the error test. Eventually, since the error test will never be satisfied, the stepsize is reduced to the point where the corrector iteration begins to diverge because the iteration matrix is very poorly conditioned. If we see this situation (the error test fails several times—and the error estimate is not reduced very much when h_{n+1} is reduced and then the corrector iteration fails to converge), then it is likely that the cause of the difficulty is this type of problem.

It is possible to use linear algebra routines [11] for solving the linear system which automatically generate an estimate of the condition number of the iteration matrix. This is expensive, especially for banded systems, but it may be worthwhile for differential/algebraic systems because this situation can happen so easily and if it is the cause of the difficulty, then the last thing we would want to do would be to reduce the stepsize.

One problem that we have not discussed very much here is that of finding a set of consistent initial conditions and an initial stepsize which reflects the scaling of the problem. We will not attempt to solve these problems here, but we will indicate why these problems are even more difficult than they may at first seem. From § 3, we know that for ($m \geqq 3$) nonstate subsystems, even the simplest numerical methods with constant stepsizes fail to give correct answers if they are started with the exact solution. Even restricting ourselves to problems which do not contain these subsystems, for nonlinear systems this is still a very difficult problem. From a practical point of view, even with a set of initial conditions and derivatives, we must be very careful about selecting the initial stepsize. If this stepsize is too small, we may fail to solve the problem because the iteration matrix is very poorly conditioned, even though the problem might have been solved successfully given a better (larger) choice of initial stepsize.

A final point which is of practical interest is the cost of computing the error estimate (4.5) and the information that we need to do this. If we restrict ourselves to systems of the form

$$B(t, y)y' - f(t, y) = F(t, y, y') = 0 \tag{5.11}$$

(where y' appears only linearly), we can avoid computing, storing, and multiplying by the matrix $E = \partial F/\partial y'$. (Note that if the iteration matrix $(\partial F/\partial y - h\beta\, \partial F/\partial y')$ is computed by numerical differencing each column can be computed using only one increment, as we are really finding $(\partial/\partial y_{n+1})F(t_{n+1}, y_{n+1}(y_{n+1} - y_n)/h_{n+1})$ instead of finding two separate matrices $\partial F/\partial y$ and $\partial F/\partial y'$, and adding them together. If this is done, then finding $\partial F/\partial y'$ separately is approximately twice as much work.) This can be done by requiring the user of the code to supply two separate routines. One routine computes $B(t, y)y'$ given (t, y, y')—that is, only those terms in (5.11) that involve y'. The other routine computes the full residual $B(t, y)y' - f(t, y)$, given (t, y, y'). (Alternatively, one routine which computes either of these possibilities, depending on the value of a flag, could be used.) The first routine can be used to compute $E(y_n^c - y_n^p)$ in the error estimate (4.5) given $(y_n^c - y_n^p)$ in place of y'. In many ways this seems to be a simpler interface and requires less storage than others that have been used for these problems. For example, Hindmarsh and Painter [5] solve problems of the form (5.11), requiring the user to supply routines to compute F, and to add B to a given matrix. With our suggestion, instead of having to find the matrix $B(t, y)$ and add it to a given matrix, the user of the code need only distinguish those terms of (5.11) that involve y'.

Obviously, we do not have answers to all the questions which have been discussed in this section. These seem to be difficulties which have been neglected in the literature. We feel that anyone who is seriously writing a code for solving DAE systems or using such a code should at least be aware of these difficulties.

6. Summary. In this paper we have considered some of the many difficulties which can occur in solving differential/algebraic systems. The behavior of numerical methods applied to these systems differs from what we would expect based on experience from solving ODE's in several important ways. For linear systems of nilpotency $m \leqq 2$, we have noted that the basic algorithms (BDF) which are in use will work, in the sense that as long as stepsizes and orders are chosen so that the method is stable (Gear–Tu [12], Gear–Watanabe [13]) then the computed solution converges to the true solution as the maximum stepsize approaches zero. Error estimates based on the difference between the predictor and the corrector may be grossly inaccurate for these systems and can even cause codes to fail unnecessarily. To overcome these difficulties, we have suggested an error estimate which would more accurately estimate errors for these problems and would eliminate the other difficulties associated with the usual estimates. With the new error estimates, and possibly some changes to other strategies used in ODE codes, we believe that the algorithms (such as variable-stepsize BDF) which have been used in codes for solving differential/algebraic systems in the past could be used to solve DAE systems where solutions behave similarly to solutions of linear problems with nilpotency $m \leqq 2$, and possibly to diagnose other problems which cannot be solved with these algorithms. This is a significant improvement over past codes which could not deal adequately with problems of nilpotency $m = 2$.

On the other hand, we have also found that none of the algorithms such as BDF are adequate for solving (with varying stepsizes chosen automatically by a code) problems with nilpotency $m \geqq 3$. One alarming fact that has come into focus is that some error estimation schemes which have been proposed [2] or used [3] in other codes, would allow wrong answers to be computed for some variables, with absolutely no warning. For these types of systems, the solution does not converge (except under very severe restrictions on how fast the stepsize can change) as the maximum stepsize approaches zero. Instead, large errors may be introduced into the solution whenever the stepsize is changed. Because of this situation, it is wise to use extreme caution in any attempt to avoid controlling error in certain components of the solution of any differential/algebraic system. Finding initial conditions for these problems seems to be extremely difficult because even if we start with initial conditions equal to the exact solution, the solution in the first few steps may be grossly in error. While this may not be fatal for a linear problem, because later the approximation will converge to the true solution, for a nonlinear problem it could be disastrous.

It is well known that some differential/algebriac systems can be thought of as limiting cases of stiff systems (as the stiffness becomes infinite). We have constructed several of these problems and our tests confirm that codes based on BDF exhibit many of the difficulties that we have described here. In particular, for these problems the usual error estimates are very unsatisfactory, especially when the stepsize changes. This tends to cause the stepsize to be reduced until hL (where L is the Lipschitz constant for the problem) is small. We do not know if this occurs very often in stiff problems which are of interest, but it may be a point worth considering in the design of stiff codes and algorithms.

REFERENCES

[1] C. W. GEAR, *Simultaneous numerical solution of differential-algebraic equations*, IEEE Trans. Circuit Theory, CT-18 (1971), pp. 89–95.

[2] R. F. SINCOVEC, B. DEMBART, M. A. EPTON, A. M. ERISMAN, S. W. MANKE AND E. L. YIP, *Solvability of Large Scale Descriptor Systems*, Boeing Computer Services Company, Seattle, WA, June 1979.

[3] R. L. BROWN AND C. W. GEAR, DOCUMENTATION FOR DFASUB—*A program for the solution of simultaneous implicit differential and nonlinear equations*, UIUCDCS-R-73-575, University of Illinois at Urbana-Champaign, 1973.

[4] T. RÜBNER-PETERSON, *An efficient algorithm using backward time-scaled differences for solving stiff differential algebraic systems*, Institute of Circuit Theory and Telecommunication, Technical University of Denmark, 2800 Lyngby, 1973.

[5] A. C. HINDMARSH AND J. F. PAINTER, Private communication.

[6] J. W. STARNER, *A numerical algorithm for the solution of implicit algebraic-differential systems of equations*, Tech. Rep. 318, Dept. of Mathematics and Statistics, Univ. of New Mexico, May 1976.

[7] A. R. CURTIS, *The* FACSIMILE *numerical integrator for stiff initial value problems*, Harwell Report AERE-R, 9352, 1978.

[8] F. R. GANTMACHER, *The Theory of Matrices*, Vol. 2, Chelsea, New York, 1964.

[9] R. SACKS-DAVIS, *Error estimates for a stiff differential equation procedure*, Math. Comp., 31 (1977), pp. 939–953.

[10] L. F. SHAMPINE, *Implementation of implicit formulas for the solution of ODE*'s, this Journal, 1 (1980), pp. 103–118.

[11] 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.

[12] C. W. GEAR AND K. W. TU, *The effect of variable mesh size on the stability of multistep methods*, SIAM J. Numer. Anal., 11 (1974), pp. 1025–1043.

[13] C. W. GEAR AND D. S. WATANABE, *Stability and convergence of variable order multistep methods*, SIAM J. Numer. Anal., 11 (1974), pp. 1044–1058.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 3, September 1982

ERRATUM: SPLINE INTERPOLATION AND SMOOTHING ON THE SPHERE*

GRACE WAHBA†

Table 1 contains several misprints in lines q [6], q [7] and q [8]. The correct table appears below.

TABLE 1

$$q_m(z)=\int_0^1 (1-h)^m(1-2hz+h^2)^{-1/2}\,dh, \qquad m=0,1,\cdots,10.$$

Key. $q[m]=q_m(z)$, $A=\ln(1+1\sqrt{W})$, $C=2\sqrt{W}$, $W=(1-z)/2$

q[0];	A
q[1];	$2AW - C + 1$
q[2];	$\dfrac{A(12W^2 - 4W) - 6CW + 6W + 1}{2}$
q[3];	$\dfrac{A(60W^3 - 36W^2) + 30W^2 + C(8W - 30W^2) - 3W + 1}{3}$
q[4];	$\dfrac{A(840W^4 - 720W^3 + 72W^2) + 420W^3 + C(220W^2 - 420W^3) - 150W^2 - 4W + 3}{12}$
q[5];	$(A(7560W^5 - 8400W^4 + 1800W^3) + 3780W^4 + C(-3780W^4 + 2940W^3 - 256W^2) - 2310W^3 + 60W^2 - 5W + 6)/30$
q[6];	$(A(27720W^6 - 37800W^5 + 12600W^4 - 600W^3) + 13860W^5 + C(-13860W^5 + 14280W^4 - 2772W^3) - 11970W^4 + 1470W^3 + 15W^2 - 3W + 5)/30$

* This Journal, 2 (1981), pp. 5–16.

† Department of Statistics, University of Wisconsin, Madison, Wisconsin 53706.

```
    q[7];
                      7            6            5           4              6
    (A (360360 W  - 582120 W  + 264600 W  - 29400 W ) + 180180 W

                  6             5             4          3            5          4
 + C (- 180180 W  + 231000 W  - 71316 W  + 3072 W ) - 200970 W  + 46830 W

           3         2
 - 525 W  + 21 W  - 7 W + 15)/105
```

```
    q[8];
                        8              7              6             5            4
    (A (10810800 W  - 20180160 W  + 11642400 W  - 2116800 W  + 58800 W )

                 7              7             6             5            4
 + 5405400 W  + C (- 5405400 W  + 8288280 W  - 3538920 W  + 363816 W )

              6             5            4         3        2
 - 7387380 W  + 2577960 W  - 159810 W  - 840 W  + 84 W  - 40 W + 105)/840
```

```
   q[9];
                         9               8              7              6             5
   (A (61261200 W  - 129729600 W  + 90810720 W  - 23284800 W  + 1587600 W )

                8                 8              7              6             5
+ 30630600 W  + C (- 30630600 W  + 54654600 W  - 29909880 W  + 5104440 W

            4               7              6             5          4        3
- 131072 W ) - 49549500 W  + 23183160 W  - 2903670 W  + 17640 W  - 420 W

        2
+ 72 W  - 45 W + 140)/1260
```

```
   q[10];
                          10              9              8              7
   (A (232792560 W  - 551350800 W  + 454053600 W  - 151351200 W

                6            5                9
+ 17463600 W  - 317520 W ) + 116396280 W

                          9               8               7              6             5
+ C (- 116396280 W  + 236876640 W  - 158414256 W  + 38507040 W  - 2462680 W )

                8               7              6           5         4        3
- 217477260 W  + 127987860 W  - 24954930 W  + 930006 W  + 2940 W  - 180 W

         2
+ 45 W  - 35 W + 126)/1260
```

SIAM J. SCI. STAT, COMPUT.
Vol. 3, No. 4, December 1982

THEORETICAL AND PRACTICAL ASPECTS OF A MULTIGRID METHOD*

P. WESSELING†

Abstract. A multigrid method is described. A novel item is the use of incomplete *LU* decomposition for smoothing. Numerical experiments show that its speed and robustness compare favorably with other multigrid methods. A fairly simple rate of convergence proof is presented.

Key words. multigrid methods, numerical methods for elliptic equations

1. Introduction. A multigrid method will be presented. Numerical experiments show that this method is faster than other multigrid methods for which numerical experiments had been reported in sufficient detail when this paper was written (Brandt (1977), Hackbusch (1978), Nicolaides (1979)). The method to be described is also robust in the sense that it does not need to be adapted to the problem at hand, does not require "tuning", and converges fast for a large variety of problems.

A difficulty with the development of multigrid methods is that there are many ways in which the basic ideas underlying these methods can be implemented. The way in which this is done has great influence on the speed and robustness of the method obtained. In order to justify as far as possible some of the choices made here (notably, incomplete *LU* smoothing and Galerkin approximation on coarser grids) some theoretical background material will be presented, including a rate of convergence proof, and some comparative experiments are described.

It is assumed that the reader has some familiarity with multigrid methods (cf. Brandt (1977)).

2. Description of the method. The method to be presented is used to solve the algebraic system that results from finite difference approximation of the following elliptic partial differential equation, denoted in Cartesian tensor notation as follows:

$$-(a_{ij}u_{,i})_{,j}-(b_iu)_{,i}+cu=f, \tag{2.1}$$

with a_{ij}, b_i, c, f and u functions of two variables x_1, x_2 with $(x_1, x_2)\in\Omega=(0,1)\times(0,1)$.

A computational grid Ω^l and a corresponding set of grid functions U^l are defined by

$$\Omega^l\equiv\{(x_1,x_2)|x_i=m_i\cdot 2^{-l}, m_i=0(1)2^l\},\qquad U^l\equiv\{u^l:\Omega^l\to\mathbb{R}\}. \tag{2.2}$$

Equation (2.1) is approximated by a finite difference scheme, for example,

$$-\tfrac{1}{2}(\nabla_i^l a_{ij}\Delta_j^l+\Delta_i^l a_{ij}\nabla_j^l)u^l-\tfrac{1}{2}(\nabla_i^l+\Delta_i^l)(b_iu^l)+cu^l=f^l, \tag{2.3}$$

with $u^l, f^l\in U^l$. The operators ∇_i^l and Δ_i^l are backward and forward difference operators in the x_i-direction. In singular perturbation problems one may wish to approximate b_iu_i by one-sided differences; this does not affect the performance of the multigrid method to be described. The linear algebraic system to which the difference scheme together with the (as yet unspecified) boundary conditions give rise will be denoted by

$$A^lu^l=f^l, \tag{2.4}$$

with $A^l: U^l\to U^l$.

* Received by the editors October 30, 1980.

† Department of Mathematics, Delft University of Technology, Julianalaan 132, Delft, the Netherlands.

The multigrid method makes use of a hierarchy of computational grids Ω^k and corresponding sets of grid functions U^k, $k = l-1(-1)1$, defined by (2.2) with l replaced by k. The step size on Ω^k is 2^{-k}, and as k decreases Ω^k gets coarser. On the coarser grids (2.4) is approximated by

$$A^k u^k = f^k, \qquad k = l-1(-1)1. \tag{2.5}$$

A^k and f^k can be chosen in many ways. Several alternatives will be discussed in the sequel. The choice that is made has a significant influence on the performance of the multigrid method.

Furthermore, let there be given restriction operators r^k and prolongation operators p^k,

$$r^k : U^k \to U^{k-1}, \qquad p^k : U^{k-1} \to U^k. \tag{2.6}$$

Examples will follow.

With the foregoing definitions, multigrid methods for linear partial differential equations can be described in general by the following quasi-Algol program:

```
ALGORITHM 1.
Procedure multigrid method ($k$), value $k$, integer $k$;
begin integer $n$;
    if $k = 1$ then
    begin for $n := 1(1)\, sa[k] + sb[k]$ do $S\,(k, u^k, A^k, f^k)$; end else
    begin for $n := 1(1)\, sa[k]$ do $S(k, u^k, A^k, f^k)$;
        $f^{k-1} := r^k(f^k - A^k u^k)$; $u^{k-1} := 0$;
        for $n := 1(1)\, sc[k]$ do multigrid method $(k-1)$;
        $u^k := u^k + p^k u^{k-1}$;
        for $n := 1(1)\, sb[k]$ do $S\,(k, u^k, A^k, f^k)$
    end
end multigrid method;
```

Here S, to be called the smoothing operator, is some iterative method with a smoothing effect on the error. The choice of S has an important influence on the efficiency of the multigrid method. Various possibilities will be discussed. Note that f^{k-1} is a coarse grid approximation, not to f^k, but to $f^k - A^k u^k$. The following statements,

```
initialize $u^l$;
for $n := 1(1)N$ do multigrid method $(l)$;
```

result in the execution of N multigrid iterations. By special choices of the integer arrays sa, sb, $sc[k]$, every multigrid strategy that has been proposed for linear problems in the literature can be recovered.

For the prolongation operator p^k the following two possibilities will be considered, both exact only for linear functions:

7-point prolongation:

$$\begin{aligned} &(p^k u^{k-1})_{2s,2t} = u^{k-1}_{s,t}, \qquad (p^k u^{k-1})_{2s+1,2t} = \tfrac{1}{2}(u^{k-1}_{st} + u^{k-1}_{s+1,t}), \\ &(p^k u^{k-1})_{2s,2t+1} = \tfrac{1}{2}(u^{k-1}_{st} + u^{k-1}_{s,t+1}), \\ &(p^k u^{k-1})_{2s+1,2t+1} = \tfrac{1}{2}(u^{k-1}_{s+1,t} + u^{k-1}_{s,t+1}) \end{aligned} \tag{2.7}$$

where the value of the grid function u^k in the grid point with coordinates $(s \cdot 2^{-k}, t \cdot 2^{-k})$ is denoted by u^k_{st}.

9-point prolongation: Like 7-point prolongation, except

$$(p^k u^{k-1})_{2s+1,2t+1} = \tfrac{1}{4}(u^{k-1}_{st} + u^{k-1}_{s+1,t} + u^{k-1}_{s,t+1} + u^{k-1}_{s+1,t+1}). \tag{2.8}$$

The following three options will be studied for the restriction operator:
injection:

$$(r^k u^k)_{st} = u^k_{2s,2t}, \tag{2.9}$$

7-point restriction:

$$(r^k u^k)_{st} = \tfrac{1}{4}u^k_{2s,2t} + \tfrac{1}{8}(u^k_{2s+1,2t} + u^k_{2s,2t+1} + u^k_{2s-1,2t} + u^k_{2s,2t-1} + u^k_{2s+1,2t-1} + u^k_{2s-1,2t+1}), \tag{2.10}$$

9-point restriction:

$$(r^k u^k)_{st} = \tfrac{1}{4} u^k_{2s,2t} + \tfrac{1}{8}(u^k_{2s+1,2t} + u^k_{2s,2t+1} + u^k_{2s-1,2t} + u^k_{2s,2t-1}) + \tfrac{1}{16}(u^k_{2s+1,2t+1} + u^k_{2s-1,2t+1} + u^k_{2s+1,2t-1} + u^k_{2s-1,2t-1}). \tag{2.11}$$

On U^k the following inner product is defined:

$$(u^k, v^k)_k \equiv 4^{-k} \sum_{s,t} u^k_{st} v^k_{st}. \tag{2.12}$$

Between p^k and r^k we have the following special relation, for (2.7) and (2.10) or for (2.8) and 2.11):

$$r^k = (p^k)^T, \tag{2.13}$$

i.e.,

$$(r^k u^k, v^{k-1})_{k-1} = (u^k, p^k v^{k-1})_k \quad \forall u^k \in U^k, \quad \forall v^{k-1} \in U^{k-1}. \tag{2.14}$$

For r^k one could think of many possible weighted averages of neighboring function values; it is useful that (2.13) selects a few distinguished cases, which will turn out to be advantageous both theoretically and practically. The operators (2.7) and (2.10) have not yet been discussed in the literature; we will find them to be preferable to the other possibilities.

For the operators $A^k (k < l)$ two possibilities will be studied:

(2.15) *finite difference approximation*,

for example, (2.3) with l replaced by k; and
Galerkin approximation

$$A^{k-1} \equiv (p^k)^T A^k p^k, \qquad k = l(-1)2. \tag{2.16}$$

The reason for the appellation "Galerkin approximation" will become clear in the sequel. Equation (2.15) has been used by Fedorenko (1962), (1964), Bakhvalov (1966) and Brandt (1977), (1979); equation (2.16) has been used by Frederickson (1975), Hackbusch (1978), Wesseling and Sonneveld (1980) and Wesseling (1980). An important property of (2.16) is that if A^l is a 7-point operator (i.e., corresponding to a 7-point difference molecule), as for (2.3) for example, and p^k is defined by (2.7) or (2.8), then A^k $(k < l)$ are 7- or 9-point operators respectively (cf. Fig. 2.1).

A discussion of the smoothing operator S is deferred to § 4.

In order to provide some insight into the question of which of the various options outlined above is preferable, some theoretical framework is developed in the next two sections.

```
  **          **          ***
 ***         ***          ***
  **          **          ***
```

A^l $\quad$ A^k, (2.7) $\quad$ A^k, (2.8)

FIG. 2.1. *Difference molecules for A^l and A^k, $k = 1(1)l-1$.*

3. Computational complexity of multigrid methods. First, a bound will be derived for the computational complexity of one multigrid iteration as defined by the procedure in the preceding section. Let $\max_k (sa[k], sb[k])$ be independent of l. The total computational complexity of the operations S and the computation of f^{k-1} and $u^k + p^k u^{k-1}$ for one execution of the statement: multigrid method (k) can be bounded by $a \cdot 4^k$, where a is some constant. Let $\max_k (sc[k]) \leqq \sigma$. Let the total computational complexity of one execution of the statement: multigrid method (k) be denoted by W_k; then

$$W_k \leqq \sigma W_{k-1} + a \cdot 4^k. \tag{3.1}$$

W_k is bounded from above by the solution of the following difference equation: $z_k = \sigma z_{k-1} + a \cdot 4^k$. Hence

$$\begin{aligned} &W_l \leqq W_1 \sigma^{l-1} + a\left(\frac{4^{l+1} - \sigma^{l+1}}{4-\sigma} - \frac{4}{\sigma+1}\sigma^l\right), \qquad \sigma \neq 4, \\ &W_l \leqq 4^l \frac{W_1}{4 + a(l-1)}, \qquad \sigma = 4. \end{aligned} \tag{3.2}$$

Therefore $W_l = O(4^l), \sigma < 4$; $W_l = O(l4^l), \sigma = 4$; $W_l = O(\sigma^l), \sigma > 4$.

The overall computational complexity depends on the accuracy required and on the rate of convergence of the multigrid method. The following authors have shown theoretically that for various special cases various specific multigrid algorithms have a rate of convergence that is independent of l (eq. numbers are for this paper):

Fedorenko (1964): Poisson equation; A^k: (2.15); p^k: (2.8); r^k: (2.9).
Bakhvalov (1966): (2.3); A^k: (2.15); p^k: must be exact for second degree polynomials; r^k: (2.9).
Frederickson (1975): Poisson equation; A^k: (2.16); p^k: (2.8); r^k: (2.11).
Wesseling (1980): (2.3); A^k:(2.16); p^k:(2.8); r^k: (2.11).

A very general rate of convergence proof in an abstract setting admitting arbitrary regions and elliptic equations of higher order has been given Hackbusch (1980).

The rate of convergence proofs referred to above are rather involved and technical. Therefore it seems useful to present here an elementary proof for the selfadjoint and positive definite case (i.e., (2.3) with $b_i = c = 0$), using only arguments that can be extended to the general case. It is assumed that A^k is given by (2.16); for (2.15) the proof would seem to be more difficult, as evidenced by the use of higher order interpolation in the theory of Bakhvalov (1966). A homogeneous Dirichlet boundary condition is assumed.

The most difficult part of the proof is to estimate how well A^{k-1} approximates A^k, i.e., to estimate $u^k - p^k u^{k-1}$ with $A^k u^k = f^k$ and $A^{k-1} u^{k-1} = r^k f^k$. We proceed to do this.

Define

$$V^k \equiv \{u^k : \Omega^k \to \mathbb{R} | u^k | \partial\Omega^k = 0\}. \tag{3.3}$$

Whenever in some formula a function value on $\partial\Omega^k$ or outside Ω^k is needed, this value is taken as zero. When there is no danger of confusion the indices k and l will be deleted for brevity. On V the following norms are defined:

$$\|u\|_0^2 \equiv (u, u), \quad \|u\|_1^2 \equiv (\nabla_i u, \nabla_i u), \quad \|u\|_2^2 \equiv (\nabla_i \Delta_j u, \nabla_i \Delta_j u), \tag{3.4}$$
$$\|u\|_{-1} \equiv \sup_{\|v\|_1 \leqq 1} |(u, v)|,$$

where the summation convention applies. These quantities are indeed norms because of the boundary condition, and are analogous to Sobolev norms. Without mentioning this explicitly, we will often use the following partial summation formula:

$$(u, \nabla_i v) = -(\Delta_i u, v). \tag{3.5}$$

The following well-known inequalities hold:

$$\|u\|_{-1} \leqq \|u\|_0 \leqq \|u\|_1 \leqq \|u\|_2. \tag{3.6}$$

Furthermore, it is easy to derive that

$$\|u^k\|_{i,k} \leqq 2^k \sqrt{8}\, \|u^k\|_{i-1,k}, \qquad i = 1, 2. \tag{3.7}$$

The prolongation and restriction operators p^k and r^k are to have the following properties. Define $p^{mk} \equiv p^m p^{m-1} \cdots p^{k+1}$.

$$C_1\|u^k\|_{1,k} \leqq \|p^{mk}u^k\|_{1,m} \leqq \|u^k\|_{1,k}, \qquad m > k, \tag{3.8a}$$

$$\|u^k - p^k r^k u^k\|_{1,k} \leqq C_2 2^{-k} \|u^k\|_{2,k}, \tag{3.8b}$$

$$C_3\|u^{k-1}\|_{0,k-1} \leqq \|p^k u^{k-1}\|_{0,k} \leqq \|u^{k-1}\|_{0,k-1}, \tag{3.8c}$$

$$\|u^k - r^{k+1}p^{k+1}u^k\|_{-1,k} \leqq C_4 4^{-k} \|u^k\|_{1,k}, \tag{3.8d}$$

$$\|r^k u^k\|_{2,k-1} \leqq \|u^k\|_{2,k}. \tag{3.8e}$$

C_i denote positive constants independent of k and l.

Define $B(u, v) \equiv (Au, v)$. Assume uniform ellipticity and boundedness for (2.3):

$$C_5 \xi_i \xi_i \leqq a_{ij}\xi_i\xi_j, \qquad a_{ij} = a_{ji}, \quad \forall \xi_i \in \mathbb{R}, \quad \forall x \in \Omega. \tag{3.9}$$

We have

$$B^l(u^l, v^l) = \tfrac{1}{2}(a_{ij}\Delta_i^l u^l, \Delta_j^l v^l)_l + \tfrac{1}{2}(a_{ij}\nabla_i^l u^l, \nabla_j^l v^l)_l. \tag{3.10}$$

Hence

$$B^l(u^l, v^l) = B^l(v^l, u^l), \tag{3.11}$$

$$B^l(u^l, u^l) \geqq C_5 \|u^l\|_{1,l}^2. \tag{3.12}$$

Furthermore, with the inequality of Schwarz,

$$(a_{ij}\Delta_i u, \Delta_j v) \leqq \sup_{i,j,\Omega} |a_{ij}| \sum_{i,j} \|\Delta_i u\|_0 \|\Delta_j v\|_0 \leqq 2C_6 \|u\|_1 \|v\|_1, \tag{3.13}$$

since from (3.9): $\sup_{i,j,\Omega} |a_{ij}| \leqq C_6$. It follows that

$$|B^l(u^l, v^l)| \leqq 2C_6 \|u^l\|_{1,l} \|v^l\|_{1,l}. \tag{3.14}$$

It will be shown that if A^k is constructed according to (2.16) and if p^k satisfies (3.8a), then $B^k(u^k, v^k)$ also satisfies (3.11), (3.12) and (3.14) (with other constants).

We have $B^{k-1}(u^{k-1}, v^{k-1}) = ((p^k)^T A^k p^k u^{k-1}, v^{k-1})_{k-1} = (A^k p^k u^{k-1}, p^k v^{k-1})_k = B^k(p^k u^{k-1}, p^k v^{k-1})$. Hence:

$$B^k(u^k, v^k) = B^l(p^{lk} u^k, p^{lk} v^k). \tag{3.15}$$

With (3.8a) it follows that for $k = 1(1)l$

$$B^k(u^k, v^k) = B^k(v^k, u^k), \tag{3.16}$$

$$B^k(u^k, u^k) \geqq C_1^2 C_5 \|u^k\|_{1,k}^2, \tag{3.17}$$

$$|B^k(u^k, v^k)| \leqq 2C_6 \|u^k\|_{1,k} \|v^k\|_{1,k}. \tag{3.18}$$

We are now in a position to show that A^k is a Galerkin approximation to A^m, $m > k$, and to derive error estimates. Define, for arbitrary (deleted) k,

$$a(u, v) \equiv B(u, v) - 2(b, v) \quad \forall u, v \in V, \quad b \in V \text{ fixed.} \tag{3.19}$$

In the standard way it can be shown that $Au = b$ and $a(u, u) = \inf_{v \in V} a(v, v)$ are equivalent. Furthermore, $a^k(u^k, v^k) = a^m(p^{mk} u^k, p^{mk} v^k)$, $m > k$. Let $A^m u^m = b^m$ and $A^k u^k = (p^{mk})^T b^m$. We have

$$a^k(u^k, u^k) = \inf_{v^k \in V^k} a^k(v^k, v^k) = \inf_{v^k \in V^k} a^m(p^{mk} v^k, p^{mk} v^k). \tag{3.20}$$

Hence, the designation of (2.16) as a Galerkin approximation is justified. Using (3.16)–(3.18), we derive the following result similarly to its equivalent in finite element theory:

$$\|u^m - p^{mk} u^k\|_{1,k} \leqq D_1 \inf_{v^k \in V^k} \|u^m - p^{mk} v^k\|_{1,k}, \tag{3.21}$$

with $D_1 \equiv (2C_6/C_1^2 C_5)^{1/2}$. This estimate is not quite what we need, because in the rate of convergence proof an estimate of the following kind will be required: $\|u^k - p^k u^{k-1}\|_{0,k} = O(4^{-k})$. This can be obtained as follows, using what is known in the finite element literature as Nitsche's trick; and specializing (3.21) to the case $m := k$, $k := m - 1$. Define $w^k \in V^k$ by

$$A^k w^k = u^k - p^k u^{k-1}; \tag{3.22}$$

then $\|u^k - p^k u^{k-1}\|_{0,k}^2 = B^k(w^k, u^k - p^k u^{k-1})$. Since $B^k(p^k u^{k-1}, p^k v^{k-1}) = (b^k, p^k v^{k-1})_k$ for all $v^{k-1} \in V^{k-1}$ and $B^k(u^k, v^k) = (b^k, v^k)_k$ for all $v^k \in V^k$, we have $B^k(u^k - p^k u^{k-1}, p^k v^{k-1}) = 0$. Hence

$$\|u^k - p^k u^{k-1}\|_{0,k}^2 = B^k(w^k - p^k v^{k-1}, u^k - p^k u^{k-1}) \quad \forall v^{k-1} \in V^{k-1}. \tag{3.23}$$

Using (3.18) and (3.21), we obtain

$$\|u^k - p^k u^{k-1}\|_{0,k}^2 \leqq 2C_6 D_1 \|w^k - p^k v^{k-1}\|_{1,k} \inf_{z^{k-1} \in V^{k-1}} \|u^k - p^k z^{k-1}\|_{1,k}, \tag{3.24}$$

for all $v^{k-1} \in V^{k-1}$. Choosing $v^{k-1} = r^k w^k$, $z^{k-1} = r^k u^k$, we find with (3.8b)

$$\|u^k - p^k u^{k-1}\|_{0,k}^2 \leqq 2C_6 D_1 C_2^2 \cdot 4^{-k} \|w^k\|_{2,k} \|u^k\|_{2,k}. \tag{3.25}$$

In Appendix A it will be shown that

$$\|v^k\|_{2,k} \leqq D_2 \|A^k v^k\|_{0,k} \quad \forall v^k \in V^k. \tag{3.26}$$

Using (3.26) and (3.22) we finally obtain

$$\|u^k - p^k u^{k-1}\|_{0,k} \leqq D_3 \cdot 4^{-k} \|b^k\|_{0,k}, \qquad D_3 \equiv 2C_6 D_1 C_2^2 D_2^2. \tag{3.27}$$

This estimate of the quality of the coarse grid approximations is one ingredient needed in the rate of convergence proof to be presented. We also need some information concerning the smoothing operator S.

Using (3.7) and (3.18), we have

(3.28) $$\frac{(A^k u^k, u^k)_k}{(u^k, u^k)_k} \leqq D_4 \cdot 4^k, \qquad D_4 \equiv 16C_6.$$

Hence

(3.29) $$\lambda(A^k) \in (0, D_4 \cdot 4^k).$$

For purposes of illustration only, the smoothing operator S is chosen as follows (Jacobi iteration):

(3.30) $$u^{k,\nu+1} = u^{k,\nu} - \alpha(A^k u^{k,\nu} - f^k),$$

with α a parameter to be specified shortly.

The rate of convergence of the multigrid method of § 2 will be estimated for the special case $sa[k]=0$, $sb[k]=m$, $sc[k]=\sigma$. Numerical experiments to be described show that this is a profitable multigrid strategy. More generality would merely add to the technicalities of the arguments to be presented.

Let $u^{k,0}$, $u^{k,1/2}$ and $u^{k,1}$ denote the iterand just before the execution of the statement, multigrid method (k), just before the $sb[k]=m$ executions of the smoothing operation S, and just after the execution of multigrid method (k), respectively. Define

(3.31) $$\varepsilon^{k,\mu} \equiv u^{k,\mu} - u^k, \qquad u^k \equiv (A^k)^{-1} f^k, \qquad \mu = 0, \tfrac{1}{2}, 1.$$

Then

(3.32) $$\varepsilon^{k,1} = (I^k - \alpha A^k)^m \varepsilon^{k,1/2},$$

with I^k the identity operator. Define

(3.33) $$V_1^{k,\gamma} \equiv \{\text{span of eigenvectors of } A^k \text{ belonging to eigenvalues} \in (0, \gamma D_4 \cdot 4^k), 0<\gamma<1\},$$

(3.34) $$V_2^{k,\gamma} \equiv \text{the orthogonal complement of } V_1^{k,\gamma} \text{ in } V^k.$$

Let $\varepsilon^{k,\mu} = \varepsilon_1^{k,\mu} + \varepsilon_2^{k,\mu}$ with $\varepsilon_i^{k,\mu} \in V_i^{k,\gamma}$, $i=1,2$. Choose

(3.35) $$\alpha = \frac{4^{-k}}{D_4};$$

then

(3.36) $$\|\varepsilon_1^{k,1}\|_{0,k} \leqq \|\varepsilon_1^{k,1/2}\|_{0,k}, \qquad \|\varepsilon_2^{k,1}\|_{0,k} \leqq (1-\gamma)^m \|\varepsilon_2^{k,1/2}\|_{0,k}.$$

Because of (3.27) we have

(3.37) $$\|\varepsilon^{k,0} + p^k u^{k-1}\|_{0,k} \leqq D_3 \cdot 4^{-k} \|A^k \varepsilon^{k,0}\|_{0,k}.$$

Let the result of the coarse grid correction (computed by $sc[k]=\sigma$ executions of the statement multigrid method (k)) be denoted by v^{k-1}, and assume

(3.38) $$\|v^{k-1} - u^{k-1}\|_{0,k-1} \leqq \delta_{k-1} \|u^{k-1}\|_{0,k-1},$$

with δ_{k-1} a constant to be determined later. Then

(3.39) $$\|\varepsilon^{k,1/2}\|_{0,k} \leqq D_3 \cdot 4^{-k} \|A^k \varepsilon^{k,0}\|_{0,k} + \delta_{k+1} \|u^{k-1}\|_{0,k-1}.$$

With (3.37) $p^k u^{k-1}$ can be bounded , and with (3.8c) one obtains

(3.40) $$\|\varepsilon^{k,1/2}\|_{0,k} \leqq D_3 \cdot 4^{-k}\left(\frac{1+\delta_{k-1}}{C_3}\right)\|A^k \varepsilon^{k,0}\|_{0,k} + \frac{\delta_{k-1}}{C_3}\|\varepsilon^{k,0}\|_{0,k}.$$

From (3.36) it follows that

(3.41) $$\|\varepsilon^{k,1}\|_{0,k} \leqq (\xi + \eta\delta_{k-1})\|\varepsilon^{k,0}\|_{0,k},$$

with $\xi \equiv D_3(\gamma + (1-\gamma)^m)$, $\eta \equiv (1+\xi)/C_3$. Hence

(3.42) $$\|u^{k,1} - u^k\|_{0,k} \leqq (\xi + \eta\delta_{k-1})\|u^k\|_{0,k},$$

and therefore,

(3.43) $$\delta_k \leqq (\xi + \eta\delta_{k-1})^\sigma.$$

In order to simplify this proof we assume $\delta_1 = 0$, i.e., on the coarsest grid Ω^1 the solution is to be calculated exactly, rather than approximately as in the procedure multigrid method of § 2. Since $\eta > 1$ we will not be able to show that $\delta_k < 1$ if $\sigma = 1$, although the method works well for $\sigma = 1$, as we shall see. Choose $\sigma > 1$. Assume (induction) that $\delta_{k-1} \leqq \xi$, and choose m and γ such that

(3.44) $$\xi^{\sigma-1}(1+\eta)^\sigma < 1,$$

(hence $\xi < 1$); then

(3.45) $$\delta_k \leqq \xi.$$

Thus we have proved the following theorem.

THEOREM. *Let the multigrid method of* § 2 *satisfy* $sa[k] = 0$, $sb[k] = m$, $sc[k] = \sigma > 1$ *and equations* (2.16), (3.8) *and* (3.44), *and let the smoothing operator S be defined by* (3.30) *and* (3.35). *Then one execution of the statement multigrid method* (l) *reduces the error by a factor* $\xi < 1$, *for equation* (2.3) *with* $b_i = c = 0$ *and* a_{ij} *satisfying* (3.9).

Because ξ is independent of l, the number of multigrid iterations required for a given precision is fixed. Choosing $\sigma < 4$ and taking (3.2) into account, we conclude that the computational complexity of the multigrid method is $O(4^l)$.

As an example, we show that the conditions of the theorem, i.e., (3.8) and (3.44), are satisfied with p^k and r^k defined by (2.7), (2.8), (2.10), (2.11). The right-hand side of (3.8a) follows from the easily verifiable fact that $\|\nabla_i^k p^k u^{k-1}\|_{0,k} \leqq \|\nabla_i^{k-1} u^{k-1}\|_{0,k-1}$. The left-hand side can be checked by employing the following identity, valid for 9-point interpolation:

$$\|\nabla_1^m p^{mk} u^k\|_{0,m}^2 = 2^{k-m}\|\nabla_1^k u^k\|_{0,k}^2 + 2^{k-m-2}\sum_{i,j=0}^{2^k}\sum_{p=1}^{n-1}\left(\frac{2n-p}{n}w_{ij}^k + \frac{p}{n}w_{i,j+1}^k\right)^2,$$

with $w^k = \nabla_1^k u^k$, $n = 2^{m-k-1}$; and by using the inequality $\sum(a_i+b_i)^2 \geqq \sum(a_i^2+b_i^2) - 2(\sum a_i^2)^{1/2}(\sum b_i^2)^{1/2}$. The case of 7-point interpolation can be treated similarly. It is easy to write $\nabla_i(u^k - p^k r^k u^k)$ as a linear combination of quantities of the type $\nabla_m \Delta_n u^k$, such that (3.8b) follows. The derivation of (3.8c) is straightforward. There are constants C_1 and C_2 such that $u^k - r^{k+1}p^{k+1}u^k = C_1 \cdot 4^{-k}\nabla_i^k \Delta_i^k u^k + C_2 \cdot 16^{-k}\nabla_1^k \Delta_1^k \nabla_2^k \Delta_2^k u^k$; using this (3.8d) is easy to derive. Equation (3.8e) is trivial. Finally, choose $\gamma = z/(2D_3)$ and $m > \log(z/2D_3)/\log(1-\gamma)$ with z to be determined; then $\xi \leqq z$. Choose $z = (1+2/C_3)^{-\sigma}$; then (3.44) is satisfied.

4. Smoothing. This section is devoted to a discussion of possible smoothing operators (procedure S in Algorithm 1). A useful yardstick by which the merits of various smoothing operators may be measured is the smoothing factor introduced by Brandt (1977). The error $\varepsilon^k \in U^k$ before application of a multigrid iteration can be represented by a Fourier series as follows (deleting the superscript k):

$$\varepsilon_{mn} = \sum_{s,t=-M}^{M}\sum c_{st} \exp(im\theta_s + in\phi_t), \tag{4.1}$$

with $\theta_s = (2s-1)\pi/2M+1)$, $\phi_t = (2t-1)\pi/(2M+1)$, $M = 2^{k-1}$. For periodic boundary conditions and constant coefficients in the differential equation, many smoothing operators have the property that the error after application of the smoothing operator $\hat{\varepsilon}_{mn}$ satisfies

$$\hat{\varepsilon}_{mn} = \sum_{s,t=-M}^{M}\sum \hat{c}_{st} \exp(im\theta_s + in\phi_t), \tag{4.2}$$

with

$$\hat{c}_{st} = \rho(\theta_s, \phi_t)c_{st}. \tag{4.3}$$

The philosophy underlying multigrid methods requires that those Fourier components which cannot be resolved on coarser grids be eliminated as fast as possible. One therefore defines the smoothing factor $\bar{\rho}$ as follows: (cf. Brandt (1977)):

$$\bar{\rho} \equiv \sup_{(\theta,\phi)\in G} |\rho(\theta, \phi)|, \qquad G \equiv \left\{(\theta, \phi)\,\middle|\, -\pi \leqq \theta, \phi \leqq \pi, |\theta| \geqq \frac{\pi}{2} \text{ or } |\phi| \geqq \frac{\pi}{2}\right\}. \tag{4.4}$$

For convenience, G is not restricted to the discrete set of values occurring in (4.1) and (4.2). Of two smoothing operators, the one with the smaller $\bar{\rho}$ (for a given amount of computational work) is judged to be the better of the two.

Two smoothing operators that are often used in multigrid methods are pointwise and line Gauss–Seidel iteration (Brandt (1977), (1979), Nicolaides (1979)). Hackbusch (1978) uses pointwise and line Gauss–Seidel iteration with a special ordering of the grid points. Wesseling and Sonneveld (1980) have introduced incomplete LU (ILU) decomposition as a smoothing operator for multigrid methods.

In order to describe ILU decomposition, some notation will be introduced. The operator A^k is denoted as follows, under the summation convention and deleting the superscript k:

$$(Au)_{ij} = \sigma_{\alpha\beta}u_{i+\alpha,j+\beta}, \tag{4.5}$$

with $\alpha, \beta \in \{-1, 0, 1\}$. If the coefficients in the differential equation are variable, one must keep in mind that $\sigma_{\alpha\beta}$ depends on i and j. The difference molecules that (4.5) may represent are depicted in Fig. 2.1.

Incomplete LU decomposition is defined as follows:

$$\begin{gathered}(Lu)^{ij} = \lambda_{\alpha\beta}u_{i+\alpha,j+\beta}, \quad (Uu)_{ij} = \mu_{\alpha\beta}u_{i+\alpha,j+\beta}, \\ LU = A + C, \qquad (Cu)_{ij} = \gamma_{\alpha\beta}u_{i+\alpha,j+\beta},\end{gathered} \tag{4.6}$$

with the elements of the error matrix C zero in those places where the elements of A are nonzero. The ILU decompositions that will be used are fully defined in Table 4.1, which lists the values of (α, β) for which $\lambda_{\alpha\beta} \neq 0$, $\gamma_{ab} \neq 0$, for what will be called 5-point, 7-point and 9-point ILU decomposition. Furthermore, $\mu_{\alpha\beta} \neq 0$ if and only if $\lambda_{-\alpha,-\beta} \neq 0$.

TABLE 4.1

	λ	γ
5-point LU	(0, 0) (0, −1) (−1, 0)	(−1, 1) (1, −1)
7-point LU	(0, 0) (0, −1) (−1, 0) (1, −1)	(−2, 1) (2, −1)
9-point LU	(0, 0) (0, −1) (−1, 0) (1, −1) (−1, −1)	(−2, 1) (2, −1)

L and U may be constructed by a standard LU-decomposition algorithm, with zero outside the nonzero patterns described above. A version in which $l_{ii} = u_{ii}$ may be computed as follows (cf. Wesseling and Sonneveld (1980)). P denotes the nonzero pattern of the ILU decomposition. A is an $N \times N$ matrix.

```
ALGORITHM 2.
$A^0 := A$;
for $r := 1(1)N$ do
begin $a^r_{rr} :=$ sqrt($a^{r-1}_{rr}$);
    for $j > r \wedge (r,j) \in P$ do $a^r_{rj} := a^{r-1}_{rj}/a^r_{rr}$;
    for $i > r \wedge (i,r) \in P$ do $a^r_{ir} := a^{r-1}_{ir}/a^r_{rr}$;
    for $(i,j) \in P \wedge i > r \wedge j > r \wedge (i,r) \in P \wedge (r,j) \in P$ do $a^r_{ij} := a^{r-1}_{ir} - a^r_{ir}a^r_{rj}$
end ILU decomposition;
```

A^N will contain L and U.

For a discussion of smoothing factors, we will construct 5-point and 7-point ILU decompositions for the constant coefficient case. The 9-point ILU decomposition equals the 7-point decomposition, unless A is a 9-point difference operator, which is never the case on the finest grid. Since the bulk of the computational work takes place on the finest grid, we will not study 9-point ILU decompositions.

In order to make the ILU decomposition unique, we (arbitrarily) require

$$\lambda_{00} = \mu_{00} \tag{4.7}$$

(as in Algorithm 2). The following equations result from (4.6):

$$\begin{aligned} &\sigma_{0,-1} = \lambda_{0,-1}\mu_{00}, \qquad \sigma_{1,-1} = \lambda_{0,-1}\mu_{10} + \lambda_{1,-1}\mu_{00},\\ &\sigma_{-1,0} = \lambda_{0,-1}\mu_{-1,1} + \lambda_{-1,0}\mu_{00}, \qquad \sigma_{00} = \lambda_{-\alpha,-\beta}\mu_{\alpha\beta},\\ &\sigma_{10} = \lambda_{1,-1}\mu_{01} + \lambda_{00}\mu_{10}, \qquad \sigma_{-1,1} = \lambda_{-1,0}\mu_{01} + \lambda_{00}\mu_{-1,1},\\ &\sigma_{01} = \lambda_{00}\mu_{01}. \end{aligned} \tag{4.8}$$

For the case of the Poisson equation, the system (4.8) is solved as follows. Let $L^T = U$, i.e., $\lambda_{\alpha\beta} = \mu_{-\alpha,-\beta}$. Eliminating μ_{01} with the first equation, $\mu_{-1,1}$ with the second and μ_{10} with the third, one finds, with $\mu \equiv \mu_{00}$, that $\mu_{01} = -1/\mu$, $\mu_{-1,1} = \mu/(1-\mu^4)$, $\mu_{10} = \mu^3/(1-\mu^4)$, and

$$\mu^{12} - 4\mu^{10} + 8\mu^6 - 4\mu^2 + 1 = 0. \tag{4.9}$$

In order to get $\gamma_{\alpha\beta}$ as small as possible the largest root is preferred, which is found to be

$$\mu^2 = 3.294168413. \tag{4.10}$$

This defines the 7-point ILU decomposition for the Poisson equation. In the 5-point case we have $\mu_{-1,1}=0$, and the equations in (4.8) in which $\sigma_{-1,1}$ and $\sigma_{1,1}$ occur are dropped. We find $\mu_{01}=\mu_{10}=-1/\mu$, and

A6(4.11) $$\mu^4-4\mu^2+2=0,$$

of which the largest root is

(4.12) $$\mu^2=2+\sqrt{2}.$$

The smoothing factors of these *ILU* decompositions are determined as follows. A smoothing operation is defined by

(4.13) $$x := x+(LU)^{-1}(f-Ax)$$

or $(A+C)x := Cx+f$. One finds

(4.14) $$\rho(\theta,\phi)=\frac{\gamma_{1,-1}\cos(\theta-\phi)}{2-\cos\theta-\cos\phi+\gamma_{1,-1}\cos(\phi-\theta)} \qquad \text{(5-point } ILU\text{)},$$

with $\gamma_{1,-1}=\mu_{01}\mu_{10}=1-\frac{1}{2}\sqrt{2}$,

(4.15) $$\rho(\theta,\phi)=\frac{2\gamma_{2,-1}\cos(2\theta-\phi)}{4-2\cos\theta-\cos\phi+2\gamma_{2,-1}\cos(2\theta-\phi)} \qquad \text{(7-point } ILU\text{)},$$

with $\gamma_{2,-1}=\mu_{-1,1}\mu_{10}\cong 0.11181$. By analytical means it follows that

(4.16) $$\bar{\rho}=\frac{\mu^{-2}}{\sqrt{3-\mu^{-2}}}=0.204 \qquad \text{(5-point } ILU\text{)}$$

(cf. Hemker (1980)). For the 7-point case it has been found numerically that

(4.17) $$\bar{\rho}\cong 0.126 \qquad \text{(7-point } ILU\text{)}.$$

For the Gauss–Seidel smoothing operators, Brandt (1977) has found the following smoothing factors:

(4.18) $$\bar{\rho}=\frac{1}{2} \qquad \text{(pointwise Gauss–Seidel)},$$

(4.19) $$\bar{\rho}=\frac{1}{\sqrt{5}}\cong .447 \qquad \text{(line Gauss–Seidel)}.$$

The smoothing operators proposed by Hackbusch (1978) are not amenable to the Fourier analysis described above, so that smoothing factors cannot be defined.

We will roughly determine the computational complexity of the smoothing operators defined above by counting operations from the set $\{+,-,*,/,\text{sqrt}\}$. We will distinguish between the variable coefficient case and the case of the Poisson equation, to be denoted *general* and *Poisson* case respectively. Equation (4.13) is rewritten as follows:

(4.20) $$(LU)x := Cx+f.$$

For the 5-point *ILU* decomposition we have $(Cx+f)_{ij}=\gamma_{-1,1}x_{i-1,j+1}+\gamma_{1,-1}x_{i+1,j-1}+f_{ij}$ (general), $(Cx+f)_{ij}=\gamma_{1,-1}(x_{i-1,j+1}+x_{i+1,j-1})+f_{ij}$ (Poisson), hence computation of $Cx+f$ takes 4 or 3 operations per grid point respectively. For the 7-point *ILU* decomposition the computational complexity of $Cx+f$ is the same. The solution of a system $Lu=y$ is computed as follows, for the 5-point *ILU* decomposition:

$$u_{i,j} := (y_{ij} - \lambda_{0,-1}u_{i,j-1} - \lambda_{-1,0}u_{i-1,j})/\lambda_{00} \qquad \text{(general)}, \tag{4.21}$$

$$u_{i,j} := (y_{ij} - (u_{i,j-1} + u_{i-1,j})/\mu)/\mu, \qquad \text{(Poisson)}, \tag{4.22}$$

and we count 5 or 4 operations per grid point in the general or Poisson case, respectively. Similarly for the 7-point *ILU*-decomposition, it is found that $Lu=y$ takes 7 or 6 operations per grid point in the general or Poisson case, respectively.

The computational complexity of the Gauss–Seidel smoothing operators is as follows. The pointwise Gauss–Seidel smoothing operation is performed according to:

$$\begin{aligned} u_{ij} := (f_{ij} &- \sigma_{0,-1}u_{i,j-1} - \sigma_{-1,0}u_{i-1,j-1} - \sigma_{0,1}u_{i,j+1} \\ &- \sigma_{1,0}u_{i+1,j} - \sigma_{1,-1}u_{i+1,j-1} - \sigma_{-1,1}u_{i-1,j+1})/\sigma_{0,0} \qquad \text{(general)}, \end{aligned} \tag{4.23}$$

$$u_{ij} := (f_{ij} + u_{i,j-1} + u_{i-1,j} + u_{i,j+1} + u_{i+1,j})/4 \qquad \text{(Poisson)}. \tag{4.24}$$

If A is a 5-point operator, $\sigma_{1,-1} = \sigma_{-1,1} = 0$. The number of operations per grid point is 13, 9 or 5 in the general 7-point, general 5-point or Poisson case, respectively. The line Gauss–Seidel smoothing operation for lines in the x_1-direction is performed according to:

$$\begin{aligned} &\sigma_{-1,0}u_{i-1,j} + \sigma_{0,0}u_{ij} + \sigma_{1,0}u_{i+1,j} \\ &\quad = f_{ij} - \sigma_{0,-1}u_{i,j-1} - \sigma_{0,1}u_{i,j+1} - \sigma_{1,-1}u_{i+1,j-1} - \sigma_{-1,1}u_{i-1,j+1} \qquad \text{(general)}, \end{aligned} \tag{4.25}$$

$$-u_{i-1,j} + 4u_{ij} - u_{i+1,j} = f_{ij} + u_{i,j-1} + u_{i,j-1} \qquad \text{(Poisson)}. \tag{4.26}$$

The right-hand side takes 8, 4 or 2 operations per grid point in the general 7-point, general 5-point or Poisson case, respectively. One can solve one or m identical tridiagonal $n \times n$ systems in about $10n$ or $6nm$ operations respectively (Isaacson and Keller (1966, p. 57)); (by our definition of an operation, our count is twice that of Isaacson and Keller). Hence, solving the tridiagonal systems (4.25), (4.26) takes the following number of operations per grid point: 10, 10 or 6 in the general 7-point, general 5-point or Poisson case, respectively.

The number of operations per grid point for the various smoothing operators is summarized in Table 4.2.

TABLE 4.2.
Operations per grid point for various smoothing operators

	General 7-point	General 5-point	Poisson
5-point *ILU*	14	14	11
7-point *ILU*	18	18	15
G–S by points	13	9	5
G–S by lines	18	14	8

Clearly, on the basis of this table and the smoothing factors, 7-point *ILU* comes out best. The Poisson equation can be considered to be representative for the class of selfadjoint elliptic equations with smoothly varying coefficients of the same order of magnitude. In the next section numerical experiments with other typical cases will also be described.

The above analysis is not exact for *ILU* smoothing, because the coefficients in L and U vary near the boundaries. The decisive test of the merits of the various options lies in numerical experiments.

If the *ILU* decomposition is constructed by means of Algorithm 2, then the computation of L and U takes 9 or 21 operations per grid point in the 5- or 7-point case respectively. If one also takes into account the computation of booleans of the type $(i, j) \in P$, then the total work becomes proportional to 8^l, thus destroying the theoretical $O(4^l)$ computational complexity of the present multigrid method. This can be avoided by computing the *ILU* decomposition by means of the following recursive formulae. Consider the 7-point case. Let, in left to right order, g, f, e, a, b, c, d denote the nonzero diagonals of the given matrix A, and let the subscript k denote the kth element, e.g., a_k is the element in the main diagonal in row k. Similarly, let η, ζ, ε and α denote the nonzero diagonals of L, and $\alpha, \beta, \gamma, \sigma$ those of U. Then we have on an $m \times n$ grid:

$$\eta_k = \frac{g_k}{\alpha_{k-m}}, \quad k \geqq m+1, \qquad \zeta_k = \frac{f_k - \eta_k\beta_{k-m}}{\alpha_{k-m+1}}, \quad k \geqq m;$$

$$\varepsilon_k = \frac{e_k - \eta_k\gamma_{k-m}}{\alpha_{k-1}}, \qquad k \geqq 2;$$

$$\alpha_k = (a_k - \varepsilon_k\beta_{k-1} - \zeta_k\gamma_{k-m+1} - \eta_k\delta_{k-m})^{1/2}, \qquad k \geqq 1; \tag{4.27}$$

$$\beta_k = \frac{b_k - \zeta_k\delta_{k-m+1}}{\alpha_k}, \qquad k \leqq mn-1;$$

$$\gamma_k = \frac{c_k - \varepsilon_k\delta_{k-1}}{\alpha_k}, \quad k \leqq mn-m+1, \qquad \delta_k = \frac{d_k}{\alpha_k}, \quad k \leqq mn-m.$$

Quantities that are not defined are to be replaced by 0. The recursion formulae (4.27) require 21 operations per grid point, just as the quasi-Algol program (Algorithm 1) (neglecting computation of booleans). However, one might prefer the quasi-Algol program because of its flexibility in changing the sparsity pattern P and its ease of generalization to systems of partial differential equations.

An efficient way to compute A^k as defined by (2.16) is given in Appendix B.

5. Numerical experiments. It has already been remarked that multigrid methods in general for linear partial differential equations can be represented by the procedure multigrid method (k) of § 2. It remains to choose $A^k (k < l)$, p^k, r^k, $sa[k]$, $sb[k]$, $sc[k]$ and the smoothing operator S. To judge the merits of all possibilities on purely theoretical grounds seems out of the question, although the fact that (2.16) allows a nicer theory to be developed than does (2.15) induces a preference for (2.16). However, numerical experiments are decisive. Some numerical experiments are described below, and comparison is made with experiments reported by Hackbusch (1978) and Nicolaides (1979). The following cases will be treated, all of them special cases of (2.1):

(5.1) *Case* 1: $u_{,ii} = 4$,

(5.2) *Case* 2: $u_{11} + 0.01u_{,22} = 2.02$,

(5.3) *Case* 3: $u_{,11} + 1.7u_{,12} + u_{,22} = 4$.

For these three cases, which are among the cases treated by Hackbusch, the boundary condition is $u|_{\partial\Omega} = x_ix_i$, exact solution $u = x_ix_i$, starting iterand $u = 0$.

In the experiments to be described, the boundary conditions were not eliminated from the equations.

Table 5.1 shows variants of the multigrid method of § 2 that have been applied to case 1. The columns p, r and A give the numbers of the equations where the prolongation, restriction and coarse grid operators are defined. The column ILU gives the number of points in the nonzero structure of the ILU decomposition as described in § 4.

TABLE 5.1

Variants of multigrid method that have been applied to case 1

Variant	$sa[k]$	$sb[k]$	$sc[k]$	p	r	A	ILU
1	0	1	1	(2.7)	(2.10)	(2.16)	7
2	0	1	1	(2.7)	(2.10)	(2.15)	7
3	0	1	1	(2.8)	(2.9)	(2.15)	7
4	0	1	1	(2.8)	(2.11)	(2.16)	9
5	1	1	1	(2.7)	(2.10)	(2.16)	7
6	1	1	1	(2.7)	(2.10)	(2.15)	7
7	0	1	2	(2.7)	(2.10)	(2.16)	7
8	0	1	1	(2.7)	(2.10)	(2.16)	5
9	1	0	1	(2.7)	(2.10)	(2.16)	7
10	1	0	1	(2.7)	(2.10)	(2.15)	7

Table 5.2 gives the number of iterations M carried out and the average reduction factor ρ (defined by ρ^M = quotient of Euclidean norms of residue $Ax\text{-}f$ after and before M multigrid iterations) for case 1.

TABLE 5.2

Results of application of ten variants of a multigrid method to case 1; $l=4$

Variant	1	2	3	4	5	6	7	8	9	10
M	4	6	7	4	3	4	4	7	6	6
ρ	0.023	0.077	0.11	0.022	0.0079	0.015	0.022	0.12	0.086	0.078

From Table 5.2 the following conclusions are drawn. Comparing variants 1, 2 and 3, we can see that (2.16) is better than (2.15). (In the interior, (2.15) and (2.16) are identical in this case, but at the boundary they differ.) Comparison of variants 1 and 4 shows that the prolongation and restriction operators (2.7), (2.10) perform just as well as the more expensive operators (2.8), (2.11). Comparison of variants 1, 5 and 6 shows that the additional smoothing of variants 5 and 6, which almost doubles the computational work, does not reduce ρ enough to make variants 5 and 6 competitive. Comparing variants 1 and 7, we conclude that improvement of the accuracy of the coarse grid corrections by an increase of $sc[k]$ beyond 1 is not worthwhile. Comparing 1 and 8, one finds that it does not pay to make the ILU decomposition more sparse. Comparison of 1, 9 and 10 shows that smoothing after, rather than before, coarse grid correction is preferable. Variant 1 is clearly superior to the other variants considered, although there is no certainty that this is also true for other problems; the other variants (except variant 2) have not been tested on other problems. (The reason for including further tests with variant 2 is to make the case for (2.16), rather than (2.15), more convincing, which seems desirable because (2.15) seems to be rather more popular at present.)

The performance of variant 1 as a function of l (mesh size $= 2^{-l}$) is given in Table 5.3. Clearly, the rate of convergence is bounded away from 1 as the mesh size tends to zero, as predicted by the theory.

TABLE 5.3
Variant 1 *applied to case* 1

l	2	3	4	5	6
M	6	8	9	8	8
ρ	0.013	0.037	0.047	0.040	0.033

Next, variants 1 and 2 and the method tested by Hackbusch (1978) are compared for cases 1, 2 and 3 (Table 5.4). The latter method is denoted by H in the sequel. Results for H are directly quoted from Hackbusch (1978). Unfortunately, similar detailed results are not available for other methods in the literature at this moment.

As in Table 5.2, variant 1 is faster than variant 2, which strengthens our preference for (2.16) over (2.15).

TABLE 5.4
Results for cases 1, 2, 3; $l=6$

Case:	1	1	1	2	2	2	3	3	
Variant:	1	2	H	1	2	H	1	2	H
M	8	10	8	10	10	8	7	8	8
ρ	0.033	0.088	0.038	0.15	0.37	0.063	0.025	0.051	0.199

Before any further conclusions can be drawn from Table 5.4, the computational complexities per iteration must be estimated for variant 1 and H. Estimation of the computational complexity is a tedious business which can be carried out only roughly. But because efficiency is what multigrid methods are all about, and because efficiency and robustness of multigrid methods vary widely (although not very much between variant 1 and H, as it turns out), the computational complexity needs to be determined and reported as well as possible.

There are two main differences between variant 1 and H. The first is that for p^k and r^k variant 1 uses the 7-point operators defined by (2.7) and (2.10), whereas H uses the 9-point operators given by (2.8) and (2.11). This makes application of p^k and r^k somewhat cheaper for variant 1, and furthermore, the coarse matrices A^k, $k < l$, are 7-point rather than 9-point operators, as for H. This makes a difference for the work done on coarser grids. More important for the efficiency is the second main difference between both methods, namely the smoothing operators that are used, because this also affects the work done on the finest grid. For the problems of Table 5.4, H uses line relaxation along lines in the x_1-direction, with a special ordering of the lines. Half the lines are visited before coarse grid correction, the other half, and all lines once more, after coarse correction. The total smoothing work is therefore equivalent to two line relaxations, or, on the finest grid, to 28 (cases 1, 2) or 36 (case 3) operations per grid point, as follows from Table 4.2. (We take the variable coefficient case in order to make the estimates more representative of the general case.) As seen from Table 4.1, for variant 1, smoothing takes 18 operations on the finest grid for cases 1, 2, 3. We expect therefore that the computational work for variant 1 for one multigrid iteration is roughly 0.65 (cases 1, 2) or 0.5 (case 3) times that of *H*. Taking the average reduction factors ρ of Table 5.4 into account, one comes to the conclusion that the total work of variant 1 will be roughly 0.42 (case 1), 0.96 (case 2) or 0.24 (case 3) times the total work of *H*.

For cases 1 and 3, variant 1 is better than *H*. But for case 2, variant 1 does much worse than for cases 1 and 3, and is not significantly faster than *H*. The following tables give results of further experiments with anisotropic model problems of the type of case 2.

TABLE 5.5

Variant 1 *applied to anisotropic model problems*; $l=4$; *exact solution*: $\Phi=x^2+y^2$

	$\Phi_{xx}+\varepsilon\Phi_{yy}=2+2\varepsilon$				$\varepsilon\Phi_{xx}+\Phi_{yy}=2+2\varepsilon$			
ε	0.5	0.1	0.01	0.0001	0.5	0.1	0.01	0.0001
M	9	10	7	5	8	7	4	2
ρ	0.047	0.10	0.052	0.00031	0.036	0.023	0.0016	4×10^{-8}

TABLE 5.6

Variant 1 *applied to* $\Phi_{xx}+0.01\Phi_{yy}=2.2$ *for various stepsizes*

l	2	3	4	5	6
M	6	10	10	10	10
ρ	0.0077	0.065	0.100	0.11	0.095

Because the pivots are chosen in a certain sequence in the *ILU* decomposition, the *ILU* smoothing properties are sensitive to rotational transformations, quite like line relaxation. This effect is visible in Table 5.5. For the present *ILU* decomposition, $\Phi_{xx}+\varepsilon\Phi_{yy}$ is the worst anisotropic case (the best for x_1-line relaxation), and $\varepsilon\Phi_{xx}+\Phi_{yy}$ is the best (here x_1-line relaxation would not work at all). Table 5.6 shows that for this problem also the rate of convergence is independent of the mesh size.

Our conclusion for anisotropic problems of the type of case 2 is that, although its performance is somewhat variable, variant 1 is always fast, and never outperformed by *H*. *H* can be made just as robust as variant 1 for this case by including x_2-line relaxation, which would double the work.

Case 1 has also been treated by method *H* using point relaxation as smoothing operator. The computational work is equivalent to two pointwise sweeps, i.e., 18 operations per point. Reasoning as before, we expect that variant 1 and *H* require about the same amount of work per iteration. For *H*, a value of $\rho=0.048$ is reported, and we conclude that the computational complexity of variant 1 will be about 0.61 times that of *H*. Note that this version of *H* would not work for cases 2 and 3.

In order to be able to compare with results reported by Brandt (1977) and Nicolaides (1979), the computational complexity of variant 1 has been measured in terms of workunits (WU). One WU, as introduced by Brandt (1977) and denoted here by WUGS, is the work of one pointwise Gauss–Seidel sweep on the finest grid. One may similarly define one WULU as the work of one *ILU* smoothing operation on the finest grid. For the computations reported in Tables 5.2, 5.3, 5.4., 5.5 and 5.6, it has been measured that the average cost of one multigrid iteration with variant 1 is 1.99 WULU with variance 0.07 WULU. The smallness of this variance is another indication of the robustness of variant 1, and makes it reasonable to assume henceforth a fixed cost of 2 WULU for one multigrid iteration with variant 1. In order to be able to compare with authors who have measured work in terms of WUGS, we will assume, on the basis of Table 4.1, that 1 WULU = 2 WUGS (general 5-point case) or 1 WULU = 3 WUGS (Poisson) exploiting the fact that the coefficients are constant. Of course, in this case one would be better off with a Fourier-analysis–cyclic-reduction method.) This figure may also be obtained by a theoretical operations count as follows. Table 5.7 gives the operations count per grid point for the operations that occur in the procedure multigrid method of § 2.

In Table 5.7, savings near the boundaries are neglected. We consider variant 1. Table 5.8, which expresses the operations count of one multigrid iteration in WULU units, is easily compiled. The work on the coarsest grid is neglected; A^l is assumed to be a 5-point operator and A^k, $k<l$, are 7-point operators. S is applied once for $k=1(1)l$, $r(f-Au)$ is executed for $k=1(1)l-1$ and $u+pu$ is computed for $k=2(1)l$. One WULU is 18 operations per gridpoint.

TABLE 5.7
Operations per grid point for portions of procedure multigrid method

	A: 5-point	A: 7-point
S	18	18
$r(f-Au)$	18	22
$u+pu$	3	3

TABLE 5.8
Operations count for multigrid method, variant 1, *expressed in* WULU

l	2	3	4	5	6	7
WULU	1.89	2.08	2.05	1.99	1.95	1.93

Nicolaides (1979) reports results for combinations of a multigrid method and two finite element methods. One of these, namely linear elements, turns out to be equivalent to variant 1 with pointwise Gauss–Seidel smoothing instead of *ILU* smoothing, if the coefficients are constant. After varying $sa[k]$ and $sc[k]$ (see procedure multigrid method (k) § 2) a best result of 3.9 WUGS per decimal digit is measured for the first 3 multigrid iterations for the Poisson equation with smooth initial error and a 64×64 grid. We find (case 1, Table 5.3) $\rho=0.033$, hence 0.67 iterations per decimal digit, at a cost of 1.34 WULU = 2.68 WUGS (3.02 if the constant coefficients are exploited). During the first three iterations the rate of convergence is $\rho=0.0064$, and a cost per decimal digit is found of 0.91 WULU = 1.8 WUGS. Apart from the Poisson equation, several interesting variable coefficient cases are treated successfully by Nicolaides (1979). One of these is (2.1) with $a_{11}=a_{22}=abs\ (\sin kx_1 \sin kx_2)$, $a_{12}=a_{21}=b_i=c=f=0$, $u/\partial\Omega=0$. The initial values for u are uniformly randomly distributed in $[-1,1]$. Three multigrid iterations are performed. The results for variant 1 are given in Table 5.9. For this test case, Nicolaides (1979) used bilinear elements. Hence A^k, $k=l(-1)1$, are 9-point operators in his case, so that one Gauss–Seidel sweep takes 17 operations per grid point, and 1 WULU = 18/17 WUGS. Table 5.10

TABLE 5.9

ρ for test case of Nicolaides

l/k	2	4	8	16	32
5	0.060	0.063	0.067	0.067	0.061
6	0.065	0.060	0.062	0.070	0.067

gives the work per decimal digit expressed in WUGS. Before the comma: variant 1; after the comma: method of Nicolaides.

TABLE 5.10

Cost per decimal digit for test case of Nicolaides

l/k	2	4	8	16	32
5	1.7, 4.7	1.7, 5.2	1.8, 5.4	1.8, 5.9	1.7, 5.0
6	1.8, 4.5	1.7, 4.8	1.7, 5.8	1.9, 6.3	1.8, 6.1

Distinguishing features of the multigrid methodology advocated by Brandt (1977), (1979) are: (2.15) is used for the coarse grid operators, (2.8) and (2.9) or (2.11) are used for prolongation and restriction and Gauss–Seidel relaxation by points or by lines is used for smoothing. At certain instances, higher order interpolation is used for p^k. Furthermore, in contrast with the fixed smoothing strategy of the methods discussed in the foregoing, the smoothing strategy is variable: The number of smoothing operations preceding and succeeding a coarse grid correction depends on the rate of convergence which is monitored by the algorithm. Brandt (1977, p. 349) introduces the quantity μ_0 as the factor by which the errors are reduced per WU of computational work, counting only smoothing work; Brandt derives a relation between μ_0 and $\bar{\rho}$ (defined in (4.4)), which in the present context reads

$$\mu_0 = \bar{\rho}^{3/4}. \tag{5.4}$$

This relation is not rigorous, but is found to be very realistic according to Brandt (1977). With Gauss–Seidel smoothing, $\bar{\rho} = \frac{1}{2}$ for the Poisson equation (see (4.18)), and one finds $\mu_0 = 0.595$, whereas with *ILU* smoothing $\bar{\rho} = 0.126$ (see (4.17)), hence $\mu_0 = 0.211$. Taking into account that 1 WULU = 2 WUGS (Table 4.1, general 5-point case), *ILU* smoothing comes out best. Counting only relaxation work, 1 multigrid iteration with variant 1 takes somewhat less than $\frac{4}{3}$ WULU, and the observed value of μ_0 comes out to be 0.077 per 1 WULU for case 1, $l = 6$ (on the basis of Table 5.3), which is considerably better than the estimate $\mu_0 = 0.211$ based on (5.4). In (5.4) the effects of the prolongation and restriction operators and the quality of the coarse grid operators are not taken into account: apparently these have a very benificial effect in variant 1, and our preference for (2.7), (2.10) and (2.16) is strengthened once more. In an experiment on the Poisson equation using the variable smoothing strategy, Brandt (1977, p. 354) measures $\mu_0 = 0.537$ per 1 WUGS; cf. our just-quoted value $\mu_0 = 0.077$ per 1 WULU ~ 0.278 per 1 WUGS. One might ask whether the inclusion of a variable smoothing strategy would further improve the performance of variant 1. Because of the (possibly problem-dependent) tuning this would require, and in view of the speed and robustness already obtained, we do not think it worthwhile to pursue this matter further.

This concludes our comparison with other methods. Another important model problem, which has not been treated by other authors, is the convection-diffusion equation. In Wesseling and Sonneveld (1980) this equation is treated with variant 1; furthermore, results are reported for the Navier–Stokes equations.

6. Final remarks. It has been shown that the Galerkin coarse grid approximation, 7-point prolongation and restriction, incomplete LU smoothing and a fixed multigrid strategy (i.e., in procedure multigrid method (k) of § 2 $sa[k]=0$, $sb[k]=sc[k]=1$ for all problems) result in a fast and robust multigrid method. Coarse grid Galerkin approximation was found to be better than finite difference approximation, and ILU-smoothing is better than other smoothing methods that have been proposed. Furthermore, 7-point prolongation and restriction is competitive with 9-point prolongation and restriction, which is important for three-dimensional problems.

It seems likely that multigrid methods will soon come to be widely used as fast solvers for elliptic problems. An important aspect that has not been touched upon here is adaptive mesh generation (cf. Brandt (1977), (1979), Hemker (1980a)). Furthermore, the application of multigrid methods is not restricted to partial differential equations (cf. Hemker and Schippers (1981)).

Appendix A

Proof of (3.26). The operator A^l is defined by (2.3) with $b_i \equiv c \equiv 0$. Define (deleting the superscript l for brevity) $\bar{A} \equiv -\frac{1}{2}a_{ij}(\nabla_i\Delta_j + \Delta_i\nabla_j)$. It is not difficult to verify that

$$\|(A-\bar{A})u\|_0 \leqq \alpha\|u\|_1, \qquad \alpha^2 \equiv 2\sum_{j=1}^{2}\sup_{\Omega}(\nabla_i a_{ij})^2. \tag{A1}$$

Nitsche and Nitsche (1960) show that

$$\|u\|_2 \leqq \sqrt{2}\, C_5^{-2}C_6\|\bar{A}u\|_0. \tag{A2}$$

(Their definition of $\bar{A}$ and $\|\cdot\|_2$ is slightly different, but their proof holds also in our case.) Furthermore, using (3.6) and (3.12),

$$\|Au\|_0 \geqq \|Au\|_{-1} \geqq \frac{(Au,u)}{\|u\|_1} \geqq C_5\|u\|_1. \tag{A3}$$

Combining (A1)–(A3) one obtains

$$\|u^l\|_{2,l} \leqq D_5\|A^l u^l\|_{0,l}, \qquad D_5 \equiv \sqrt{2}\, C_5^{-2}C_6\left(\frac{1+\alpha}{C_5}\right). \tag{A4}$$

Next, (A4) is generalized to coarser grids. First some useful consequences of (3.8) are derived. Define $r^{km} \equiv r^{k+1}r^{k+2}\dots r^m$, $m>k$. We have

$$\begin{aligned}\|r^{km}u^m\|_{-1,k} &= \sup_{\|v^k\|_{1,k}\leqq 1}|(u^m, p^{mk}v^k)_m| \\ &\leqq \sup_{\|p^{mk}v^k\|_{1,m}\leqq 1}|(u^m, p^{mk}v^k)_m| \leqq \|u^m\|_{-1,m}.\end{aligned} \tag{A5}$$

Furthermore,

$$\|u^m - p^{mk}r^{km}u^m\|_{1,m} \leqq 2^{-k}C_2\|u^m\|_{2,m}, \qquad m>k, \tag{A6}$$

which may be derived by noting that $1-p^kr^{km} = 1-p^mr^m + p^m(1-p^{m-1}r^{m-1})r^m + \dots + p^{m,k+1}(1-p^{k+1}r^{k+1})r^{k+1,m}$, and from (3.8a, b, e). Similarly one may derive

(A7) $$\|u^k - r^{km}p^{mk}u^k\|_{-1,k} \leqq \tfrac{4}{3}C_4 \cdot 4^{-k}\|u^k\|_{1,k}, \qquad m > k.$$

Next we note that, for arbitrary (deleted) k

(A8) $$\|u\|_1 \leqq C_1^{-2}C_5^{-1}\|Au\|_{-1},$$

since $\|Au\|_{-1} = \sup_{\|v\|_1 \leqq 1}|B(u, v)| \geqq C_1^2C_5\|u\|_1$, because of (3.17).

Now (3.26) will be derived. Let $A^ku^k = f^k$, $k < l$. Define $\hat{u}^k$ and $\hat{u}^l$ by $A^k\hat{u}^k = r^{kl}p^{lk}f^k$, $A^l\hat{u}^l = p^{lk}f^k$. Using (3.8a), (3.21) and (A6), we get

(A.9) $$\begin{aligned}\|\hat{u}^k - r^{kl}\hat{u}^l\|_{1,k} &\leqq C_1^{-1}\|p^{lk}\hat{u}^k - p^{lk}r^{kl}\hat{u}^l\|_{1,l}\\ &\leqq C_1^{-1}\|p^{lk}\hat{u}^k - \hat{u}^l\|_{1,l} + C_1^{-1}\|\hat{u}^l - p^{lk}r^{kl}\hat{u}^l\|_{1,l}\\ &\leqq C_1^{-1}(1+D_1)\|\hat{u}^l - p^{lk}r^{kl}\hat{u}^l\|_{1,l} \leqq C_1^{-1}C_2(1+D_1)2^{-k}\|\hat{u}^l\|_{2,l}.\end{aligned}$$

From (A4) and (3.8c) it follows that

(A10) $$\|\hat{u}^l\|_{2,l} \leqq D_5\|f^k\|_{0,k}.$$

Since $A^k(u^k - \hat{u}^k) = f^k - r^{kl}p^{lk}f^k$, we conclude from (A7), (A8) and (3.7) that

(A11) $$\|u^k - \hat{u}^k\|_{1,k} \leqq \tfrac{4}{3}\sqrt{8}\,C_1^{-2}C_4C_5^{-1}2^{-k}\|f^k\|_{0,k}.$$

From (A9), (A10), (A11) and (3.7) we have

(A12) $$\|u^k - r^{kl}\hat{u}^l\|_{2,k} \leqq C_1^{-2}\left(\frac{32}{3}\frac{C_4}{C_5} + \sqrt{8}\,C_1C_2D_5(1+D_1)\right)\|f^k\|_{0,k}.$$

With (3.8e) and (A10) one obtains

(A13) $$\|r^{kl}\hat{u}^l\|_{2,k} \leqq D_5\|f^k\|_{0,k}.$$

Combination of (A12) and (A13) results in

(A14) $$\|u^k\|_{2,k} \leqq D_2\|f^k\|_{0,k},$$

with D_2 following in an obvious way from (A12), (A13). This is the desired result.

Appendix B. One way to compute A^k as defined by (2.16) is as follows, following Frederickson (1975). Let (2.7) and (2.10) hold, and let A^k be a 7-point operator; the following treatment is easily extended to other cases. Define $Z \equiv \mathbb{Z} \times \mathbb{Z}$, $M = \{(m_1, m_2) | m_{1,2} = 0, \pm 1\} \backslash \{1, 1), (-1, -1)\}$, $R^k = \{(m_1, m_2) | m_{1,2} = 0(1)2^k\}$. M will be used to enumerate the atoms of the difference molecule corresponding to A^k, and R^k to enumerate the points of Ω^k. Elements of the matrix A^k will be denoted by A^k_{ij}, $i \in R^k$, $j \in M$. With this notation, matrix-vector multiplication is defined by $(A^ku^k)_i = \sum_{j \in M} A^k_{ij}u^k_{i+j}$. Prolongation and restriction may be defined as follows. Define $t: Z \to \mathbb{R}$ by $t(0, 0) = 1$; $t(j) = \frac{1}{2}$, $j \in M\backslash(0, 0)$; $t(j) = 0$, $j \notin M$. Then

(B1) $$(p^ku^{k-1})_i = \sum_{j \in Z} t(i - 2j)u^{k-1}_j, \qquad i \in R^k,$$

(B2) $$(r^ku^k)_i = \tfrac{1}{4}\sum_{j \in Z} t(j)u^k_{2i+j}, \qquad i \in R^{k-1},$$

(B3) $$A^{k-1}_{ij} = \tfrac{1}{4}\sum_{u,v \in Z}\sum t(u)t(v)A^k_{2i+u,2j+u-v}, \qquad i \in R^{k-1}, \quad j \in M.$$

The (obvious) key to efficient computation of A^{k-1} is to pick a combination (j, u, v) and to check whether $2j+u-v \in M$ before sweeping through R^{k-1}. The obtained contribution to A_{ij}^{k-1} is stored and is augmented by further sweeps through R^{k-1} for different combinations (j, u, v), until all relevant combinations (j, u, v) have been exhausted. By checking off all possible combinations (j, u, v), one arrives at an operations count of about 110 per point of Ω^{k-1}. Thus, computation of A^k, $k = l-1(-1)1$, takes less than 37 operations per point of Ω^l, or 2 WULU.

REFERENCES

N. S. BAKHVALOV (1966), *On the convergence of a relaxation method with natural constraints on the elliptic operator*, USSR Comp. Math. Math. Phys., 6, pp. 101–135.

A. BRANDT (1977), *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31, pp. 333-390.

——— (1979), *Multi-level adaptive techniques (MLAT) for singular perturbation problems*, in Numerical Analysis of Singular Perturbation Problems, P. W. Hemker and J. J. Miller, eds., Academic Press, London.

R. P. FEDORENKO (1962), *A relaxation method for solving elliptic difference equations*, USSR Comp. Math. Math. Phys., 1, pp. 1092–1096.

——— (1964), *The speed of convergence of one iterative process*, USSR Comp. Math. Math. Phys., 4 no. 3, pp. 227–235.

P. O. FREDERICKSON (1975), *Fast approximate inversion of large sparse linear systems*, Mathematics Report 7-75, Lakehead University, Thunder Bay, Ontario, Canada.

W. HACKBUSCH (1978), *On the multi-grid method applied to difference equations*, Computing, 20, pp. 291–306.

——— (1980), *Convergence of multi-grid iterations applied to difference equations*, Math. Comp., 34, pp. 425–440.

P. W. HEMKER (1980), *The incomplete LU-decomposition as a relaxation method in multi-grid algorithms*, in Boundary and Interior Layers—Computational and Asymptotic Methods, J. J. H. Miller, ed., Boole Press, Dublin, pp. 306–311.

——— (1980a), *On the structure of an adaptive multi-level algorithm*, BIT, 20, pp. 289–301.

P. W. HEMKER AND H. SCHIPPERS (1981), *Multigrid methods for the solution of Fredholm equations of the second kind*, Math. Comp., 36, pp. 215–232.

E. ISAACSON AND H. B. KELLER (1966), *Analysis of Numerical Methods*, John Wiley, New York.

R. A. NICOLAIDES (1979), *On some theoretical and practical aspects of multigrid methods*, Math. Comp., 30, pp. 933–952.

J. NITSCHE AND J. J. C. NITSCHE (1960), *Error estimates for the numerical solution of elliptic differential equations*, Arch. Rat. Mech. Anal., 5, pp. 293–306.

P. WESSELING (1980), *The rate of convergence of a multiple grid method*, in Numerical Analysis, Proceedings, Dundee 1979, G. A. Watson, ed., Lecture Notes in Mathematics 773, Springer-Verlag, Berlin, pp. 164–184.

P. WESSELING AND P. SONNEVELD (1980) *Numerical experiments with a multiple grid and a preconditioned Lanczos method*, in Approximation Methods for Navier–Stokes Problems, Proceedings, Paderborn 1979, R. Rautmann, ed., Lecture Notes in Mathematics 771, Springer-Verlag, Berlin, pp. 543–562.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

STABLE BOUNDARY APPROXIMATIONS FOR IMPLICIT TIME DISCRETIZATIONS FOR GAS DYNAMICS*

BERTIL GUSTAFSSON† AND JOSEPH OLIGER‡

Abstract. We consider the problem of constructing stable difference methods for the initial boundary value problem for the linearized equations of gas dynamics in one space dimension using the implicit time differencing methods considered by Beam and Warming [2]. Centered spacial differences are used in the interior. We investigate the stability of this class with two forms of extrapolation for the scalar outflow problem. (We consider the problem of specifying data in the primitive variables and computing in terms of the conservative variables in the interior.) We show that the whole class of methods is stable for the subsonic inflow and outflow problems with various data specifications and extrapolation methods. We also show that the methods considered are stable for the solid wall boundary problem when we set $u = 0$ and use one-sided differences in the other equations.

Key words. gas dynamics, initial boundary value problems, difference methods

1. Introduction.

The main problem which we consider here is that of constructing stable difference methods for the initial boundary value problem for the equations of gas dynamics. We investigate the class of implicit time differencing methods which has been considered by Beam and Warming [2] for computations in aerodynamics. We use second-order centered differences to approximate spacial derivatives at interior points. This class of time differencing methods includes one- and two-step methods of first- and second-order accuracy.

We begin in § 2 by investigating the stability of eight different methods of the class of Beam and Warming for the scalar outflow problem. We consider two different extrapolations. The stability of some of these methods has been proved earlier [3]. These results are included in our unified treatment.

We discuss boundary conditions which yield well-posed problems for the linearized Eulerian equations of gas dynamics in § 3. Since we regard this as a pilot study for problems in two and three space dimensions we restrict this discussion to boundary conditions for the differential equations which can be extended to yield well-posed problems in more space dimensions.

The conservative and primitive variable forms of the equations are both discussed and we relate boundary conditions for these two formulations. We conclude § 3 with a discussion of boundary approximations whose stability follows immediately from the analysis of the scalar problem in the preceding section.

In the last two sections we consider approximations which utilize boundary data specified directly in terms of the primitive variables ρ, u and p as opposed to combinations of them such as Riemann invariants. In § 4, we treat the open boundary problem using the normal mode analysis technique described in [3]. In particular, we are able to show that all methods considered are stable for the subsonic inflow problem if ρ and u are specified at the boundary and p is extrapolated linearly in space to the boundary from the interior (Theorem 4.1). We also show that all methods are stable for the subsonic outflow problem if u or p is specified at the boundary and ρ, and p

* Received by the editors November 12, 1980, and in revised form October 26, 1981. This work was supported by the National Aeronautics and Space Administration–Ames University Consortium Interchange NCA2-0R745-004, by the Swedish Natural Science Research Council F4370-101, and by the Office of Naval Research under contracts N00014-75-C-1132 and N00014-80-C-0076.

† Department of Computer Sciences, Uppsala University, Uppsala, Sweden.
‡ Department of Computer Science, Stanford University, Stanford, California 94305.

or u, respectively, are linearly extrapolated from the interior (Theorem 4.2). This extrapolation can also be done in conservative variables. The supersonic cases follow immediately from the scalar results.

We treat the solid wall boundary problem in § 5, using the so-called energy method [6]. This is the only method known to be valid in this situation. We show that all of the implicit methods considered are stable if we set $u = 0$ at the boundary and use one-sided differences in space in the ρ and p equations (Theorem 5.1).

2. Description of the methods and analysis of the scalar outflow problem. In this section we consider approximations of the scalar problem

$$u_t = au_x, \qquad 0 \leqq x < \infty, \quad 0 \leqq t,$$

where $a > 0$ is a constant and initial data are given at $t = 0$. The solution of this problem is uniquely determined by the initial data and one cannot specify the solution at $x = 0$ for $t > 0$. A mesh is defined to be the set of points (x_j, t_n) with $x_j = jh$, $j = 0, 1, 2, \cdots$, and $t_n = nk$, $n = 0, 1, 2, \cdots$, where $h > 0$ and $k > 0$. We use the notation $u_j^n = u(x_j, t_n)$ for a function defined on this mesh. The differential operator $\partial/\partial x$ is approximated by the difference operator D_0 defined by

$$D_0 u_j^n = (u_{j+1}^n - u_{j-1}^n)/2h.$$

Beam and Warming [2] defined a general class of approximations for

$$\frac{\partial u_j}{\partial t} = aD_0 u_j$$

by

$$\tau(E)u_j^n = ka\sigma(E)D_0 u_j^n. \tag{2.1}$$

Here E is the shift operator defined by $Eu_j^n = u_j^{n+1}$. $\tau(E)$ (the notation $\rho(E)$ is used in [2]) and $\sigma(E)$ are second-degree polynomials defined by

$$\tau(E) = (1+\xi)E^2 - (1+2\xi)E + \xi, \tag{2.2}$$

$$\sigma(E) = \theta E^2 + (1-\theta+\phi)E - \phi. \tag{2.3}$$

The general scheme can be written

$$(1+\xi)u_j^{n+2} - (1+2\xi)u_j^{n+1} + \xi u_j^n = kaD_0(\theta u_j^{n+2} + (1-\theta+\phi)u_j^{n+1} - \phi u_j^n). \tag{2.4}$$

In [2] the most well-known methods of type (2.4) are listed. We list those methods which we discuss here. We add the two-step backward Euler scheme, which we also consider.

TABLE 2.1

Method	ξ	θ	ϕ
1. Backward Euler	0	1	0
2. Two-step backward Euler	$-\frac{1}{2}$	1	0
3. Trapezoidal (Crank–Nicolson)	0	$\frac{1}{2}$	0
4. Backward differentiation	$\frac{1}{2}$	1	0
5. Adams	0	$\frac{3}{4}$	$-\frac{1}{4}$
6. Lees	$-\frac{1}{2}$	$\frac{1}{3}$	$-\frac{1}{3}$
7. Two-step trapezoidal	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
8. A-contractive	$-\frac{1}{6}$	$\frac{5}{9}$	$-\frac{2}{9}$

These methods are all implicit and unconditionally stable for the Cauchy problem.

Equation (2.4) cannot be used at $x=0$. The difference equations require an extra boundary condition and we consider two types of extrapolation: *space extrapolation,*

$$\Delta_+^q u_0^n = 0, \qquad q \geqq 1, \tag{2.5}$$

where

$$\Delta_+ u_j^n = u_{j+1}^n - u_j^n,$$

and *space-time extrapolation,*

$$({}^{-}\Delta_+)^q u_0^n = 0, \qquad q \geqq 1, \tag{2.6}$$

where

$${}^{-}\Delta_+ u_j^n = u_{j+1}^{n-1} - u_j^n.$$

If $q=2$ these formulas are equivalent to linear extrapolation, (2.5) extrapolates in the x-direction and (2.6) along diagonals in the (x–t)-plane. The extrapolation (2.5) with $q=2$ is equivalent to using (2.4) for $j=1$ if $D_+ = h^{-1}\Delta_+$ is substituted for D_0. If either of these extrapolations yield stable methods and $q \geqq 2$, we will obtain an $O(h^2)$ convergence rate in space [4].

THEOREM 2.1. *If the space extrapolation* (2.5) *is used with the methods* (2.4) *in Table* 2.1, *then the resulting methods are all stable for the initial boundary value problem.*

THEOREM 2.2. *If the space-time extrapolation* (2.6) *is used with the methods* (2.4) *in Table* 2.1, *then the resulting method is unconditionally stable with backward Euler, trapezoidal, backward differentiation, Adams and A-contractive time differencing*; *is conditionally stable for trapezoidal* (*Crank–Nicolson*) *time differencing for* $\lambda = ak/2h < 1$; *and is unconditionally unstable for the remaining time differencing methods.*

The stability definition for the initial boundary value problem used in the formulation of these two theorems is that of Gustafsson, Kreiss and Sundström [3, Def. 3.3]. We will use the normal mode technique developed and justified in that paper to prove both of these theorems. We will develop the structure of the arguments and the use of the theory in a general way before we turn to the details of the two proofs.

Results obtained by means of normal mode analysis are based upon the behavior of solutions of the so-called resolvent equations. These are formally derived from (2.4), (2.5) and (2.6) by substituting $u_j^n = z^n v_j$ where z is a complex number. We obtain, respectively,

$$[(1+\xi)z^2 - (1+2\xi)z + \xi]v_j = kaD_0[\theta z^2 + (1-\theta+\phi)z - \phi]v_j, \qquad j = 1, 2, \cdots, \tag{2.7}$$

$$\Delta_+^q v_j|_{j=0} = 0, \tag{2.8}$$

$$\Delta_+^q (z^{q-j} v_j)|_{j=0} = 0. \tag{2.9}$$

The general solution of (2.7) which is bounded as $j \to \infty$ for $|z| > 1$, can be written in the form

$$v_j = v_0 \kappa^j, \tag{2.10}$$

where κ is the root of the characteristic equation

$$[(1+\xi)z^2 - (1+2\xi)z + \xi]\kappa = \lambda(\kappa^2 - 1)[\theta z^2 + (1-\theta+\phi)z - \phi], \tag{2.11}$$

such that $|\kappa| < 1$ if $|z| > 1$.

Equation (2.11) is formally obtained from (2.7) by substituting $v_j = \kappa^j$. It follows from the discussion following Lemma 5.1 of [3] that only one root of the quadratic

(2.11) has modulus less than one. This is an immediate consequence of the stability of (2.4) for the Cauchy problem and justifies (2.10). It is proved in [3] (see Lemma 10.3 and the following sentence) that our approximations are stable for the initial boundary value problem if, and only if, (2.7) with boundary condition (2.8) or (2.9) has no nontrivial bounded solution of the form (2.10) for $|z|\geqq 1$ (note that one must include $|z|=1$). This is established by substituting (2.10) into (2.8) or (2.9) and showing that $v_0=0$. When $|z|=1$, one or both of the roots of (2.11) may have modulus one. If this is the case, the κ in (2.10) is defined by continuity to be that root which is the limit of the root $\kappa(z')$, $|\kappa(z')|<1$ for $|z'|>1$, as $|z'|\to 1$.

Proof of Theorem 2.1. If we substitute (2.10) into (2.8) we find that $v_0=0$ unless $\kappa=1$, which can only happen when $|z|=1$. If $\kappa=1$, the right-hand side of (2.11) vanishes so that $\kappa=1$ only at the roots of $\tau(z)=0$. $\tau(z)=0$ has the roots $z=1$ and $z=\xi/(1+\xi)$. ξ is real, so $|\xi/(1+\xi)|=1$ can only occur when $\xi=-\frac{1}{2}$ which yields $z=-1$.

We examine the point $z=1$ first. We consider a perturbation $z=1+\delta, \delta>0$, write $\kappa=1+\varepsilon$, substitute into (2.11) and obtain

$$(1+\xi)2\delta-(1+2\xi)\delta=2\lambda\varepsilon[\theta+(1+\phi-\theta)-\phi]+O(\delta^2).$$

Consequently $\varepsilon\approx\delta/2\lambda>0$ for δ sufficiently small. Thus, the root $\kappa=1$ of (2.11) at $z=1$ is not the limit of roots of modulus less than one and need not be considered for any set of parameters ξ, θ, ϕ.

Now we consider the point $z=-1$ with $\xi=-\frac{1}{2}$, and make the perturbation analysis for $z=-1-\delta$, $\delta>0$, in the same way as before. We obtain

$$\varepsilon\approx\delta\{2\lambda[2(\theta-\phi)-1]\}^{-1}$$

and find that $\kappa\approx 1+\varepsilon>1$ if and only if $\theta-\phi>\frac{1}{2}$.

Proof of Theorem 2.2. We now substitute (2.10) into (2.9) and find that $v_0=0$ unless $z=\kappa$. We substitute $z=\kappa$ into (2.11) and get

$$[(1+\xi)z^2-(1+2\xi)z+\xi]z=\lambda(z^2-1)[\theta z^2+(1+\phi-\theta)z-\phi]. \tag{2.12}$$

From our proof of Theorem 2.1 we know that $z=1$ is a root of this equation and that $\kappa(z=1)\neq 1$. Therefore we may divide (2.12) by $(z-1)$ to get

$$[(1+\xi)z-\xi]z=\lambda(z+1)[\theta z^2+(1+\phi-\theta)z-\phi]. \tag{2.13}$$

We now consider several different cases.

Methods 2, 6, 7 ($\xi=-\frac{1}{2}$). In this case $z+1$ is a factor of (2.13). If we consider the perturbation $z=-1-\delta$, $\delta>0$ with $\kappa=-1-\varepsilon$ and substitute into (2.11), we find

$$\varepsilon\approx-\delta\{2\lambda[2(\theta-\phi)-1]\}^{-1}.$$

Therefore, $\varepsilon<0$ and $|\kappa|<1$ if and only if $\theta-\phi>\frac{1}{2}$, which implies that such methods are unstable. Referring to Table 2.1 we find that the Lees method, the two-step backward Euler method, and two-step trapezoidal method satisfy $\theta-\phi>\frac{1}{2}$ and are therefore unstable.

Methods 1, 4 ($\theta=1, \phi=0$). Equation (2.11) is now

$$[(1+\xi)z-\xi]z=\lambda(z+1)z^2.$$

If we divide out the root $z=0$ which does not concern us, we get

$$z^2-\left(\frac{1+\xi}{\lambda}-1\right)z+\frac{\xi}{\lambda}=0. \tag{2.14}$$

The roots z_1 and z_2 of this equation are either real or complex conjugates. The condition $|z|=1$ implies that either $z=1$, $z=-1$, or $z_1z_2=1$. We have already treated the case $z=1$, which causes no trouble. If $z=-1$ then $\xi=-\frac{1}{2}$, which is not true for methods 1 and 4. If $z_1z_2=1$, then $\lambda=\xi$. This means that the only nontrivial situation is $\lambda=\frac{1}{2}$ for method 4. But then (2.14) implies $\kappa=z=1$, which we have already discussed. So methods 1 and 4 are stable.

Method 3 (*trapezoidal*). This method was said to be stable with (2.6) in [3]. The correct stability condition is $\lambda<1$, [R. Beam, R. Warming, and H. Yee, private communication]. (The same condition is also valid with the boundary condition

$$u_0^{n+1}=u_0^n+2\lambda(u_1^n-u_0^n),$$

as shown by Sköllermo [7].)

Method 5 (*Adams*). We substitute the parameter values $\xi=0$, $\theta=\frac{3}{4}$ and $\phi=-\frac{1}{4}$ into (2.13) and get

$$3\lambda z^3+(3\lambda-4)z^2+\lambda z+\lambda=0. \tag{2.15}$$

We will use a continuity argument. If $\lambda=0$, (2.15) has two roots $z_{1,2}=0$. If λ is small $z_1\approx-\sqrt{\lambda}/2$, $z_2\approx\sqrt{\lambda}/2$, and z_3 is real and large in magnitude. Let λ increase. In order for us to have $|z|=1$ for some λ, z must be ±1 or z must first be a double root of (2.15) on the real axis. We can rule out $z=-1$ immediately since it does not satisfy (2.15). Obviously, $-1<z_1<0$ for $\lambda>0$ since (2.15) has no root $z=0$ for $\lambda>0$. To have $\kappa_1=z_2$ or z_3 with κ_1 complex and z_2 and z_3 conjugates implies $|\kappa_1|=|z_2|=|z_3|=1$. Since $z_1z_2z_3=-\frac{1}{3}$ we must have $z_1=-\frac{1}{3}$. But $-\frac{1}{3}$ is a root of (2.15) only for $\lambda=\frac{1}{2}$ which has $\kappa_1=-1\neq z_2=z_3=1$. So the method is stable.

Method 8 (*A-contractive*). We substitute the parameter values $\xi=-\frac{1}{6}$, $\theta=\frac{5}{9}$, $\phi=-\frac{2}{9}$ into (2.13) and obtain

$$10\lambda z^3+(14\lambda-15)z^2+(8\lambda-3)z+4\lambda=0. \tag{2.16}$$

We use a continuity argument as in the previous case. When λ is small we have $z_1\approx-\frac{1}{5}$, $z_2\approx4\lambda/3$ and z_3 is large in magnitude. As before, if we are to have a z-value on the unit circle then one of the roots must take on the values $z=1$ or $z=-1$ as λ increases. Again, $\lambda=\frac{1}{2}$ yields a double root $z_1=z_3=1$ and $-1<z_2<0$. As before $z=\pm i$ must be the solutions of (2.16) but this is impossible for any $\lambda>0$ and stability is proved.

This concludes the proof of Theorem 2.2.

3. The Euler equations and characteristic extrapolation. We now consider the initial boundary value problem for the linearized Eulerian equations of gas dynamics. ρ is the density, u is the velocity, and p is the pressure of the fluid. We linearize about a constant state $\hat\rho$, $\hat u$, $\hat p$ and obtain the equations

$$\begin{bmatrix}\rho\\ u\\ p\end{bmatrix}_t+\begin{bmatrix}\hat u & \hat\rho & 0\\ 0 & \hat u & \hat\rho^{-1}\\ 0 & \gamma\hat p & \hat u\end{bmatrix}\begin{bmatrix}\rho\\ u\\ p\end{bmatrix}_x=0,$$

which we also write as

$$U_t+AU_x=0. \tag{3.1}$$

The variable γ is the ratio of specific heats at constant pressure and constant volume. The state variables ρ, u and p are often called the primative variables. Alternatively, these equations can be written in terms of the so-called conservative variables $\rho, m = \rho u$, and e, where m is the momentum and e is the total energy per unit volume. We assume that we are dealing with a perfect gas. In this case we can relate e to the primitive variables via

$$e = \frac{p}{\gamma - 1} + \frac{\rho u^2}{2}. \tag{3.2}$$

The linearized equations in conservative variables are

$$\begin{bmatrix} \rho \\ m \\ e \end{bmatrix}_t + \begin{bmatrix} 0 & 1 & 0 \\ -\frac{\gamma-3}{2}\left[\frac{\hat m}{\hat\rho}\right]^2 & (3-\gamma)\frac{\hat m}{\hat\rho} & \gamma - 1 \\ -\frac{\gamma \hat e \hat m}{\hat\rho^2} + (\gamma-1)\frac{\hat m^3}{\hat\rho^3} & \frac{\gamma \hat e}{\hat\rho} + \frac{3(1-\gamma)\hat m^2}{2\hat\rho^2} & \gamma\frac{\hat m}{\hat\rho} \end{bmatrix} \begin{bmatrix} \rho \\ m \\ e \end{bmatrix}_x = 0,$$

which we also write as

$$V_t + BV_x = 0. \tag{3.3}$$

The linearized relation between the two sets of variables is

$$V = EU \tag{3.4}$$

where

$$E = \begin{bmatrix} 1 & 0 & 0 \\ \hat u & \hat\rho & 0 \\ \frac{\hat u^2}{2} & -\hat\rho\hat u & \frac{1}{\gamma-1} \end{bmatrix}.$$

We can now formulate our initial boundary value problem. We look for solutions of (3.1) or (3.3) on the domain $0 \leqq x < \infty$, $t \geqq 0$ satisfying an initial condition. We must also specify boundary conditions at $x = 0$ for $t > 0$. We will consider several different cases. We will only consider those boundary conditions which can be extended to yield well-posed problems in two and three spacial dimensions. The eigenvalues of the matrices A and B are $\hat u$, and $\hat u \pm c$ where $c = \sqrt{\hat p \gamma / \hat\rho}$ is the local sound speed. The number of boundary conditions given at $x = 0$ must equal the number of positive eigenvalues of A or B.

Supersonic inflow. If $\hat u > c$ we have supersonic inflow. In this case all variables must be specified at the boundary and stability follows trivially for all of the difference approximations we are considering for both systems, (3.1) and (3.3).

Supersonic outflow. If $\hat u < -c$ we have supersonic outflow. In this case none of the variables may be specified. If all of the variables for either system (3.1) or (3.3) are extrapolated using any of the stable extrapolations discussed in § 2, the resulting method is stable. In order to see this, we need the transformation of systems (3.1) and (3.3) to diagonal form. We define the matrices

$$T = \begin{bmatrix} c^2 & 0 & -1 \\ 0 & \hat\rho c & 1 \\ 0 & -\hat\rho c & 1 \end{bmatrix}$$

and

$$S = \begin{bmatrix} 2c^2 - (\gamma-1)\hat u^2 & 2(\gamma-1)\hat u & -2(\gamma-1) \\ -2c\hat u + (\gamma-1)\hat u^2 & 2c - 2(\gamma-1)\hat u & 2(\gamma-1) \\ 2c\hat u + (\gamma-1)\hat u^2 & -2c - 2(\gamma-1)\hat u & 2(\gamma-1) \end{bmatrix}.$$

Then we can transform A and B to diagonal form using T and S, i.e.,

$$TAT^{-1} = \Lambda, \qquad SBS^{-1} = \Lambda$$

where $\Lambda = \operatorname{diag}(u, u+c, u-c)$. If we now use the approximation (2.4) for equation (3.1), we have

$$(3.5)\quad (1+\xi)U_j^{n+2} - (1+2\xi)U_j^{n+1} + \xi U_j^n = -kAD_0(\theta U_j^{n+2} + (1-\theta+\phi)U_j^{n+1} - \phi U_j^n).$$

If we now multiply through by T^{-1} on the left and substitute $W_j^n = T^{-1}U_j^n$, we obtain a diagonal system. The boundary extrapolations (1.5) and (1.6) are now written in terms of W_j^n and do not couple the components of W_j^n. Thus, the equation (3.4) and accompanying extrapolation (1.5) or (1.6) are just a collection of scalar equations and the results of § 2 apply directly. This same argument applies to the approximations (2.4) of equation (3.3).

Subsonic inflow. If $0 < \hat{u} < c$ we have subsonic inflow. In this case, A and B have exactly two positive eigenvalues so two boundary conditions are required. It has been shown [5] that we can give

1) ρ and u, or
2) $c^2\rho - p$ and $\hat{\rho}cu + p$,

for the system (3.1). The variables specified in 2) are the first two components of $W = T^{-1}U$. These conditions can be extended to boundary conditions which yield well-posed problems in two and three space dimensions (see [5]). If we gave p and ρ we would get a well-posed problem in one space dimension, but this condition cannot be extended to more space dimensions and we will not consider it here.

If we use the approximation (3.5) in all interior points and use the extrapolation (2.5) or (2.6) in the variable $\hat{\rho}cu - p$, then we can use the scalar results of § 2. We can solve for all three variables on the boundary in terms of data and values at interior points. We use the diagonalization argument of the previous subsection with the fact that the extrapolated values are bounded to obtain

RESULT 3.1. *If we use an approximation* (3.5) *with an extrapolation of the form* (2.5) *or* (2.6) *of $\hat{\rho}cu - p$, which has been shown to be stable for the scalar problem in* § 2, *then the method is stable with either one of the inflow boundary conditions,* 1) *or* 2).

Suppose we use an approximation (2.4) of equation (3.3) with conservative variables. Then Result 3.1 can be applied simply by using the relation (3.4).

In § 4 we will consider the problem of extrapolating p with boundary condition 1).

Subsonic outflow. If $-c < \hat{u} < 0$ we have subsonic outflow. In this case, one boundary condition is required. It has been shown [5] that specifying

3) u or
4) p or
5) $\hat{\rho}cu + p$

yield well-posed problems. Using the same arguments as above we obtain:

RESULT 3.2. *If we use an approximation* (3.5) *with extrapolation of the form* (2.5) *or* (2.6) *of $c^2\rho - p$ and $\hat{\rho}cu - p$, which has been shown to be stable for the scalar problem in* § 2, *then the method is stable with any of the outflow boundary conditions* 3), 4) *or* 5).

We can apply this result to computation with the conservative variables as remarked earlier.

We consider extrapolations of ρ and u or p in § 4.

4. The Euler equations with other extrapolations.

4.1. Subsonic inflow. We reconsider the subsonic inflow problem and investigate an extrapolation of p with the specification of ρ and u. The homogeneous form of the boundary conditions for the stability analysis is

$$\rho_0^n = 0, \quad u_0^n = 0, \quad p_0^n = 2p_1^n - p_2^n. \tag{4.1}$$

THEOREM 4.1. *All of the approximations* (3.5) *and the analogous approximations in terms of conservative variables with parameter values in Table* 1 *are stable with the boundary conditions* (4.1).

Proof. We rewrite our approximation (3.5) in the form

$$\tau(E)U_j^n + \lambda\sigma(E)A(U_{j+1}^n - U_{j-1}^n) = 0, \tag{4.2}$$

where τ, σ and λ were defined in § 2. The corresponding resolvent equations are

$$\tau(z)\hat{U}_j + \lambda\sigma(z)A(\hat{U}_{j+1} - \hat{U}_{j-1}) = 0 \tag{4.3}$$

for the interior points and

$$\hat{\rho}_0 = 0, \quad \hat{u}_0 = 0, \quad \hat{p}_0 = 2\hat{p}_1 - \hat{p}_2 \tag{4.4}$$

for the boundary points.

The general solution of (4.3) can be written

$$\hat{U}_j = a_1 U^{(1)}\kappa_1^j + a_2 U^{(2)}\kappa_2^j + a_3 U^{(3)}\kappa_3^j \tag{4.5}$$

where the κ_m's are roots of the characteristic equation

$$\det Q = 0 \tag{4.6}$$

with

$$Q(\kappa) = \begin{bmatrix} \alpha & \hat{\rho}\lambda\sigma(z)(\kappa^2 - 1) & 0 \\ 0 & \alpha & \dfrac{\lambda\sigma(z)}{\hat{\rho}}(\kappa^2 - 1) \\ 0 & \gamma\hat{p}\lambda\sigma(z)(\kappa^2 - 1) & \alpha \end{bmatrix},$$

where

$$\alpha = \kappa\tau(z) + \hat{u}\lambda\sigma(z)(\kappa^2 - 1).$$

The vectors $U^{(m)}$ are solutions of

$$Q(\kappa^m)U^{(m)} = 0, \qquad m = 1, 2, 3.$$

The scalars a_m in (4.5) are determined by the boundary conditions (4.4). We must show that $a_1 = a_2 = a_3 = 0$ for $|z| \geqq 1$ and $|\kappa_m| \leqq 1$, $m = 1, 2, 3$.

It is easily seen that the κ_m's satisfy

$$\begin{aligned} &\tau(z)\kappa_1 + \hat{u}\lambda\sigma(z)(\kappa_1^2 - 1) = 0, \\ &\tau(z)\kappa_2 + (\hat{u} + c)\lambda\sigma(z)(\kappa_2^2 - 1) = 0, \\ &\tau(z)\kappa_3 + (\hat{u} - c)\lambda\sigma(z)(\kappa_3^2 - 1) = 0, \end{aligned} \tag{4.7}$$

where, as before, $|\kappa_m| < 1$ for $|z| > 1$, $m = 1, 2, 3$.

The corresponding eigenvectors are

$$U^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad U^{(2)} = \begin{bmatrix} 1 \\ c/\hat{\rho} \\ c^2 \end{bmatrix}, \quad U^{(3)} = \begin{bmatrix} 1 \\ -c/\hat{\rho} \\ c^2 \end{bmatrix}. \tag{4.8}$$

We note that the three terms in (4.5) are linearly independent when there are multiple eigenvalues κ_m, so that the general solution can always be written in the form (4.5) with the $U^{(m)}$ given by (4.8).

If we substitute (4.5) into (4.4) we obtain the conditions

$$a_1+a_2+a_3=0, \quad a_2-a_3=0, \quad (\kappa_2-1)^2a_2+(\kappa_3-1)^2a_3=0. \tag{4.9}$$

The condition for a nontrivial solution of (4.9) becomes

$$(\kappa_2-1)^2+(\kappa_3-1)^2=0,$$

i.e.,

$$\kappa_2=1\pm i(\kappa_3-1). \tag{4.10}$$

From the last two of the equations (4.7) we obtain

$$\frac{\kappa_2-1}{\kappa_3-1}=\frac{(\hat{u}-c)(\kappa_3+1)\kappa_2}{(\hat{u}+c)(\kappa_2+1)\kappa_3}. \tag{4.11}$$

Let

$$b=\frac{\hat{u}-c}{\hat{u}+c}$$

where $-1<b<0$ since the flow is subsonic inflow. Equations (4.10) and (4.11) imply

$$-(1\pm bi)\kappa_3^2+(1-b\pm 2i)\kappa_3-b(1\mp i)=0. \tag{4.12}$$

If $b=-1$ then (4.12) becomes

$$\kappa_3^2\mp 2i\kappa_3-1=0,$$

which has the double roots $\kappa_3=\pm i$.

We now use a continuity argument letting b increase from $b=-1$. We will see that (4.10) forces κ_2 outside the unit circle and that once κ_2 goes outside the unit circle it can never again be inside it as long as $b<0$. Let

$$b=-1+\delta, \quad 0<\delta\ll 1, \qquad \kappa_3=(i+\varepsilon), \quad |\varepsilon|\ll 1.$$

Consider (4.12) with the upper signs. Then

$$\varepsilon^2-\frac{\delta}{1-i+\delta i}\varepsilon+\frac{(1-i)\delta}{1-i+\delta i}=0,$$

which has the solution

$$\varepsilon=-i\sqrt{\delta}+\frac{1+i}{4}+O(\delta^{3/2}).$$

(The other possible value of ε would yield $|\kappa_3|>1$.) The eigenvalues κ_2 and κ_3 can now be expressed as

$$\kappa_3=i(1-\sqrt{\delta}+O(\delta))+\frac{\delta}{4}+O(\delta^{3/2}),$$

$$\kappa_2=\sqrt{\delta}-i\left(1-\frac{\delta}{4}\right)+O(\delta^{3/2})$$

where

$$|\kappa_2|^2=1+\frac{\delta}{2}+O(\delta^{3/2})>1.$$

Now consider (4.12) with the lower signs. Proceeding in the same way we again find $|\kappa_2|>1$. Thus, if there is to be a nontrivial solution for $|z|\geqq 1$, then κ_2 must pass through the unit circle for some value of b in the interval $-1<b<0$. Let $\kappa_3=x+iy$, where x and y are real. Then

$$\kappa_2=1\mp y\pm i(x-1)$$

subject to

$$|\kappa_2|^2=(x-1)^2+(y\mp 1)^2=1.$$

Figure 4.1 shows the location of κ_2, κ_3 in the (x-y)-plane. We note that κ_2 and κ_3 are always in opposite half-planes. The second and third of the characteristic equations (4.7) yield

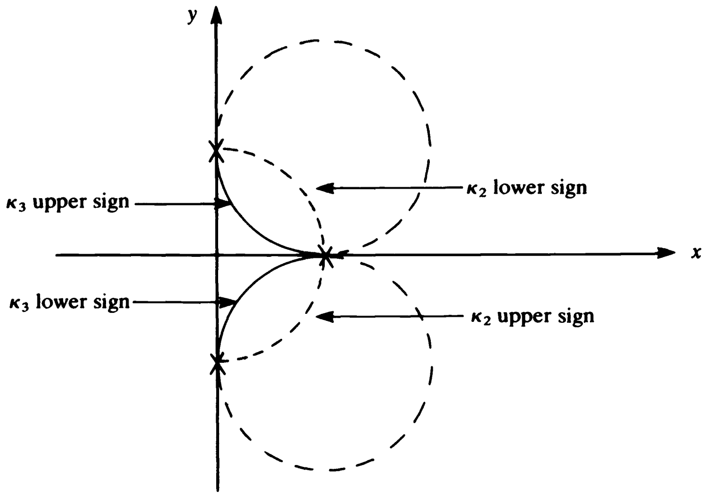


FIG. 4.1

$$\kappa_2^{-1}-\kappa_2=b(\kappa_3^{-1}-\kappa_3).$$

Since $|\kappa_2|=1$ we can write $\kappa_2=e^{i\theta}$ for real θ. If we substitute this above we get

(4.13) $$\kappa_3^{-1}-\kappa_3=i2b^{-1}\sin\theta.$$

Since b is real we have either

$$x^2+y^2=1 \quad\text{and}\quad y=-b^{-1}\sin\theta$$

or

$$x=0 \quad\text{and}\quad y+y^{-1}=-2b^{-1}\sin\theta.$$

It follows from these relations that $\theta=0$ is the only possibility, i.e.,

(4.14) $$\kappa_2=\kappa_3=1.$$

Note that we have been investigating the behavior of κ_2 when κ_3 is inside the unit circle. To conclude this part of the argument we must investigate the other case, i.e., we must exchange the roles of κ_2 and κ_3. The argument goes just as above: we obtain (4.13) with κ_3 replaced by κ_2 and b^{-1} replaced by b. We again arrive at (4.14). From the analysis of the scalar problem in § 2 we know that $\kappa_3=1$ is impossible for $|z|\geqq 1$.

The results for the conservative variable approximations follow immediately by the transformation E of § 3. This concludes the proof of Theorem 4.1.

4.2. Subsonic outflow. We now consider extrapolations in primitive and conservative variables for the subsonic outflow problem. The method (3.5) or its analogous formulation in conservative variables is used at all interior points. The boundary points may be computed through any one of the following methods (stated here in homogeneous form):

$$\rho_0^n = 2\rho_1^n - \rho_2^n, \quad u_0^n = 0, \quad p_0^n = 2p_1^n - p_2^n, \tag{4.15}$$

$$\rho_0^n = 2\rho_1^n - \rho_2^n, \quad u_0^n = 2u_1^n - u_2^n, \quad p_0^n = 0, \tag{4.16}$$

$$\rho_0^n = 2\rho_1^n - \rho_2^n, \quad m_0^n = 2m_1^n - m_2^n, \quad p_0^n = 0. \tag{4.17}$$

THEOREM 4.2. *All of the methods in Table* 2.1 *are stable with any one of the boundary approximations* (4.15), (4.16) *or* (4.17).

Proof. The general form of the solutions to the resolvent equations is again (4.5). The boundary conditions (4.15), (4.16) and (4.17) yield the following conditions on the coefficients a_m, respectively:

$$\begin{aligned} (\kappa_1-1)^2a_1+(\kappa_2-1)^2a_2+(\kappa_3-1)^2a_3&=0,\\ a_2 \qquad -a_3&=0,\\ (\kappa_2-1)^2a_2+(\kappa_3-1)^2a_3&=0; \end{aligned} \tag{4.15a}$$

$$\begin{aligned} (\kappa_1-1)^2a_1+(\kappa_2-1)^2a_2+(\kappa_3-1)^2a_3&=0,\\ (\kappa_2-1)^2a_2-(\kappa_3-1)^2a_3&=0,\\ a_2 \qquad +a_3&=0; \end{aligned} \tag{4.16a}$$

$$\begin{aligned} (\kappa_1-1)^2a_1 \quad +(\kappa_2-1)^2a_2 \quad +(\kappa_3-1)^2a_3&=0,\\ \hat{u}(\kappa_1-1)^2a_1+(\hat{u}+c)(\kappa_2-1)^2a_2+(\hat{u}-c)(\kappa_3-1)^2a_3&=0,\\ a_2 \qquad +a_3&=0. \end{aligned} \tag{4.17a}$$

In the derivation of the second equation in (4.17a) we have used the relation $m = \hat{u}\rho = \hat{\rho}u$.

We find that a nontrivial solution of (4.15a), (4.16a) or (4.17a) must satisfy the same condition we obtained in the proof of Theorem 4.1, equation (4.10), or that $\kappa_1 = 1$. The proof that κ_2 and κ_3 cannot satisfy (4.10) is similar to that given in Theorem 4.1. The parameter $b = (\hat{u}-\hat{c})/(\hat{u}+c)$ now satisfies $-\infty < b < -1$. We again do a perturbation calculation about $b = -1$. There is no root κ_3 of (4.12) which is smaller than one in magnitude near $b = -1$. Therefore, κ_3 must pass through the unit circle for some value of b if this method is to be unstable. However, this cannot happen since the remainder of the argument in Theorem 4.1 goes as before. The only requirement was that b be real and negative.

We now consider the possibility that $\kappa_1 = 1$. If we look at the first of the equations (4.7) and note that $\hat{u}$ is negative, we can conclude that the only possibility of $\kappa_1 = 1$ occurs at $z = \pm 1$. This was examined in § 2 and was found to be impossible for any of the methods in Table 2.1. This completes the proof.

Remark. There is a very good possibility that the results in this section can be extended to a larger class of methods than that of Table 2.1. The reason for this is that up to the point where the condition (4.14) is derived, the proof is independent of $\tau(z)$, $\sigma(z)$. Accordingly, the only way to get an unstable scheme is to define $\tau(z)$, $\sigma(z)$ such that there is a point z_0, $|z_0|=1$ with $\kappa_2(z_0)=\kappa_3(z_0)=1$.

5. The Euler equations with solid wall boundaries. In this section we will treat the system (3.1) with $\hat{u}=0$ and the boundary condition $u=0$. This can be shown to be a well-posed problem for the differential equations using the energy method [5]. We will use the energy method in this section to establish the stability of difference approximations. The normal mode technique we have been using has not been justified for problems of this type—when the boundary is a characteristic surface—for the variable coefficient problem.

We will consider this problem on the bounded interval $0 \leqq x \leqq 1$ using the approximation (4.2) at all interior points $x_j = jh$, with $j=1,\cdots,N-1$ and $N=1/h$. At the boundaries we use one-sided differences, i.e.,

$$\begin{aligned} &\tau(E)\rho_0^n + \frac{k}{h}\sigma(E)\hat{\rho}(u_1^n - u_0^n) = 0, \\ &u_0^n = 0, \\ &\tau(E)p_0^n + \frac{k}{h}\sigma(E)\gamma\hat{p}(u_1^n - u_0^n) = 0 \end{aligned} \tag{5.1}$$

and

$$\begin{aligned} &\tau(E)\rho_N^n + \frac{k}{h}\sigma(E)\hat{\rho}(u_N^n - u_{N-1}^n) = 0, \\ &u_N^n = 0, \\ &\tau(E)p_N^n + \frac{k}{h}\sigma(E)\gamma\hat{p}(u_N^n - u_{N-1}^n) = 0. \end{aligned} \tag{5.2}$$

THEOREM 5.1. *All the methods of Table* 2.1 *are stable with the boundary approximations* (5.1) *and* (5.2).

Proof. The coefficient matrix can be symmetrized using the Abarbanel–Gottlieb symmetrizer [1],

$$S = \begin{bmatrix} \frac{\sqrt{\gamma}}{c}\hat{\rho} & 0 & 0 \\ 0 & 1 & 0 \\ \frac{c}{\sqrt{\gamma}}\hat{\rho} & 0 & \sqrt{\frac{\gamma-1}{\gamma}}\hat{\rho}c \end{bmatrix}.$$

If the new variables $\tilde{\rho}$, $\tilde{u}$ and $\tilde{p}$ are defined by

$$\tilde{U} = \begin{bmatrix} \tilde{\rho} \\ \tilde{u} \\ \tilde{p} \end{bmatrix} = S^{-1}\begin{bmatrix} \rho \\ u \\ p \end{bmatrix},$$

the Euler equations can be written

$$\tilde{U}_t + \tilde{A}\tilde{U}_x = 0$$

where

$$\tilde{A} = \begin{bmatrix} 0 & a & 0 \\ a & 0 & b \\ 0 & b & 0 \end{bmatrix}$$

and $a = c/\sqrt{\gamma}$, $b = c\sqrt{(\gamma-1)/\gamma}$. The approximation can now be written

$$\tau(E)\tilde{U}_j^n + \lambda\sigma(E)\tilde{A}(\tilde{U}_{j+1}^n - \tilde{U}_{j-1}^n) = 0, \qquad j = 1, 2, \cdots, N-1; \tag{5.3a}$$

$$\begin{aligned} &\tau(E)\tilde{\rho}_0^n + \frac{k}{h}\sigma(E)a(\tilde{u}_1^n - \tilde{u}_0^n) = 0, \\ &\tilde{u}_0^n = 0, \\ &\tau(E)\tilde{p}_0^n + \frac{k}{h}\sigma(E)b(\tilde{u}_1^n - \tilde{u}_0^n) = 0; \end{aligned} \tag{5.3b}$$

$$\begin{aligned} &\tau(E)\tilde{\rho}_N^n + \frac{k}{h}\sigma(E)a(\tilde{u}_N^n - \tilde{u}_{N-1}^n) = 0, \\ &\tilde{u}_N^n = 0, \\ &\tau(E)\tilde{p}_N^n + \frac{k}{h}\sigma(E)b(\tilde{u}_N^n - \tilde{u}_{N-1}^n) = 0. \end{aligned} \tag{5.3c}$$

We define the vector

$$\tilde{W} = (\tilde{\rho}_0, \tilde{p}_0, \tilde{\rho}_1, \tilde{u}_1, \tilde{p}_1, \cdots, \tilde{\rho}_{N-1}, \tilde{p}_{N-1}, \tilde{u}_{N-1}, \tilde{\rho}_N, \tilde{p}_N)^T$$

and the matrix

$$\tilde{M} = \lambda \left[\begin{array}{cc|ccc|c|c|c|cc} 0 & 0 & 0 & 2a & 0 & & & & & \\ 0 & 0 & 0 & 2b & 0 & & & & & \\ \hline 0 & 0 & & & & & & & & \\ -a & -b & & 0 & & \tilde{A} & & & & \\ 0 & 0 & & & & & & & & \\ \hline & & & -\tilde{A} & & 0 & \tilde{A} & & & \\ & & & & & \ddots & \ddots & \ddots & & \\ & & & & & -\tilde{A} & 0 & \tilde{A} & & \\ \hline & & & & & & -\tilde{A} & 0 & 0 & 0 \\ & & & & & & & & a & b \\ & & & & & & & & 0 & 0 \\ \hline & & & & & & & \begin{matrix} 0 & -2a & 0 \\ 0 & -2b & 0 \end{matrix} & 0 & 0 \\ & & & & & & & & 0 & 0 \end{array}\right].$$

We can now write our approximations in the form

$$\tau(E)\tilde{W}^n + \tilde{M}\sigma(E)\tilde{W}^n = 0. \tag{5.4}$$

We can make $\tilde{M}$ skew-symmetric by the transformation

$$R = \operatorname{diag}(\sqrt{2}, \sqrt{2}, 1, 1, \cdots, 1, 1, \sqrt{2}, \sqrt{2}),$$

i.e.,

$$M = R^{-1}\tilde{M}R$$

is skew-symmetric. Therefore, the matrix M can be diagonalized by a unitary transformation Q such that

$$Q^*MQ = D$$

is a diagonal matrix with pure imaginary eigenvalues. After the transformations Q and R, the system (5.4) becomes a collection of scalar equations of the form

$$\tau(E)w_\nu^n + d_\nu \sigma(E) w_\nu^n = 0, \qquad \nu = 1, 2, \cdots, 3N+1, \tag{5.5}$$

where the d_ν are pure imaginary scalars. We are only considering methods which are A-stable for ordinary differential equations, hence

$$|w_\nu^n| \leqq K(|w_\nu^0| + |w_\nu^1|)$$

where K is independent of n and ν. Since the condition numbers of Q, R and S are bounded independently of N, we obtain a stability estimate for the original system (4.2) with boundary conditions (5.1), (5.2).

REFERENCES

[1] S. ABARBANEL AND D. GOTTLIEB, *Optimal time splitting for two and three dimensional Navier–Stokes equations with mixed derivatives*, ICASE Rep. 80-6, NASA, Langley, VA, 1980.

[2] R. BEAM AND R. WARMING, *An implicit factored scheme for the compressible Navier–Stokes equations II: The numerical ODE connection*, AIAA 4th Computational Fluid Dynamics Conference, Williamsburg, Virginia, 1979, Paper 79-1446.

[3] B. GUSTAFSSON, H.-O. KREISS AND A. SUNDSTRÖM, *Stability theory of difference approximations for mixed initial boundary value problems*, II. Math. Comp., 26 (1972), pp. 649–686.

[4] B. GUSTAFSSON, The convergence rate for difference approximations to mixed initial boundary value problems, Math. Comp., 29 (1975), pp. 396–406.

[5] J. OLIGER AND A. SUNDSTRÖM, *Theoretical and practical aspects of some initial boundary value problems in fluid dynamics*, SIAM J. Appl. Math., 35 (1978), pp. 419–446.

[6] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, 2nd ed., Interscience, New York, 1967.

[7] G. SKÖLLERMO, *How the boundary conditions affect the stability and accuracy of some implicit methods for hyperbolic equations*, Rep. 62, Dept. of Computer Sci., Uppsala University, Uppsala, Sweden, 1975.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

WHY PARTICLE METHODS WORK

J. J. MONAGHAN*

Abstract. The theme of this paper is that particle methods are closely related to both finite difference and spectral methods because the three methods can be considered special cases of interpolation by kernel estimation. The kernels for a number of special cases are given in detail, and the accuracy of the resulting interpolation is analyzed. A general procedure for deriving equations for numerical work from the equations of hydrodynamics is described. It is applied to the derivation of the SPH equations which conserve linear and angular momentum exactly.

Key words. particle methods, numerical hydrodynamics

1. Introduction. Although a meal can be enjoyed without understanding the processes of digestion, numerical methods should be both understood and enjoyed. This requirement is not merely the whim of a tidy mind, for a method once understood can often be improved with little effort. The standard methods of fluid dynamics (e.g. finite difference and spectral methods) meet this requirement since they are based on well established approximation methods for functions, and their stability is at least partially understood through the use of von Neumann's method. Particle methods are in a much less satisfactory state. They give a good description of a wide variety of phenomena in plasma physics (Birdsall and Fuss (1969)) in astrophysical fluid dynamics (Lucy (1977), Gingold and Monaghan (1977), (1978), (1982)) and galaxy simulation (Aarseth (1972)), but the equations of motion are derived by intuitive methods which bear little resemblance to well-known processes of analysis.

In this paper an attempt will be made to correct this discrepancy by showing firstly that the way a function is represented in particle methods is a special case of interpolation using information from a set of points. The interpolation procedure is based on kernel estimation, and includes those processes which involve linear operations on the function of interest. Orthogonal function and local polynomial approximation are typical examples. It will be shown, using the same formalism, that general equations for numerical work can be derived without specifying the details of the interpolation method. In this way, the very close relation between finite difference, spectral and particle methods becomes obvious.

2. Kernel estimation. The kernel estimate of the function $f(\mathbf{r})$ in the domain D is defined by

$$f_K(\mathbf{r}) = \int_D W(\mathbf{r}, \mathbf{r}', h) f(\mathbf{r}')\, d\mathbf{r}' \tag{2.1}$$

where the function W is a kernel with the property that

$$\lim_{h\to 0} \int_D W(\mathbf{r}, \mathbf{r}', h)\, d\mathbf{r}' = 1. \tag{2.2}$$

The idea behind (2.1) is that if W can be so chosen that it mimics $\delta(\mathbf{r}-\mathbf{r}')$ then $f_K(\mathbf{r})$ will be close to $f(\mathbf{r})$. Generally h will be chosen so that

$$f_K(\mathbf{r}) \to f(\mathbf{r}) \quad \text{as } h \to 0. \tag{2.3}$$

* Institute of Astronomy, Madingley Road, Cambridge CB3 0HA, England. Permanent address: Department of Mathematics, Monash University, Clayton, Victoria 3168, Australia.

For example, we could choose

$$W(\mathbf{r}, \mathbf{r}', h) = \frac{\exp[-(\mathbf{r}-\mathbf{r}')^2/h^2]}{(\pi h^2)^{3/2}}. \tag{2.4}$$

In general, as in (2.4), h will be defined so that it determines the extent to which W confines the major contribution to f_K to the neighbourhood of $\mathbf{r}' = \mathbf{r}$. We shall refer to h as the smoothing length.

For the one-dimensional case with $0 < x < 1$, the choice

$$W\left(x, x', \frac{1}{n}\right) = \sqrt{\frac{n}{\pi}}\,[1-(x-x')^2]^n, \tag{2.5}$$

where n is an integer, is the basis of Landau's proof (see, e.g., Hobson (1926, Chap. IV)) of Weierstrass's approximation theorem (proved by Weierstrass using the one-dimensional version of (2.4)).

Now assume, as in the particle methods, that we have a set of N points $\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N$ distributed in space according to the number density $n(\mathbf{r})$. We can evaluate (2.1) approximately in the form

$$f_K(\mathbf{r}) = \sum_{j=1}^{N} \frac{W(\mathbf{r}, \mathbf{r}_j, h) f(\mathbf{r}_j)}{n(\mathbf{r}_j)}, \tag{2.6}$$

with an error that depends on the degree to which the points are disordered (see § 6). If $f(\mathbf{r})$ is the mass density $\rho(\mathbf{r})$, and if $\rho(\mathbf{r}) = mn(\mathbf{r})$, which is the usual case in the particle methods (where m is the mass of each particle), then (2.6) becomes

$$\rho_K(\mathbf{r}) = m \sum_{j=1}^{N} W(\mathbf{r}, \mathbf{r}_j, h). \tag{2.7}$$

This expression is normally interpreted by saying that the density of a point mass at $\mathbf{r}_j$ is smeared or smoothed out according to the smoothing function $W(\mathbf{r}, \mathbf{r}_j, h)$, and ρ is the sum of such smoothed point masses.

We now want to show that the procedure leading from (2.1) to (2.6) or (2.7) has an exact analogue in interpolation processes based on orthogonal function expansions or local polynomial approximations.

If we interpolate a function in $-\pi < x < \pi$ by using its Fourier expansion truncated at the Nth term (Natanson (1961)) the resulting expression can be written as a kernel estimate

$$f_K(x) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\sin[(2N+1)(x-x')/2]}{\sin[(x-x')/2]} f(x')\,dx'. \tag{2.8}$$

A variety of other kernels can be used (e.g. Fejer's, Jackson's or Poisson's) depending on the way the limit of the Fourier sum is defined. Other orthogonal function expansions can also be written as kernel estimates with kernels that may be written in a compact form using the Christoffel–Darboux formula (Appendix).

If the function values are given at the points $x_j = j\pi/N$ we can write (2.8) in the form

$$f_K(x) = \frac{1}{2N} \sum_{j=-N+1}^{N} f(x_j) \frac{\sin[(2N+1)(x-x_j)/2]}{\sin[(x-x_j)/2]}, \tag{2.9}$$

which is exact for functions in the form of a truncated Fourier expansion. We can interpret (2.9), as we did (2.7), in terms of the smoothing of the function values about the points where they are defined. The particle methods and orthogonal function interpolation are therefore similar—they can both be based on kernel estimates and they both involve smoothing operations. They only differ in the type of kernel used and the errors involved in going from the integral expression (2.1) to its approximation by a summation. Of these differences the first is trivial since the particle methods can use arbitrary kernels (we could, for example, use the kernel in (2.8) in the one-dimensional version of (2.7) to produce a hybrid particle-spectral interpolation). The second of the differences is the more important and we discuss it further in § 6.

When interpolating by orthogonal functions it is natural to examine the sum of the truncated series which then leads naturally to the kernel estimate. Representations of the form (2.7) are then a convenience for numerical work. The reverse is true for local polynomial interpolation since the natural representation is a summation similar to (2.7) while the kernel representation is a convenience for the analysis.

All the standard local polynomial interpolation formulae can be converted from summations to kernel representations of the form (2.1) and their relation to other interpolation methods seen clearly. As a simple example consider the function $f(x)$ given at equi-spaced points $x_j = jh$ $(j = 0, \pm 1, \pm 2, \cdots)$. The interpolation function is

$$\tilde{f}(x) := h \sum_{-\infty}^{\infty} f(x_j) L(x - x_j, h), \tag{2.10}$$

where

$$L(u, h) = \begin{cases} (1 - |u|/h), & |u| \leqq h, \\ 0, & |u| \geqq h. \end{cases} \tag{2.11}$$

(If the interpolation is only required in $a < x < b$, the only points that can contribute to (2.10) are those in $a - h < x < b + h$.)

The form of (2.10) suggests that we define the kernel estimate equivalent to the linear interpolation (2.10), to be

$$f_K(x) = \int_{-\infty}^{\infty} L(x - x', h) f(x')\, dx'. \tag{2.12}$$

Since L has argument $x - x'$ and the domain is $-\infty < x < \infty$, the integral in (2.12) can be evaluated with negligible error by the trapezoidal rule and (2.10) is recovered. This example shows that linear interpolation can be considered as a kernel estimate with a special kernel.

If higher accuracy local polynomial interpolation is required, it is necessary to use polynomials of higher degree. The kernels for Bessel's interpolation (up to and including second differences) and the symmetric Lagrangian interpolation formula are given in the Appendix.

The various examples described here and in the Appendix show that the procedures used to estimate functions in particle methods are a disguised form of the standard interpolation methods used in numerical analysis. The major difference between the methods is that spectral or local polynomial interpolation normally uses a fixed set of ordered points while the particle method interpolates from a moving set of points which are usually disordered.

3. How to derive equations for numerical work. The procedure we adopt to construct equations suitable for numerical work is the following. Each of the exact equations is multiplied by a kernel and then integrated over the solution domain.

Integration by parts, possibly with further approximations to deal with nonlinear terms, then produces equations for the kernel estimates.

As a simple example, consider the one-dimensional continuity equation when the velocity v is a constant in space:

$$\frac{\partial \rho}{\partial t}+v\frac{\partial \rho}{\partial x'}=0, \qquad -\infty<x'<\infty, \tag{3.1}$$

with the initial condition

$$\rho(x',0)=G(x').$$

The first step of the procedure produces the equation

$$\int_{-\infty}^{\infty} W(x-x',h)\left[\frac{\partial \rho}{\partial t}+v\frac{\partial \rho}{\partial x'}\right]dx'=0. \tag{3.2}$$

Integration by parts shows that

$$\frac{\partial \rho_K}{\partial t}+v\frac{\partial \rho_K}{\partial x}=0, \tag{3.3}$$

which is to be solved with the interpolated initial condition

$$\rho_K(x,0)=\int_{-\infty}^{\infty} W(x-x',h)G(x')\,dx'. \tag{3.4}$$

Equation (3.3) is exact regardless of the kernel we use in the kernel estimate.

If we use the linear interpolation estimate (2.10), then $\rho_k(x_i)=\rho(x_i)$, and with the derivative defined from the right, (3.3) becomes

$$\frac{\partial}{\partial t}\rho(x_i)+\frac{v[\rho(x_{i+1})-\rho(x_i)]}{h}=0. \tag{3.5}$$

If we use the particle method with the Gaussian kernel, ρ_k becomes

$$m\sum_{j=1}^{N}\frac{\exp[-(x-x_j)^2/h^2]}{h\sqrt{\pi}}, \tag{3.6}$$

and the continuity equation (3.3) is identically satisfied at each particle because $dx_a/dt=v$ and then

$$\frac{d}{dt}\rho_K(x_a)=0,$$

which is (3.1).

The advantages of the procedure described are that (a) in the linear case if the exact equations are satisfied then so are the equations for the kernel estimates, and (b) since the kernel need not be specified until the last stage, the relationships between different methods are obvious. The procedure has the further advantage that when h is allowed to vary in space and with time, valid equations can be derived without difficulty.

4. The particle equations for a fluid. The example shown in § 3 was linear, and the equation of motion for the kernel estimates was exact. We now consider fluid dynamical equations where the existence of nonlinear terms will require further approximation. To keep the analysis simple we shall confine our attention to nondissipative motion in a gravitational field and use kernel estimates for the particle method (SPH) which does not employ a grid. The exact equations are

$$\frac{d}{dt}\mathbf{v} = -\frac{1}{\rho}\nabla P - \nabla\Phi, \tag{4.1}$$

$$\nabla^2\Phi = 4\pi G\rho, \tag{4.2}$$

$$\frac{d\rho}{dt} + \rho\nabla\cdot\mathbf{v} = 0. \tag{4.3}$$

If we multiply (4.1) by the kernel and integrate over the solution domain assuming that h is constant in space and time, the acceleration term becomes

$$\left(\frac{d\mathbf{v}}{dt}\right)_K = \left(\frac{\partial\mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v}\right)_K = \frac{\partial}{\partial t}\mathbf{v}_K + ((\mathbf{v}\cdot\nabla)\mathbf{v})_K. \tag{4.4}$$

The nonlinear term can be approximated according to

$$((\mathbf{v}\cdot\nabla)\mathbf{v})_K \doteqdot \mathbf{v}_K\cdot(\nabla\mathbf{v})_K = (\mathbf{v}_K\cdot\nabla)\mathbf{v}_K, \tag{4.5}$$

where integration by parts has been used to rewrite the combination $\nabla\mathbf{v}$. We assume here, and in the remainder of the paper, that either W or the variable or both vanish on the boundary. This is the case for astrophysical hydrodynamics where the boundaries can be taken at infinity. Other problems may, however, introduce surface terms, both in (4.4) and in the force terms.

The errors in (4.5) are comparable to the truncation error, and are therefore consistent with the kernel estimate. The acceleration term can therefore be taken as

$$\frac{\partial}{\partial t}\mathbf{v}_K + (\mathbf{v}_K\cdot\nabla)\mathbf{v}_K, \tag{4.6}$$

and in the particle method this is

$$\frac{d}{dt}\mathbf{v}. \tag{4.7}$$

The pressure term can be written in various forms, but there are two which are attractive because they result in exact linear and angular momentum conservation. The first from results from noting

$$\frac{\nabla P}{\rho} = \nabla\left(\frac{P}{\rho}\right) + \frac{P}{\rho^2}\nabla\rho, \tag{4.8}$$

so that, with approximations which are obvious,

$$\left(\frac{\nabla P}{\rho}\right)_K = \nabla\left(\frac{P}{\rho}\right)_K + \frac{P_K}{\rho_K^2}\nabla\rho_K. \tag{4.9}$$

Referring now to (2.6) and using subscripts to specify function values at particles gives

$$\left(\frac{P}{\rho}\right)_K = m\sum_{j=1}^{N}\frac{P_j}{\rho_j^2}W(\mathbf{r}-\mathbf{r}_j), \tag{4.10}$$

so that the right-hand side of (4.9), when evaluated at particle i, becomes

$$m\sum_{j=1}^{N}\left(\frac{P_j}{\rho_j^2}+\frac{P_i}{\rho_i^2}\right)\nabla_i W(\mathbf{r}_i-\mathbf{r}_j, h), \tag{4.11}$$

where ∇_i means that the gradient operator acts on the variable $\mathbf{r}_i$.

If $W(\mathbf{u}, h)$ is an even function of $\mathbf{u}$, then $\nabla_i\, W(\mathbf{r}_i - \mathbf{r}_j, h)$ will be anti-symmetric in i and j. Accordingly

$$\sum_i \sum_j \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2}\right) \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h) = 0, \tag{4.12}$$

and the contribution of the pressure to the rate of change of the total momentum vanishes.

The second form for the $\nabla P/\rho$ term follows from the relation

$$\frac{\nabla P}{\rho} = \frac{2\sqrt{P}}{\rho} \nabla(\sqrt{P}), \tag{4.13}$$

so that

$$\left(\frac{\nabla P}{\rho}\right)_K = \frac{2\sqrt{P_K}}{\rho_K} \nabla(\sqrt{P})_K. \tag{4.14}$$

At particle i, the right-hand side of (4.14) becomes

$$2m \sum_{j=1}^{N} \frac{\sqrt{P_i}}{\rho_i} \frac{\sqrt{P_j}}{\rho_j} \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h), \tag{4.15}$$

and the contribution of the pressure to the rate of change of the total momentum vanishes. It is easy to show that the total angular momentum also vanishes. These results are of course obvious when it is recognized that the forms we choose for $\nabla P/\rho$ are equivalent to specifying forces between particles which act parallel to the line joining their centres.

The gravitational force term becomes

$$-\nabla \Phi_K, \tag{4.16}$$

with

$$\nabla^2 \Phi_K = 4\pi G \rho_K. \tag{4.17}$$

The latter equation follows from (4.2) using the kernel procedure.

The solution to (4.17) is

$$\Phi_K(\mathbf{r}) = -Gm \sum_{j=1}^{N} \int \frac{W(\mathbf{r} - \mathbf{r}_j, h)}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}', \tag{4.18}$$

so that

$$\Phi_K(\mathbf{r}) = -Gm \sum_{j=1}^{N} K(|\mathbf{r} - \mathbf{r}_j|, h), \tag{4.19}$$

where K is defined by the integral in (4.18).

This procedure for Φ_K is exact though not necessarily the most efficient. It has been used in SPH because useful results in astrophysics can be achieved with $N \sim 10^3$ when direct summation is acceptable. The gravitational force computed from (4.19) clearly leads to linear and angular momentum conservation.

Combining (4.7), (4.11), (4.16) and (4.19) we find

$$\frac{d^2\mathbf{r}_i}{dt^2} = -m \sum_{j=1}^{N} \left(\frac{P_j}{\rho_j^2} + \frac{P_i}{\rho_i^2}\right) \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h) + Gm \sum_{j=1}^{N} \nabla_i K(|\mathbf{r}_i - \mathbf{r}_j|, h), \tag{4.20}$$

which is the equation of motion for particle i in the SPH method.

When we apply the kernel procedure to the continuity equation (4.3) it becomes

$$\frac{\partial}{\partial t}\rho_K + \nabla \cdot (\rho \mathbf{v})_K = 0,$$

so that

$$\frac{d}{dt}\rho_K + \nabla \cdot (\rho \mathbf{v})_K - \mathbf{v}_K \cdot \nabla \rho_K = 0. \tag{4.21}$$

At particle i, (4.21) can be written

$$\frac{d}{dt}\rho_K(\mathbf{r}_i) + m\sum_{j=1}^{N} (\mathbf{v}_j - \mathbf{v}_i) \cdot \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h) = 0, \tag{4.22}$$

and this equation is identically satisfied since, from its definition, $d\rho_k/dt$ is

$$m\sum_{j=1}^{N} \frac{d}{dt} W(\mathbf{r}_i - \mathbf{r}_j, h).$$

With this particle method the continuity equation is satisfied exactly locally, that is at each particle, and total mass is conserved exactly.

5. Energy conservation. We have already noted that, for consistency, the kernel estimate should be used for both the equations and the boundary conditions. The appropriate form of the energy equation results if we also apply the kernel procedure to the energy equation. For the example considered in § 4 the thermal energy equation is

$$\frac{de}{dt} = \frac{-P}{\rho}\nabla \cdot \mathbf{v}. \tag{5.1}$$

The kernel procedure results in

$$\frac{de_K}{dt} = -\left(\frac{P}{\rho}\nabla \cdot \mathbf{v}\right)_K. \tag{5.2}$$

The nonlinear term in (5.2) can be written in various ways, but consistent energy conservation only results if it is written in a way which corresponds to the form of the pressure term in the equation of motion. Recalling the steps that led to (4.11), we write (5.2) in the form

$$\frac{de_K}{dt} = -\left(\nabla \cdot \left(\frac{P\mathbf{v}}{\rho}\right) + \mathbf{v} \cdot \nabla\left(\frac{P}{\rho}\right)\right)_K \tag{5.3}$$

which, with the approximation used in (4.9), is

$$\frac{de_K}{dt} = -\nabla \cdot \left(\frac{P\mathbf{v}}{\rho}\right)_K + \mathbf{v}_K \cdot \nabla\left(\frac{P}{\rho}\right)_K. \tag{5.4}$$

If (5.4) is evaluated at particle i, and summed over all particles, the total rate of change of thermal energy is found to be

$$\frac{d}{dt}\sum_i (e_K)_i = m\sum_i\sum_j \left\{\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right\}\mathbf{v}_i \cdot \nabla_i W(\mathbf{r}_i - \mathbf{r}_j, h) \tag{5.5}$$

so that (see (4.11))

$$\frac{d}{dt}\sum_i (e_K)_i = \sum_i \mathbf{v}_i \cdot \left(\frac{\nabla P}{\rho}\right)_{K,i},$$

and the work done by the pressure forces is, as expected, at the expense of thermal energy. In the derivation of (5.5) we have assumed $W(\mathbf{u}, h)$ is an even function of $\mathbf{u}$.

If (4.15) is used in the momentum equation, then (5.2) should be written in the form

$$\frac{de_K}{dt} = \left(-\frac{\sqrt{P}}{\rho}\nabla\cdot(\mathbf{v}\sqrt{P}) + \frac{\sqrt{P}(\mathbf{v}\cdot\nabla\sqrt{P})}{\rho}\right)_K, \tag{5.6}$$

which results in

$$\frac{d}{dt}\sum_i (e_K)_i = 2m\sum_i\sum_j \frac{\sqrt{P_iP_j}}{\rho_i\rho_j}\mathbf{v}_i\cdot\nabla_i W(\mathbf{r}_i-\mathbf{r}_j, h).$$

The gravitational energy Eg is defined by

$$Eg := -\frac{G}{2}\int\int\frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\,d\mathbf{r}'\,d\mathbf{r} = \frac{1}{2}\int\rho(\mathbf{r}')\Phi(\mathbf{r}')\,d\mathbf{r}', \tag{5.7}$$

and we define its smoothed value for the particle methods as

$$(Eg)_K := \frac{1}{2}m\sum_j \Phi_K(\mathbf{r}_j). \tag{5.8}$$

Referring to (4.19), we find from (5.8) that

$$\frac{d}{dt}(Eg)_K = m\sum_j \mathbf{v}_j\cdot\nabla_j\Phi_K(\mathbf{r}_j), \tag{5.9}$$

and the work done by the gravitational forces is at the expense of the gravitational energy.

By taking the scalar product of $\mathbf{v}_i$ and (4.20), and summing over all values of i, we find

$$\frac{d}{dt}\sum_i\frac{1}{2}\mathbf{v}_i^2 = -\frac{d}{dt}\sum_i(e_K)_i - \frac{d}{dt}(Eg)_K \tag{5.10}$$

so that, with a consistent treatment of the energy terms, the method conserves energy.

The derivation of the energy equation for the case considered here is a simple exercise. If, however, the smoothing length is allowed to vary with time, then the procedure we have adopted allows correct account to be taken of this time variation.

6. Accuracy of kernel estimates.

If the integration for the kernel estimate can be carried out with no error, the accuracy of the kernel estimate is determined by the kernel. If we use orthogonal function interpolation the error can be expressed (formally) in terms of the terms omitted from the truncated series. For the other kernels we have considered, the accuracy can be estimated by expanding the function $f(x')$ in the integrand of (2.1) about the point $\mathbf{r}'=\mathbf{r}$. We shall assume that the Taylor expansion is valid, and work in one dimension for simplicity since the results are easily generalized.

With the assumptions stated,

$$f_K(x) = \int_{-\infty}^{\infty} W(u,h)\left\{f(x) - uf'(x) + \frac{u^2}{2!}f''(x) - \cdots\right\}du. \tag{6.1}$$

It is convenient to choose

$$\int_{-\infty}^{\infty} W(u,h)\,du = 1, \tag{6.2}$$

rather than require it to be true in the limit $h \to 0$ as in (2.2). The error terms involving odd powers of u vanish if $W(u, h)$ is an even function of u: the usual choice for particle methods. Any kernel which is an even function of its argument is therefore at least equivalent to linear interpolation.

To achieve higher accuracy it is necessary to choose kernels for which successive even moments (excluding the 0th) vanish. The Bessel interpolation (kernel B in the Appendix) has the property that (6.2) is satisfied and

$$\int_{-\infty}^{\infty} u^k B(u, h)\, du = 0 \quad \text{for } k = 1, 2, 3, \tag{6.3}$$

so that

$$f_K(x) = f(x) + O(h^4).$$

For kernels belonging to the Gaussian family, higher accuracy can be achieved by noting (for one dimension) that

$$\delta(u) = \exp\left(-\frac{h^2}{4}\frac{d^2}{du^2}\right) e^{-u^2/h^2} \frac{1}{h\sqrt{\pi}}. \tag{6.4}$$

Expanding the derivative operator gives rise to a succession of kernels

(i) $$e^{-u^2/h^2} \frac{1}{h\sqrt{\pi}},$$

(ii) $$e^{-u^2/h^2} \frac{1}{h\sqrt{\pi}}\left(\frac{3}{2} - \frac{u^2}{h^2}\right),$$

and so on. The first kernel achieves linear interpolation. The second kernel is equivalent to the Bessel kernel $B(u, h)$. In several dimensions (6.4) is generalized by replacing d^2/du^2 by ∇^2.

Interpolation by either orthogonal functions or by local polynomial interpolation allows the integration for f_K to be carried out either exactly or with an error that is comparable to the error of the interpolation. If the points are disordered, as is the case for particle methods, then the integration is not exact. Estimates of the error are sometimes made by describing the approximation of

$$\int W(x - x', h) f(x')\, dx'$$

by

$$\sum_{j=1}^{N} \frac{W(x - x_j, h) f(x_j)}{n(x_j)}$$

as a Monte Carlo estimate. If this were true then the error would be $\propto 1/\sqrt{N}$, and it could dominate the total error. In practice (Gingold and Monaghan (1978)) the errors are always much less than the Monte Carlo estimate. The reason for this seems to be that when the initial state is ordered, and fluctuations are low, the system of particles becomes disordered, but the disorder does not produce large fluctuations because large fluctuations require too much energy. An exception occurs when the physics of the problem produces an instability. Even in this case the error estimate is more likely to be that based on disordered equi-distributed numbers (Niederreiter (1978)) that is $O(\log N/N)$.

Experience with the particle method SPH shows that results with errors <5 percent can be achieved with $\sim 10^3$ particles in nonaxisymmetric fluid problems (Gingold and Monaghan (1981)). This clearly shows that in stable flow the particle form of the equations of fluid dynamics show no tendency to greatly amplify an initial field of fluctuations. An analytic proof of this conjecture would be a valuable contribution to the understanding of particle methods.

Provided that the number of particles used is sufficiently large, the integration for the kernel estimate will be accurate enough to warrant using kernels which remove more of the higher order even moments. This accuracy should be achieved in current plasma simulation when $>20{,}000$ particles are normally employed.

7. Discussion and conclusions. The principal conclusion of this paper is that particle methods work because they make use of interpolation methods which are closely related to standard interpolation methods. Of course in practice other aspects of the particle methods contribute to their effectiveness; for example, they automatically satisfy local matter conservation and they transport angular momentum accurately.

Apart from establishing the connection between various interpolation methods, the kernel formalism provides a safe and convenient procedure for constructing equations suitable for numerical work. In this paper we have limited the analysis to those problems where a constant smoothing length is useful, but in more complicated problems it is necessary to allow h to vary with time (Gingold and Monaghan) or with both time and position (Wood (1981)). The kernel formalism shows that, in the former case, time derivatives of h appear in the energy equation. In the latter case time and space derivatives appear, and in Wood's formulation the latter have been omitted. These problems will be discussed elsewhere.

By working with the kernel formalism, attention is naturally directed towards choosing the kernel to have desirable properties. We have shown how the kernel should be chosen to reduce the interpolation error. Other requirements can be easily introduced. For example the kernel can be chosen so that all functions can have a specified number of derivatives, or chosen so that particles are prevented from coalescing.

The kernels currently used in plasma CIC methods smooth the density and then area-weight to the mesh. This procedure produces an effective kernel (Gingold and Monaghan (1982)) to which the considerations of this paper can be applied.

Appendix. In this appendix we collect further examples of kernels associated with interpolation.

(a) *Bessel's interpolation formula.* Scale x so that the function $f(x)$ is specified at $x=0, \pm 1, \pm 2$, etc., and consider Bessel's interpolation for $f(x)$ in $0<x<1$. When 4th and higher differences are neglected,

$$\begin{aligned}\tilde{f}(x) &= (1-x)f(0)+xf(1)+\tfrac{1}{4}x(x-1)[f(2)-f(1)-f(0)+f(-1)]\\ &= f(0)[1-x-\tfrac{1}{4}x(x-1)]+f(1)[x-\tfrac{1}{4}x(x-1)]\\ &\quad +f(2)\tfrac{1}{4}x(x-1)+f(-1)\tfrac{1}{4}x(x-1).\end{aligned} \tag{A.1}$$

The coefficient of $f(0)$ shows that the kernel, if it is an even function (as we assume), must have the form

$$(1-|x|)(1+\tfrac{1}{4}|x|) \tag{A.2}$$

for $0 \leqq |x| \leqq 1$. Examination of the other coefficients shows that, for $1 \leqq |x| \leqq 2$, it has the form

(A.3) $$\tfrac{1}{4}(1-|x|)(2-|x|),$$

and is zero for $|x| \geqq 2$. With this kernel the interpolation formula becomes

(A.4) $$\tilde{f}(x) = \sum_{j=-\infty}^{\infty} f(j)B(x-j).$$

If interpolation is required in the domain $a < x < b$, points in $a-2h < x < b+2h$ are required. The summation in (A.4) can be replaced by integration since the error is no greater than the interpolation error.

(b) *Lagrangian interpolation.* We assume the data points are equi-spaced and there are always enough to allow n-point Lagrangian interpolation. To keep the analysis simple we take $n=4$ and assume unit spacing.

To interpolate $f(x)$ we use 4 points $k-1$, k, $k+1$ and $k+2$ where k is the largest integer $\leqq x$. Then

(A.5) $$\tilde{f}(x) = \sum_{j=k-1}^{k+2} f(j)L_{4,j}(x),$$

where $L_{4,j}(x)$ is a standard Lagrangian interpolation function. We wish to write (A.5) in the form

(A.6) $$\tilde{f}(x) = \sum_{j=-\infty}^{\infty} f(j)W(x-x_j).$$

By comparing (A.6) and (A.5) and noting that, in (A.5)

$$k \leqq x \leqq k+1,$$

we deduce that $W(u)$ takes the form stated below for the specified domain.

(A.7)

Domain	$W(u)$
$1 \leqq u \leqq 2$	$L_{4,k-1}(u+k-1)$
$0 \leqq u \leqq 1$	$L_{4,k}(u+k)$
$-1 \leqq u \leqq 0$	$L_{u,k+1}(u+k+1)$
$-2 \leqq u \leqq -1$	$L_{4,k+2}(u+k+2)$

The kernel is symmetric because

$$L_{4,k}(-u+k) = L_{4,k+1}(u+k+1)$$

and

$$L_{4,k-1}(-u+k-1) = L_{4,k+2}(u+k+2).$$

If the number of interpolation points $n \neq 4$, the above procedure can be easily generalized. If $n \to \infty$ we find Whittaker's cardinal function (Hartree (1958))

$$C(x) = \sum_{j=-\infty}^{\infty} f(j)\frac{\sin \pi(x-j)}{x-j}.$$

(c) *Interpolation with Legendre polynomials.* For $-1 < x < 1$ we can use the approximation

(A.8) $$\tilde{f}(x) = \sum_{j=0}^{n} a_j P_j(x)$$

where $P_j(x)$ is the jth order Legendre polynomial and

$$a_j = \frac{2j+1}{2} \int_{-1}^{1} f(x')P_j(x')\,dx'. \tag{A.9}$$

Hence

$$\tilde{f}(x) = \int_{-1}^{1} f(x') \sum_{j=0}^{n} \frac{2j+1}{2} P_j(x)P_j(x')\,dx' \tag{A.10}$$

which is a kernel estimate with

$$W(x,x') = \sum_{j=0}^{n} \frac{2j+1}{2} P_j(x)P_j(x'). \tag{A.11}$$

The Christoffel–Darboux theorem then shows

$$W(x,x') = \frac{n+1}{2} \frac{[P_{n+1}(x)P_n(x') - P_n(x)P_{n+1}(x')]}{x-x'}. \tag{A.12}$$

The kernel is peaked at $x = x'$ and is symmetric in x and x', but it is not a function of $(x-x')$. This creates analytical complications with derivatives since

$$\int_{-1}^{1} \frac{df}{dx'} W(x,x')\,dx' = [f(x')W(x,x')]_{-1}^{1} - \int_{-1}^{1} f(x') \frac{d}{dx'} W(x,x')\,dx'. \tag{A.13}$$

The second term on the right-hand side of (A.12) cannot be written

$$\frac{d}{dx} f_K,$$

as it can when W is a function of $(x-x')$.

REFERENCES

S. Aarseth (1972), *Gravitational N-body Problem*, M. Lecar, ed., D. Reidel, Dordrecht.

D. K. Birdsall and D. Fuss (1969), *Clouds-in-clouds, clouds-in-cells, physics for many body plasma simulation*, J. Comp. Phys., 3, pp. 494–511.

R. A. Gingold and J. J. Monaghan (1977), *Smoothed particle hyrdodynamics: Theory and application to non spherical stars*, Mon. Not. Roy. Astr. Soc., 181, pp. 375–389.

——— (1978), *Binary fission in damped rotating polytropes*, Mon. Not. Roy. Astr. Soc., 184, pp. 481–499.

——— (1982), *Kernel estimates as a basis for particle methods*, J. Comp. Phys., in press.

D. R. Hartree (1958), *Numerical Analysis*, Clarendon Press, Oxford.

E. W. Hobson (1926), *Theory of Functions of a Real Variable*, Vol. II, p. 460. Cambridge University Press, Cambridge.

L. Lucy (1977), *A numerical approach to testing the fission hypothesis*, Astron. J., 82, pp. 1013–1024.

I. P. Natanson (1961), *Theory of Functions of a Real Variable*, Vol. 2, F. Ungar, New York.

H. Niederreiter (1978), *Quasi-Monte Carlo methods and pseudo-random numbers*, Bull. Amer. Math. Soc., 84, pp. 957–1041.

D. Wood (1981), *Collapse and fragmentation of isothermal gas clouds*, Mon. Not. Roy. Astr. Soc., 194, pp. 201–218.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

A METHOD FOR COMPUTING THE INTEGRAL OF THE BIVARIATE NORMAL DISTRIBUTION OVER AN ARBITRARY POLYGON*

A. R. DiDONATO† AND R. K. HAGEMAN†

Abstract. An efficient programmable procedure is given for evaluating the integral of the bivariate normal distribution (IBND) over an arbitrary polygon Π. The class of arbitrary polygons includes the subclasses: simple polygons, limit elements of sequences of uniformly bounded N-sided simple polygons with the same orientation, and self-intersecting (SI) polygons. For a given element Π defined by N ordered points in the plane, the subclass need not be specified. The method evaluates the IBND over N exterior angular regions $A_1, A_2, \cdots, A_N$ of Π to determine the IBND for Π. If Π is SI, a quantity called the "winding number" of Π is introduced which is given by the sum of the angular measures of the A_i $(i = 1, 2, \cdots, N)$ divided by 2π. A detailed numerical example, using a Fortran IV program, with approximately 9-decimal-digit accuracy is included.

Key words. statistical functions, bivariate normal distribution, probability over polygon

1. Introduction.

The objective of this paper is to give a procedure for computing the integral P of the bivariate normal density function $Z(w, z)$ over an arbitrary polygon[1] $\bar{\Pi}$, where

$$Z(w,z) \equiv [2\pi\sigma_w\sigma_z(1-\rho^2)^{1/2}]^{-1} \tag{1}$$
$$\cdot \exp\left\{-\left[\left(\frac{w-\mu_w}{\sigma_w}\right)^2 - 2\rho\left(\frac{w-\mu_w}{\sigma_w}\right)\left(\frac{z-\mu_z}{\sigma_z}\right) + \left(\frac{z-\mu_z}{\sigma_z}\right)^2\right]2/(1-\rho^2)\right\},$$

with (μ_w, μ_z) the mean and

$$\begin{bmatrix} \sigma_w^2 & \rho\sigma_w\sigma_z \\ \rho\sigma_z\sigma_w & \sigma_z^2 \end{bmatrix}$$

the covariance matrix of the normal random variable (w, z) with correlation coefficient ρ. In an earlier work [1], [2] an efficient method was given for the evaluation of

$$P(\bar{\Pi}) = \int\int_{\bar{\Pi}} Z(w,z)\, dw\, dz \tag{2}$$

for $\bar{\Pi} = C$, where C denotes a convex polygon. Here we shall extend those ideas for the efficient computation of (2) over arbitrary polygons. Greater detail is given in [3].

Transforming the variable (w, z) to the new variable (x, y) by the linear transformation (3) of [2] reduces the integrand of (2) to a simpler one with circular symmetry, i.e.,

$$P(\Pi) = \frac{1}{2\pi}\int\int_{\Pi} \exp[-(x^2+y^2)/2]\, dx\, dy, \tag{3}$$

where $\bar{\Pi}$ and Π have the same structure, namely if $\bar{\Pi}$ is convex, simple, or self-intersecting, then Π has the same property, respectively. Hereafter we deal only with (3).

* Received by the editors April 9, 1981.

† Naval Surface Weapons Center, Dahlgren, Virginia 22448.

[1] A polygon or polygonal element will always mean a finite closed continuous line, made up of a finite number of line segments, and its interior. Its boundary is specified by $N+1$ ordered points in the plane, where the first and last points coincide.

The approach used to find (3) over a convex polygon C of N vertices was equivalent to evaluating the integral in (3) over an exterior angular region at each vertex of C. An angular region is defined as the semi-infinite part of the plane bounded by two intersecting directed straight lines. Since there are generally four such regions, it is always necessary to specify which one is needed. An angular region A can be specified by the distance R of its vertex V from the origin and the angles θ_1 and θ_2 which the two sides 1 and 2 of A make with the extension L of the straight line passing through the origin and V. The angles are measured positive in the counterclockwise direction about V from L. This is shown in [2, Fig. 1].

Taking advantage of the circular symmetry of the integrand in (3), a rotation of axes and a translation of the origin are made so that the new x-axis is along L and the new origin is at the vertex of A. Introduction of polar coordinates centered at the new origin ($x = R + r\cos\theta$, $y = r\sin\theta$) yields an expression for P over an angular region, as given in [1], [2], i.e.,

$$(4)\qquad P(A) = e^{-R^2/2}\left\{\frac{\Delta\theta}{2\pi} - \frac{1}{\pi}\int_{\theta_1}^{\theta_2} u[\operatorname{erfc}(u)/z(u)]\,d\theta\right\},$$

where

$$(5)\qquad \begin{aligned} &\Delta\theta = \theta_2 - \theta_1 \text{ (for polygons } |\Delta\theta| \leqq \pi),\\ &u = (R/\sqrt{2})\cos\theta,\\ &z(u) = (2/\sqrt{\pi})\exp(-u^2),\\ &\operatorname{erfc}(u) = \int_u^\infty z(t)\,dt. \end{aligned}$$

The function $\operatorname{erfc}(u)/z(u)$ in (4) is approximated, for $u \geqq 0$ ($\cos\theta \geqq 0$) by a polynomial in u obtained by a minimax procedure. The integral can then be evaluated by recurrence relations which are given in [1], [2]. The coefficients of the polynomial are listed in [1] for 3, 6, 9 and 12 digit accuracy. When $\cos\theta_j < 0$, $j = 1, 2$, the additional relation

$$(6)\qquad P[A(R, 0, \theta_j)] = \tfrac{1}{2}\operatorname{erfc}\left(\frac{R}{\sqrt{2}}\sin\theta_j\right) - P[A(R, 0, \pi - \theta_j)],$$

is used. The details are given in [1]. For computing efficiency, when R is sufficiently large or small $P(A)$ is evaluated directly by approximating expressions [1, pp. 6–8]. Also, note that when $R = 0$, $P(A) = (\theta_2 - \theta_1)/2\pi$.

We have then that IBND over a convex polygon is given by[2]

$$(7)\qquad P(C) = 1 - \sum_{i=1}^{N} P(A_i),$$

where A_i is the exterior angular region of C at vertex i as shown in Fig. 1. The vertices $\{(i)\}$, $i = 1, \cdots, N$, are given in counterclockwise order.

We proceed to extend our results for convex polygons by obtaining a relation similar to (7) for simple polygons.

2. Simple polygons. We say a simple polygon S is positively oriented (PO) if its vertices (1), (2), $\cdots$, (N) are given in counterclockwise order, i.e., the interior of S

[2] This result for $P(C)$ is expressed in slightly different form in [1], [2].

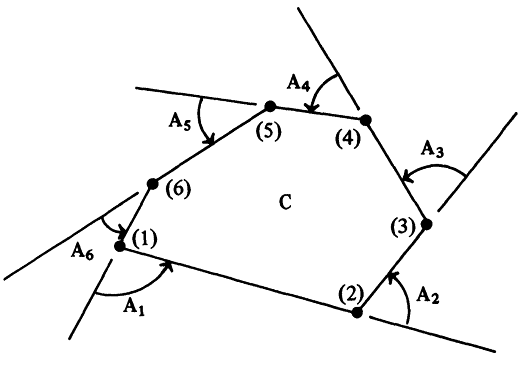


FIG. 1. *Angular regions* $A_1, \cdots, A_6$ *for* C, $N = 6$.

is on the left as the boundary is continuously transversed in the direction of the increasing successive (i). If the interior is on the right, we say S is negatively oriented (NO).

In this section we shall show that, just as for convex polygons, the integral of (3) can be evaluated over any N-sided simple polygon S by computing $P(A_i)$, (4), for N angular regions, where the vertex V for each A_i is located at (i). In fact when S is PO then

$$P(S) = 1 - \sum_{i=1}^{N} P(A_i). \tag{8}$$

A glance shows that (7), for $P(C)$, and (8), for $P(S)$, are the same. In (7) each $P(A_i)$ is positive since $0 < \Delta\theta_i < \pi$ (assuming C is PO). In (8), however, this will not be the case if the interior angle at (i) exceeds π radians. For example, in Fig. 2 the interior angle at (3) exceeds π, so that A_3 is measured in the clockwise rather than the counterclockwise direction. Hence $-\pi < \Delta\theta_3 < 0$ for A_3 and, from (6) of [2], $P(A_3) < 0$. Note however that $P(A_i) > 0$, $i = 1, 2, 4$.

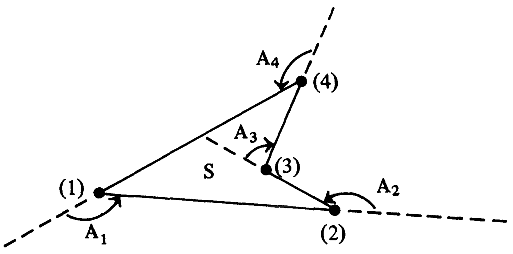


FIG. 2. *Polygon* S, *angular regions* A_i, $i = 1, 2, 3, 4$.

The truth of (7) for convex polygons is apparent. In the case of (8), we give a heuristic argument for its validity which can be made rigorous.

The argument is inductive. Certainly (8) holds for $N = 3$. Some insight is gained by considering $N = 4$ with S not convex, say as in Fig. 2. We see there that $1 - P(S)$ is obtained by considering $\sum_1^4 P(A_i)$, where $P(A_3)$, which is negative, compensates exactly for the excessive positive contribution from $P(A_2)$. Thus (8) holds for $N = 4$.

Now assume (8) is true for $N = J - 1$, $J \geqq 4$. We want to show (8) holds for $N = J$. We look at the special case $J = 8$ with Fig. 3, since the essentials of a more lengthy rigorous proof are contained in the arguments for this case.

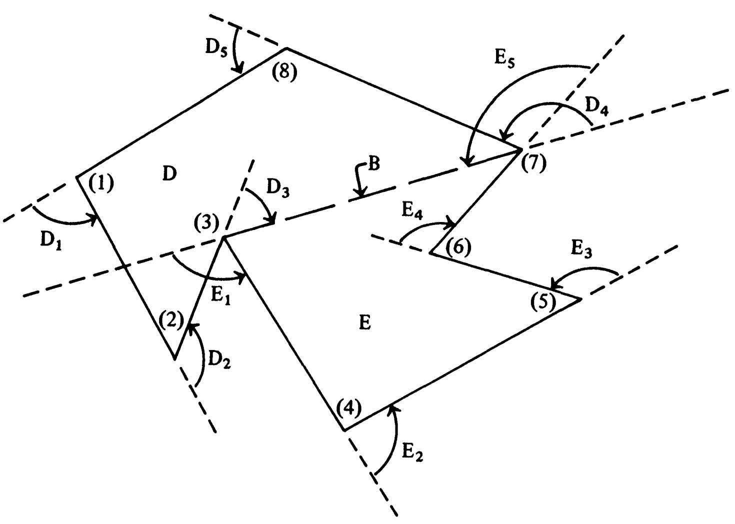


FIG. 3. *Angular regions of simple polygons D and E.*

First, a diagonal B is drawn from vertex (3) to vertex (7) which remains inside S. Such a line can always be found for any simple polygon; a proof of this fact is given in [3, Appendix B]. The line B separates S into two simple disjoint polygons (except for their common boundary B) with the same orientation as S. Call the separated polygons D and E. Each has fewer than J vertices. From Fig. 3, D has vertices at (1), (2), (3), (7), (8). E has vertices at (3), (4), (5), (6), (7). By assumption, (8) holds for both D and E, so that, with D_i and E_i, $i = 1, 2, \cdots, 5$, the angular regions of D and E, respectively

$$(9) \qquad P(D) = 1 - \sum_{i=1}^{5} P(D_i), \quad P(E) = 1 - \sum_{i=1}^{5} P(E_i), \quad P(D) + P(E) = P(S).$$

Now with A_i, $i = 1, 2, \cdots, 8$, the angular regions associated with S, we have

$$(10) \qquad D_1 = A_1, \quad D_2 = A_2, \quad D_5 = A_8, \quad E_2 = A_4, \quad E_3 = A_5, \quad E_4 = A_6.$$

Then with angular regions D_3, D_4, E_1, E_5 as shown in Fig. 3, we also have,

$$(11) \qquad \begin{aligned} P(A_3) &= P(D_3) - [P(B) - P(E_1)] < 0, \\ P(A_7) &= P(E_5) + P(D_4) - [1 - P(B)] > 0, \end{aligned}$$

where $P(B)$ denotes the (positive) probability over the half-plane *below* the extended line B.

From (9)

$$(12) \qquad P(S) = P(D) + P(E) = 2 - \sum_{i=1}^{5} P(D_i) - \sum_{i=1}^{5} P(E_i).$$

Now, using (10) and (11) in (12), we have

$$(13) \qquad \begin{aligned} P(S) &= 2 - P(A_1) - P(A_2) - [P(A_3) + P(B) - P(E_1)] - [P(A_7) - P(E_5) + 1 - P(B)] \\ &\quad - P(A_8) - P(E_1) - P(A_4) - P(A_5) - P(A_6) - P(E_5) \\ &= 1 - \sum_{i=1}^{8} P(A_i). \end{aligned}$$

This completes the argument based on Fig. 3. In order to make the proof rigorous, it is necessary to consider the other possibilities for the angular regions at the two vertices of S on the diagonal B. In the case of Fig. 3, the interior angle at (3) was greater than π and the one at (7) was less than π. The three other possibilities were checked; the arguments for these cases require nothing new and are omitted.

Up to this point it has been assumed that the vertices of a C or S polygon were given in counterclockwise order. In the next section it will be necessary to allow for NO simple polygons, i.e., where the vertices are specified in clockwise order.

If S is PO, then (4) and (8) are designed so that $P(S)>0$. Hence we require when S is NO that $P(S)<0$. Using this fact and taking into account the sign change for $\sum_1^N P(A_i)$ when the orientation of S is reversed leads directly, using (8), *when S is* NO to

$$P(S)=-1-\sum_{i=1}^{N} P(A_i). \tag{14}$$

Note A_{N+2-i} from (8) and A_i from (14) are vertical angles with their angular measures of opposite sign, with $(N+1)\equiv(1)$.

If a limit element $\bar{S}$ is obtained from a sequence, $\{S_n(N)\}$, of uniformly bounded N-sided simple polygons of PO (NO), then $\bar{S}$ is considered PO (NO) and by a simple continuity argument (8) ((14)) holds for P($\bar{S}$).

Our program for computing $P(S)$ must determine whether the $+1$ of (8) or the (-1) of (14) is correct which, of course, means the program must decide whether S is PO or NO. In the next section we show how this decision is made. The correct choice can also be made by computing the signed area of S, $A(S)$. This follows because the signed area of S (or of any polygon) can be expressed in terms of vector cross-products which depend on the orientation of S (see [3, Appendix D] or [5]). Thus if $A(S)>0$, then S is PO and (8) is used to yield a positive value of $P(S)$; if $A(S)<0$, then S is NO and (14) is used to obtain $P(S)<0$.

A very efficient formula, which is derived in [3], for computing the signed area of any polygon Π is given by

$$A(\Pi)=\frac{1}{2}\sum_{i=1}^{N} x_i(y_{i+1}-y_{i-1}), \qquad y_0\equiv y_N, \quad y_{N+1}\equiv y_1, \tag{15}$$

where (x_i, y_i) denotes the xy-coordinate location of the ith numbered point of Π. The absolute value of $A(\Pi)$ represents the "area" of Π provided Π is in $\{\bar{S}\}$, but just as for the P function, if Π is SI then the interpretation of $A(\Pi)$ rests on the analyst. Our program yields $A(\Pi)$ as an auxiliary output.

3. Arbitrary polygons. We arrive at the class of arbitrary polygons by first extending the class $\{S\}$ to include limit elements of sequences $\{S_n(N)\}$ of uniformly bounded simple polygons of N vertices with the same orientation. We designate this class by $\{\bar{S}\}$ and an element of the class by $\bar{S}$, where by previous remarks $P(\bar{S})$ is given by (8) or (14) depending on whether $\bar{S}$ is PO or NO. This class is then enlarged to include SI polygons, and we designate it as the class of arbitrary polygons $\{\Pi\}$, where an element of the class is denoted by Π. Thus, we have

$$\{C\}\subseteq\{S\}\subseteq\{\bar{S}\}\subseteq\{\Pi\}. \tag{16}$$

A simple example of an element in $\{\bar{S}\}$ and of one in $\{\Pi\}$, but not in $\{\bar{S}\}$, are shown in Figs. 4 and 5 respectively.

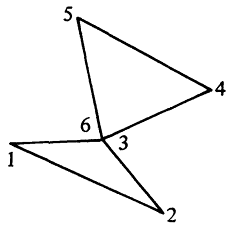


FIG. 4. *An element in* $\{\bar{S}\}$.

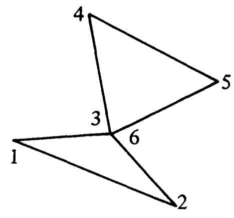


FIG. 5. *An* SI *polygon.*

Our objective is to show that the integral in (3) can be evaluated for any element in $\{\Pi\}$ by computing the $P(A_i)$, as expressed by (4), for the N exterior angular regions $\{A_i\}$ of that element. Significantly, it will not be necessary to specify to which of the four classes the element belongs.

It is assumed that the polygon under consideration is sequentially numbered at all of the following points: vertices (a vertex is defined below), where two segments meet or cross at a point, and initial and terminal ends of overlapping segments. The numbering, as the element is traced, is in the natural order of the integers, starting at one, i.e., starting at some point (1), each time a situation as described is met it is numbered in sequential order until Π is completely traced, where the last point $(N+1)$, as noted earlier, coincides with the first point. The set of $(N+1)$ xy-points at which Π is numbered is denoted by G. A reduced numbering scheme can sometimes be used for actual computations as described in [3] and noted in the numerical example at the end of the paper.

In order to establish our results below, it is necessary to define exactly what is meant by an SI element. We say Π is SI if it is not in $\{\bar{S}\}$, i.e., if it cannot be a limit element of $\{S_n(N)\}$. Before characterizing this property, we introduce some definitions and notation.

The jth *node*, (j), associates the integer j with the jth xy-point of the ordered point set G which defines Π. The set G is also denoted by $(1, 2, \cdots, j-1, j, j+1, \cdots, N, N+1)$ with $(N+1)=(1)$. Let the jth *edge* of Π, denoted by $\bar{j}$, be the directed line segment of Π originating at (j) and terminating at $(j+1)$, so that $\overline{j-1}$ terminates and $\bar{j}(\bar{0}\equiv\bar{N})$ begins at (j). We say $\overline{j-1}$ and $\bar{j}$ are *associated* with the jth node. If only one node occurs at an xy-point, it is called a *simple node*, (SN). Of course, more than one node can exist at the same point, in which case that point is called a *multiple node*, (MN). We identify a particular MN by MN (j), where (j) refers to the first node met at that xy-point, i.e., $j \leqq k$, where (k) is any node at that MN. In Figs. 4 and 5 we have MN (3). A *vertex* j of Π is a point of G such that $\overline{j-1}$ and $\bar{j}$ have different slopes. We define a *path* $[m, n]$ of Π as a line made up of consecutive edges $\bar{m}, \overline{m+1}, \cdots, \overline{n-1}, \bar{n}, m \leqq n$.

Certainly a polygonal element is SI if the path $[k-1, k]$, formed by its two edges associated with (k) at an *MN crosses* another such path $[j-1, j]$ at the same *MN*. By

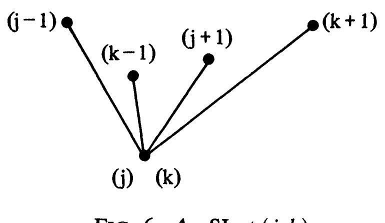


FIG. 6. *An* SI *at* (j, k).

cross, we mean pass through rather than just meet. In this case, we say (j, k) is a *self-intersection point* of Π, or simply an *intersection point*, and that the element is SI at (j, k). Such a situation is shown in Fig. 6. If two paths just meet at an MN, rather than cross, and have no other points in common, as shown in Fig. 7, then (j, k) is not an intersection point. Using these notions, it is easy to see that the polygon of Fig. 5 is SI at (3, 6), whereas the element in Fig. 4 is not SI.

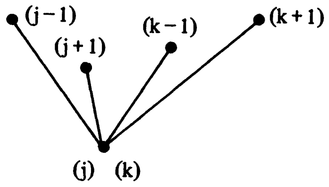


FIG. 7. *Not* SI *at* (j, k).

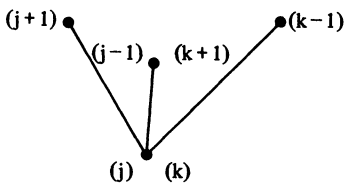


FIG. 8. SI *at* (j, k) *indeterminate.*

In addition to the path configurations of Figs. 6 and 7 more subtle configurations can occur such as those in Fig. 8 and in the polygons of Figs. 9a and 9b. In such cases (j, k) may or may not be an SI point, which points out the need for a more precise characterization of an SI polygon. We proceed with that as an objective.

With MN (j) denoting the MN of Π at (j), where j is the first node numbered at that point, let $J(j, \delta_j)$ represent a disk centered at (j) with radius δ_j, where δ_j is chosen so small that $J(j, \delta_j)$ intersects only those edges which originate or terminate at MN (j). We also refer to such a disk as a J-disk.

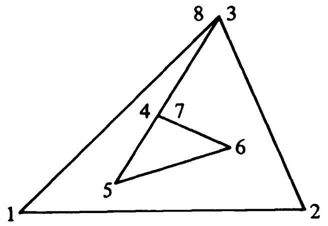


FIG. 9a. *An* SI *polygon at* (3, 8).

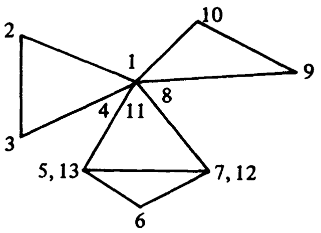


FIG. 9b. *A polygon in* $\{\bar{S}\}$.

The approach now is to construct a polygon T "close" to Π and *simple*, if possible. If T is simple, then Π is in $\{\bar{S}\}$. If T cannot be constructed as a simple polygon, then Π is not in $\{\bar{S}\}$ and we say it is SI and in $\{\Pi\}$.

To proceed with the construction of T, consider two successive nodes (k), $(k+1)$ with $1 \leqq k \leqq N$ and $(N+1) \equiv (1)$. Let T_k denote the edge of T which is to be taken close to edge $\bar{k}$. This segment is constructed as follows:

(A) $T_k = \bar{k}$ if (k) and $(k+1)$ are SN.

(B) $T_k = B_k$ if (k) is an SN and $(k+1)$ is at MN (j), $1 \leqq j \leqq k+1$, where B_k emanates from (k) and terminates at a point s_{k+1} in $J(j, \delta_j)$. If $k = N$, then require $s_{N+1} = (N+1) = (1)$, $T_N = \bar{N}$, so that T is closed.

(C) $T_k = D_k$ if (k) is at MN (i) and $(k+1)$ is an SN, $1 \leqq i \leqq k$, where D_k emanates from s_k, a point in $J(i, \delta_i)$, and terminates at $(k+1)$. If $i = k = 1$, then $s_1 = (1)$ so that T will be closed at (1).

(D) $T_k = L_k$ if (k) is at MN (i) and $(k+1)$ is at MN (j), $i \neq j$, $1 \leqq i \leqq k$, $1 \leqq j \leqq k+1$, where L_k emanates from s_k in $J(i, \delta_i)$ and terminates at s_{k+1} in $J(j, \delta_j)$. If $k = N$, then $j = 1$ and $s_{N+1} = (N+1) = (1)$ so that T is closed. For the same reason, if $i = k = 1$ then $s_1 = (1)$.

This construction is carried out for each $k=1,2,\cdots,N-1,\ N$ to obtain T. Clearly, by choosing the δ radii sufficiently small T can be obtained arbitrarily close to Π. Now if the s_k can be chosen for any δ_i so that T is simple, then by choosing a sequence of δ_i approaching zero, a sequence of T's can be constructed which make up $\{S_n(N)\}$ converging to Π. Hence in this case Π is in $\{\bar{S}\}$. If this cannot be done, i.e., if in some J-disk paths of T must intersect (i.e., actually cross), then Π is SI, since it cannot be obtained as the limit of a sequence $\{S_n(N)\}$. If an intersection takes place in $J(j,\delta_j)$ between paths $[T_{k-1},T_k]$ and $[T_{k+m-1},T_{k+m}]$, we say Π has an intersection point at $(k,k+m)$.

By this characterization of $\bar{S}$ and SI elements, the polygons in Figs. 4 and 5 are, as noted before $\bar{S}$ and SI, respectively. In Fig. 10a a typical T-construction is shown for the element of Fig. 9a. Clearly it is SI at (3, 8), because L_7 must intersect $T_2=\bar{2}$ ot $T_3=\bar{3}$ in order to join L_8 at s_8.[3] On the other hand, the element in Fig. 9b is in $\{\bar{S}\}$ since as Fig. 10b shows T can be constructed as a simple polygon.

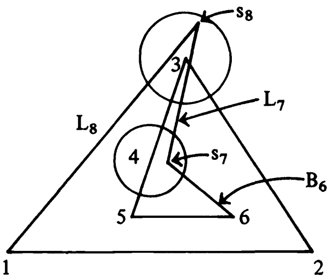


FIG. 10a. *Typical T-construction for polygon of Fig.* 9a.

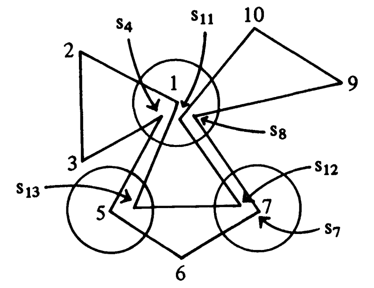


FIG. 10b. *Typical T-construction for polygon of Fig.* 9b.

SI polygons should not appear often in practice. If, however, the generation of a polygon is not under control of the analyst, say the nodes are computer assigned, then SI polygons can occur. (See [3, Fig. 20].) Negative values of P, as well as $|P|>1$, are possible for such elements and, as in the case for NO simple polygons, proper interpretation of such results is required of the analyst.

The main result of the paper can now be expressed by one equation. For any element Π of $\{\Pi\}$, with $N+1$ numbered points, $(N+1)=(1)$, (3) is given by

$$P(\Pi)=W-\sum_{i=1}^{N}P(A_i), \tag{17}$$

[3] We have L_8 instead of D_8, because we assume every polygon has a multiple node at (1) in the sense that $(1)=(N+1)$.

where $P(A_i)$ denotes the value of P for the ith exterior angular region of Π, as defined earlier (see Figs. 1 and 2), and W is a new quantity, which we call the *winding number* of Π and define below. Thus there are two steps to evaluating $P(\Pi)$ for any polygon. The first is to evaluate the $P(A_i)$ for each $i = 1, 2, \cdots, N$; the second is to compute the winding number of Π. The computation of the $P(A_i)$ no longer offers any difficulty. It has been summarized earlier and is discussed extensively in [1], [2], [3]. We shall indicate below that W is obtained by simply adding up the N angular measures $\Delta\theta_i$ of the A_i and dividing the result by 2π. Thus for an element in $\{S\}$ that is PO (NO), $W = 1(-1)$ which gives agreement with (8) ((14)). This follows since the sum of angular measures of the A_i for a PO (NO) simple polygon is $2\pi(-2\pi)$. By a continuity argument the results also hold for elements of $\{\bar{S}\}$.

In order to establish (17) for elements in $\{\Pi\}$, we need a few additional symbols and definitions.

A *circuit* C of Π is a closed path of Π with no self-intersections. Thus a circuit is in $\{\bar{S}\}$, and its first and last points are located at the same MN. In case Π is in $\{\bar{S}\}$, then Π is a circuit; its first and last points (1) and $(N+1)$ are at MN (1).

The *primary circuit* (PC) of Π, denoted by $C_p(\Pi)$ is the first circuit detected, in tracing Π from (1), which closes at $(k+m)$, where $(k, k+m)$ is the first intersection point encountered with paths $[k-1, k]$ and $[k+m-1, k+m]$ intersecting at MN (j). We have $j \leqq k < k+m$ with $C_p(\Pi) = (k, k+1, \cdots, k+m-1, k+m)$. This circuit includes all nodes $(k+i)$ at MN (j), where $0 \leqq i \leqq m$. If Π is in $\{\bar{S}\}$, then $C_p(\Pi) = \Pi$.

We use Fig. 9b and a reordering of some of its nodes to cite two examples of primary circuits. If nodes (12) and (13) are interchanged then Π is SI and $C_p(\Pi) = (11, 12, 13, 14)$, where $j = 1$, $k = 11$, $k+m = 14 = N+1$. The first intersection point is at (11, 14). If in Fig. 9b nodes (5) and (7) are interchanged, then $C_p(\Pi) = (4, 5, 6, 7, 8)$. Later we shall refer to these modifications of Fig. 9b as 9b(1) and 9b(2), respectively.

Now let $\Pi_1 \equiv \Pi$. Then decompose Π into PC(s) as follows:

(α) Obtain $C_p(\Pi_1)$. Set $i = 1$ and go to (δ).

(β) Find $C_p(\Pi_i)$ and go to (δ).

(γ) Π has been decomposed into a set of disjoint elements[4] of $\{\bar{S}\}$. The decomposition is complete. If $i = K$, we say Π has been decomposed into K primary circuits.

(δ) If $C_p(\Pi_i) = \Pi_i$ go to (γ). Otherwise, delete $C_p(\Pi_i)$ from Π_i.[5] Call the result Π_{i+1}, i.e., $\Pi_{i+1} = \Pi_i - C_p(\Pi_i)$. Set $i = i+1$ and go to (β).

Clearly this decomposition of Π into primary circuits can always be carried out.[6] Referring to Fig. 9a and cases 9b(1) and 9b(2), three examples are given of decompositions into primary circuits. For Fig. 9a, we have

$$C_p(\Pi_1) = (3, 4, 5, 6, 7, 8), \qquad C_p(\Pi_2) = (1, 2, 8, 9).$$

(Note that if (5) and (6) are interchanged Π is in $\{\bar{S}\}$.) For 9b(1) (nodes 12 and 13 of Fig. 9b are interchanged) we have

$$C_p(\Pi_1) = (11, 12, 13, 14), \qquad C_p(\Pi_2) = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 14)$$

[4] Two elements of Π are disjoint if neither has more than one of its nodes in common with the other. A set of elements of Π are disjoint if they are disjoint pairwise.

[5] The deletion of $C_p(\Pi_i)$ from Π_i means all nodes of $C_p(\Pi_i)$, except the last, are dropped from Π_i. The resulting set of nodes specifies Π_{i+1}.

[6] Another decomposition of Π, found in Knopp's constructive proof that every polygon can be decomposed into union of a set of simple polygons and a set of overlapping line segments [4] is discussed and used in Appendix A of [3]. A one-to-one correspondence between Knopp's decomposition of Π and that by primary circuits can be made.

and for 9b(2) (nodes 5 and 7 of Fig. 9b are interchanged)

$$C_p(\Pi_1)=(4,5,6,7,8), \qquad C_p(\Pi_2)=(8,9,10,11),$$
$$C_p(\Pi_3)=(11,12,13,14), \qquad C_p(\Pi_4)=(1,2,3,14).$$

In general, with Π decomposed into K PC (s), we have

$$\Pi=\bigcup_{i=1}^{K} C_p(\Pi_i). \tag{18}$$

Now since the PC (s) are disjoint

$$P(\Pi)=\sum_{1}^{K} P[C_p(\Pi_i)]. \tag{19}$$

Also the $C_p(\Pi_i)$ are in $\{\bar{S}\}$, hence from (8) and (14)

$$P[C_p(\Pi_i)]=W_i-\sum_{n=1}^{N_i} P(A_{in}), \qquad W_i=\begin{cases}1 & \text{if } C_p(\Pi_i) \text{ is PO,}\\ -1 & \text{if } C_p(\Pi_i) \text{ is NO.}\end{cases} \tag{20}$$

Here A_{in} denotes the nth angular region of $C_p(\Pi_i)$ and N_i+1 denotes the number of nodes specifying $C_p(\Pi_i)$. Substituting (20) into (19) gives

$$P(\Pi)=\sum_{i=1}^{K} W_i-\sum_{i=1}^{K}\sum_{n=1}^{N_i} P(A_{in}). \tag{21}$$

The *winding number of* Π is defined to be

$$W=\sum_{1}^{K} W_i. \tag{22}$$

Thus, for the polygons of Figs. 9a and 9b, and for cases 9b(1) and 9b(2), $W=2, 1, 2, 0$, respectively.

Now let

$$\Omega\equiv\sum_{1}^{N}\Delta\theta_i, \qquad |\Delta\theta_i|\leqq\pi, \tag{23}$$

where $\Delta\theta_i$ appears as the first term on the right-hand side of (4) and denotes, as usual, the angular measure in radians of the exterior angular region A_i of Π.[7] It is necessary to show that

$$\sum_{r=1}^{N} P(A_r)=\sum_{i=1}^{K}\sum_{n=1}^{N_i} P(A_{in}), \tag{24a}$$

$$W=\Omega/2\pi. \tag{24b}$$

We present the elements of a proof. Suppose $C_p(\Pi_1)=(k, k+1, \cdots, k+m)$ so that $\Pi_2=(1, 2, \cdots, k-1, k+m, \cdots, N, N+1)$, where $\Pi_2=\Pi-C_p(\Pi_1)$ (see footnote 5). Denote the exterior angles of $C_p(\Pi_1)$ and (Π_2) at (k) by $A_{1,1}$ and $A_{2,k}$, respectively. Denote their corresponding angular measures by ζ and λ. Also recall $\Delta\theta_k$ and $\Delta\theta_{k+m}$ denote the measures of A_k and A_{k+m} of Π, respectively.

In the particular case of Fig. 11 below we have

$$\zeta+\lambda=\Delta\theta_k+\Delta\theta_{k+m}. \tag{25}$$

[7] Note that since $\Delta\theta_i$ is needed in (4), Ω is available with only N additions.

From (25) and the way $A_{1,1}$, $A_{2,k}$, A_k, A_{k+m} are defined it follows directly, in that case, that

$$P(A_{1,1})+P(A_{2,k})=P(A_k)+P(A_{k+m}). \tag{26}$$

In fact, a straightforward geometrical or analytical argument shows (25) and (26) are always true.

Assume now that Π has only one self-intersection. Then the only angular regions affected by the decomposition of Π to $C_p(\Pi_1)\cup\Pi_2$ are at (k) and $(k+m)$. Hence, from (19) and (20) with $C_p(\Pi_1)$ and Π_2 in $\{\bar{S}\}$

$$\begin{aligned} P(\Pi)&=P[C_p(\Pi_1)]+P(\Pi_2)\\ &=W_1-\left[P(A_{1,1})+\sum_{k+1}^{k+m-1}P(A_r)\right]+W_2 \\ &\quad-\left[\sum_{1}^{k-1}P(A_r)+P(A_{2,k})+\sum_{k+m+1}^{N}P(A_r)\right]. \end{aligned} \tag{27}$$

By (22) and (26), the basic result given by (17) follows.

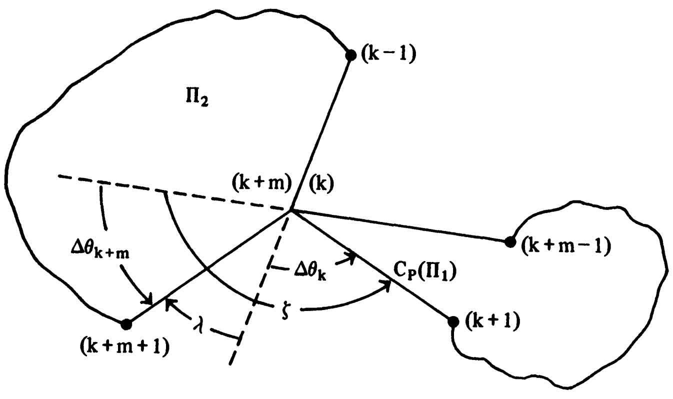


FIG. 11. *Parts of* $C_p(\Pi_1)$ *and* Π_2 *of* SI *polygon at* $(k, k+m)$.

In the case of one self-intersection, it remains to show that (24b) holds. We have the winding numbers W_1 for $C_p(\Pi_1)$ and W_2 for Π_2 given by

$$2\pi W_1=\zeta+\sum_{r=k+1}^{k+m-1}\Delta\theta_r,\qquad 2\pi W_2=\lambda+\sum_{r=1}^{k-1}\Delta\theta_r+\sum_{r=k+m+1}^{N}\Delta\theta_r. \tag{28}$$

We have used the fact that since Π has only one self-intersection $C_p(\Pi_2)=\Pi_2$. Now using (22) and (25), we get

$$W=W_1+W_2=\frac{1}{2\pi}\left[\zeta+\lambda+\sum_{\substack{r=1\\(r\neq k,k+m)}}^{N}\Delta\theta_r\right]=\frac{1}{2\pi}\sum_{r=1}^{N}\Delta\theta_r=\frac{\Omega}{2\pi}. \tag{29}$$

An induction argument can be used to treat the case where Π decomposes into $K(>2)$ PC elements. Assume (17) holds for all polygons Π which are decomposable into no more than $K-1$ PC elements. The essence of a proof that (17) holds for elements decomposable into K PC elements is obtained from the argument above for $K=2$.

Indeed, let Π have K such circuits. Then by the decomposition procedure described above

$$\Pi = C_p(\Pi_1) \cup \Pi_2, \tag{30}$$

where Π_2 can be decomposed into $K-1$ PC elements with winding numbers $W_2, W_3, \cdots, W_K$. But, by the induction hypothesis (17) holds for Π_2. Therefore, the remainder of the argument goes as above for $K = 2$, where W_2 is replaced by $\sum_{i=2}^{K} W_i$ in (27), (28), (29).

Numerical results for a variety of configurations are given in [3]. A Fortran IV computer program has been developed which finds $P(\Pi)$ and $A(\Pi)$ (see (15)) to approximately 3, 6 or 9 decimal-digit-accuracy, provided the xy-points which specify Π are given to the same accuracy. The numerical results were checked by an independent computing program which decomposed Π into a set of separate triangles. For each triangle, taking its orientation into account, the value of P was found, using (8) or (14), where probabilities over angular regions were computed by an independent procedure. The P values for the set of triangles were properly combined to yield $P(\Pi)$. The details are given in [3].

In Table 1 numerical results correct to approximately 9 decimal digits are shown for an SI polygon, Π. The polygon is numbered so that $P(\Pi)$ represents the probability for an event, governed by a normal bivariate distribution, occurring in S_1 and/or S_2, where $S_1 = (1, 2, 3, 4, 5, 6, 7, 8)$, $S_2 = (9, 10, 11, 12, 13, 14)$. From probability theory, we have

$$P(\Pi) = P(S_1 \cup S_2) = P(S_1) + P(S_2) - |P(S_1 \cap S_2)|, \tag{31}$$

where $S_1 \cap S_2 = (14, 15, 16, 17, 18, 19)$ is NO.

In the first column of the table the node numbers are shown. The next two columns contain the xy-coordinate values of the nodes. The fourth column lists the values of $P(A_r)$ for $r = 1, 2, \cdots, N (N = 19)$, and the last column contains the angular measure in radians for each A_r.

TABLE 1

(r)	x	y	$P(A_r)$	$\Delta\theta_r$
1	−3	0	1.6803 8191 (−2)	2.3561 9449
2	0	−3	7.8697 697 (−3)	1.4288 9927
3	4	0	4.9999 7913 (−1)	2.4980 9154
4	1	0	0	0
5	0	0	−2.5000 0000 (−1)	−1.5707 9633
6	0	1	0	0
7	0	3	1.6803 8191 (−2)	2.3561 9449
8	−3	0	4.9985 6392 (−1)	2.3561 9449
9	−2	0	−3.1610 4292 (−1)	−4.6364 7609 (−1)
10	0	−1	1.6619 3815 (−1)	1.2490 4577
11	1	0	0	0
12	2	1	1.5721 5682 (−1)	2.3561 9449
13	0	1	6.1402 8574 (−2)	4.6364 7609 (−1)
14	−2	0	8.1445 3315 (−1)	3.1415 9265
15	0	1	−4.5518 3800 (−1)	−2.0344 4394
16	0	0	2.5000 0000 (−1)	1.5707 9633
17	1	0	−2.1101 0015 (−1)	−2.3561 9449
18	0	−1	−1.1199 0438 (−1)	−1.2490 4577
19	−2	0	1.6509 772 (−3)	4.6364 7609 (−1)

It is worth noting that, for the actual computations, nodes (4), (6), (11) could have been omitted. They were retained, because they are helpful here in specifying the primary circuits, namely

$$C_p(\Pi_1) = (4, 5, 6, 7, 8, 9, 10, 11),$$

$$C_p(\Pi_2) = (1, 2, 3, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20).$$

These PC's are both PO, hence $W = 2$ which agrees with the computation of W using (24b).

From (15), Table 1 and (17), we have

$$A(\Pi) = 16.5, \qquad W = \sum_{r=1}^{19} \frac{\Delta\theta_r}{2\pi} = 2, \qquad P(\Pi) = 2 - \sum_{r=1}^{19} P(A_r) = 8.5204\,0186\,(-1).$$

By separate computations, we also have as a check

$$P(S_1) = 7.0866\,8287\,(-1), \qquad P(S_2) = 4.3294\,2915\,(-1),$$

$$P(S_1 \cap S_2) = -2.8957\,1016\,(-1).$$

A Fortran IV listing of the CDC-6700 computer program is given in [3].

REFERENCES

[1] A. R. Di Donato, M. P. Jarnagin, Jr and R. K. Hageman, *Computation of the bivariate normal distribution over convex polygons*, Tech. Rep. NSWC/DL TR-3886, Naval Surface Weapons Center, Dahlgren, VA, September 1978.

[2] ———, *Computation of the integral of the bivariate normal distribution over convex polygons*, this Journal, 1 (1980), pp. 179–186.

[3] A. R. Di Donato and R. K. Hageman, *Computation of the integral of the bivariate normal distribution over arbitrary polygons*, Tech. Rep. NSWC/DL TR 80-166, Naval Surface Weapons Center, Dahlgren, VA, 22448, June 1980.

[4] K. Knopp, *Theory of Functions, Part One*, F. Bagemihl, trans., Dover, New York, 1945, p. 15.

[5] S. I. Warshaw, *Area of intersection of arbitrary polygons*, Rep. UCID-17430, Lawrence Livermore Laboratory, Univ. California, Livermore, March 1977.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

ON THE SOLUTION OF THE FINITE ELEMENT EQUATIONS FOR NONLINEAR SHELL ANALYSIS*

LOIS MANSFIELD†

Abstract. The relative efficiencies of the finite element methods derived from the potential energy formulation and from the mixed formulation for nonlinear shell analysis are compared. The result of this comparison is that the mixed method is considerably more efficient.

Key words. nonlinear shell analysis, arch problem, mixed finite element method, multigrid method

1. Introduction. The purpose of this paper is to compare the relative efficiencies of two finite element methods for the large displacement analysis of plates and shells. The first method comes from the minimum potential energy formulation where the stress-strain and strain-displacement relations are substituted into the equilibrium equations resulting in a system of equations to be solved for the displacements alone. This system represents the Euler–Lagrange equations associated with the minimization of the strain energy functional. The other method we consider, the so-called mixed method, results from using instead a modified form of the Hellinger–Reissner stationary variational principle, where the stresses and bending moments appear as independent variables in addition to the displacements. These methods are described in more detail in § 2.

The method of solution on which our comparison is based consists of taking the load to be given by $\mathbf{f} = \lambda \mathbf{f}_0$ and tracing successive solutions for a sequence of values of λ. At each load step we solve the finite element systems by Newton's method. At each Newton iteration step one is required to solve a linear system of equations. One can use either direct or iterative methods to do this. It seems to us that iterative methods might have some advantage here because one always has a very good initial guess in the previous iterate.

For the mixed method, however, the Jacobian matrix is indefinite, so there is the question of finding a good iterative method to use with the finite element method. Fortunately, since the finite element subspaces that are used to approximate the stresses and bending moments can be chosen so that their elements have support on a single triangle only, the unknowns associated with these quantities can be easily solved for in each iteration step in terms of the displacements from the previous step. The displacements can then be updated using a Richardson type iteration which always reduces and smooths the error. We have then chosen to accelerate this scheme by use of the multigrid method. Details are given in § 3.

The result of our comparison is that the mixed method is considerably more efficient than the method obtained from the minimum potential energy formulation. The greater efficiency in the mixed method comes because the Jacobian contains no worse than linear polynomial terms in the unknowns, while in the minimum potential energy method, quadratic polynomial terms appear. Also for the systems themselves, the mixed method has quadratic terms, while the minimum potential energy method has cubic terms. This means that far fewer arithmetic operations are required to set

* Received by the editors May 1, 1981, and in final form January 15, 1982. This research was supported by the National Science Foundation under grants MCS78-27482 and MCS80-16532.

† Department of Applied Mathematics and Computer Science, University of Virginia, Charlottesville, Virginia 22901.

up and evaluate these quantities, even though there are many more unknowns in the mixed method. Actual counts of arithmetic operations are given for both methods in § 4.

In the case of an arch or shell, as the load is increased a critical load is reached, which occurs at what is called a limit point in applied mechanics. At this point λ can no longer be used as a parameter in tracing the load curve. In § 5 we give experimental results for the arch problem where a new parameter s similar to [2] is introduced, and the load curve is traced beyond the limit point using this new parameter.

2. Description of the methods. The shell surface is defined parametrically by a position vector $\mathbf{r}(x, y)$ which is a vector function of two curvilinear surface coordinates x and y. Let Ω be the shell domain, a bounded open set in R^2 with boundary Γ. For simplicity we consider the shallow shell equations rather than the conventional shell equations. We assume the presence of large displacements but small strains, and that Kirchhoff's hypothesis holds. We assume that body forces $\mathbf{f}$ act on the shell and take as the boundary conditions

$$u = v = w = w_n = 0 \quad \text{on } \Gamma. \tag{1}$$

Let

$$\alpha_1^2 = \left(\frac{\partial r_1}{\partial x}\right)^2 + \left(\frac{\partial r_2}{\partial x}\right)^2 + \left(\frac{\partial r_3}{\partial x}\right)^2, \qquad \alpha_2^2 = \left(\frac{\partial r_1}{\partial y}\right)^2 + \left(\frac{\partial r_2}{\partial y}\right)^2 + \left(\frac{\partial r_3}{\partial y}\right)^2.$$

Then the displacements $\mathbf{u} = (u, v, w)$ may be obtained by solving the variational equation $W'(\mathbf{u}) = 0$ where $W(\mathbf{u})$ is the strain energy functional given by

$$\begin{aligned} W(\mathbf{u}) = \frac{E\delta}{2(1-\nu^2)} \int_\Omega \Bigg[&\left(\frac{1}{\alpha_1} u_x + k_{11} w + \frac{1}{2}\left(\frac{1}{\alpha_1} w_x\right)^2\right)^2 \\ &+ 2\nu \left(\frac{1}{\alpha_1} u_x + k_{11} w + \frac{1}{2}\left(\frac{1}{\alpha_1} w_x\right)^2\right)\left(\frac{1}{\alpha_2} v_y + k_{22} w + \frac{1}{2}\left(\frac{1}{\alpha_2} w_y\right)^2\right) \\ &+ \left(\frac{1}{\alpha_2} v_y + k_{22} w + \frac{1}{2}\left(\frac{1}{\alpha_2} w_y\right)^2\right)^2 \\ &+ \frac{1}{2}(1-\nu)\left(\frac{1}{\alpha_1} v_x + \frac{1}{\alpha_2} u_y + k_{12} w + \frac{1}{\alpha_1 \alpha_2} w_x w_y\right)^2 \Bigg] \alpha_1 \alpha_2 \, dx \, dy \\ + \frac{E\delta^3}{24(1-\nu^2)} \int_\Omega \Bigg[&\frac{1}{\alpha_1^4} w_{xx}^2 + \frac{1}{\alpha_2^4} w_{yy}^2 + \frac{2\nu}{\alpha_1^2 \alpha_2^2} w_{xx} w_{yy} + \frac{2(1-\nu)}{\alpha_1^2 \alpha_2^2} w_{xy}^2 \Bigg] \alpha_1 \alpha_2 \, dx \, dy \\ - \int_\Omega &\mathbf{f} \cdot \mathbf{u} \alpha_1 \alpha_2 \, dx \, dy. \end{aligned} \tag{2}$$

Here E is Young's modulus, ν is Poisson's ratio, δ denotes the thickness of the shell, and the $k_{\alpha\beta}$ denote the curvature of the shell.

The finite element method consists of choosing finite dimensional subspaces $S_u^h \subset H_0^1(\Omega)$, $S_v^h \subset H_0^1(\Omega)$, $S_w^h \subset H_0^2(\Omega)$, and solving

$$\begin{aligned} \frac{E\delta}{1-\nu^2} \int_\Omega \Bigg[&\left(\frac{1}{\alpha_1} u_x^h + k_{11} w^h + \frac{1}{2}\left(\frac{1}{\alpha_1^2} w_x^h\right)^2\right)\left(\frac{1}{\alpha_1} \hat{u}_x + k_{11} \hat{w}^h + \frac{1}{\alpha_1^2} w_x^h \hat{w}_x^h\right) \\ &+ \nu \left(\frac{1}{\alpha_1} u_x^h + k_{11} w^h + \frac{1}{2}\left(\frac{1}{\alpha_1^2} w_x^h\right)^2\right)\left(\frac{1}{\alpha_2} \hat{v}_y^h + k_{22} \hat{w}^h + \frac{1}{\alpha_2^2} w_y^h \hat{w}_y^h\right) \end{aligned}$$

$$+\nu\left(\frac{1}{\alpha_2}v_y^h+k_{22}w^h+\frac{1}{2}\left(\frac{1}{\alpha_2^2}w_y^h\right)^2\right)\left(\frac{1}{\alpha_1}\hat{u}_x^h+k_{11}\hat{w}^h+\frac{1}{\alpha_1^2}w_x^h\hat{w}_x^h\right)$$

$$+\left(\frac{1}{\alpha_2}v_y^h+k_{22}w^h+\frac{1}{2}\left(\frac{1}{\alpha_2^2}w_y^h\right)^2\right)\left(\frac{1}{\alpha_2}\hat{v}_y^h+k_{22}\hat{w}^2+\frac{1}{\alpha_2^2}w_y^h\hat{w}_y^h\right)$$

$$+\frac{1}{2}(1-\nu)\left(\frac{1}{\alpha_1}v_x^h+\frac{1}{\alpha_2}u_y^h+k_{12}w^h+\frac{1}{\alpha_1\alpha_2}w_x^hw_y^h\right)$$

$$\times\left(\frac{1}{\alpha_1}\hat{v}_x^h+\frac{1}{\alpha_2}\hat{u}_y^h+k_{12}\hat{w}^h+\frac{1}{\alpha_1\alpha_2}\left\{w_x^h\hat{w}_y^h+w_y^h\hat{w}_x^h\right\}\right)\Bigg]\alpha_1\alpha_2\,dx\,dy$$

$$+\frac{E\delta^3}{12(1-\nu^2)}\int_\Omega\left[\frac{1}{\alpha_1^4}w_{xx}^h\hat{w}_{xx}^h+\frac{1}{\alpha_2^4}w_{yy}^h\hat{w}_{yy}^h+\frac{\nu}{\alpha_1^2\alpha_2^2}\{w_{xx}^h\hat{w}_{yy}^h+w_{yy}\hat{w}_{xx}^h\}\right.$$

$$\left.+\frac{2(1-\nu)}{\alpha_1^2\alpha_2^2}w_{xy}^h\hat{w}_{xy}^h\right]\alpha_1\alpha_2\,dx\,dy$$

$$-\int_\Omega \mathbf{f}\cdot\hat{\mathbf{u}}^h\alpha_1\alpha_2\,dx\,dy=0,\quad \text{all } \hat{u}^h\in S_u^h,\quad \hat{v}^h\in S_v^h,\quad \hat{w}^h\in S_w^h,$$

for $u^h\in S_u^h$, $v^h\in S_v^h$, $w^h\in S_w^h$.

We assume the region Ω has been triangulated and that the spaces S_u^h, S_v^h, and S_w^h consist of piecewise polynomials over this triangulation. In this paper we will not deal with the complication of curved boundaries so we shall assume that Ω is a convex polygon. Our comparisons with the mixed method will be based on the use of C^0-quadratic finite elements to approximate u and v and the 12-parameter Clough–Tocher finite elements to approximate w.

Let $\boldsymbol{\phi}=(\mathbf{u},\mathbf{N},\mathbf{M})$ where $N_{\alpha\beta}$, $M_{\alpha\beta}$, $1\leqq\alpha$, $\beta\leqq 2$ are the stress resultants and bending moments. The mixed method results from solving $S'(\boldsymbol{\phi})=0$, where $S(\boldsymbol{\phi})$ given by

$$S(\boldsymbol{\phi})=\frac{(1-\nu^2)}{2E\delta}\int_\Omega[-N_{11}^2+2\nu N_{11}N_{22}-2(1+\nu)N_{12}^2-N_{22}^2]\alpha_1\alpha_2\,dx\,dy$$

$$+\int_\Omega\left[\left(\frac{1}{\alpha_1}u_x+k_{11}w+\frac{1}{2}\left(\frac{1}{\alpha_1}w_x\right)^2\right)N_{11}+\left(\frac{1}{\alpha_2}v_y+k_{22}w+\frac{1}{2}\left(\frac{1}{\alpha_2}w_y\right)^2\right)N_{22}\right.$$

$$\left.+\left(\frac{1}{\alpha_2}u_y+\frac{1}{\alpha_1}v_x+k_{12}w+\frac{1}{\alpha_1\alpha_2}w_xw_y\right)N_{12}\right]\alpha_1\alpha_2\,dx\,dy$$

$$+\frac{6(1-\nu^2)}{E\delta^3}\int_\Omega[-M_{11}^2+2\nu M_{11}M_{22}-2(1+\nu)M_{12}^2-M_{22}^2]\alpha_1\alpha_2\,dx\,dy \tag{4}$$

$$-\int_\Omega\left[\frac{1}{\alpha_1^2}w_{xx}M_{11}+\frac{2}{\alpha_1\alpha_2}w_{xy}M_{12}+\frac{1}{\alpha_2^2}w_{yy}M_{22}\right]\alpha_1\alpha_2\,dx\,dy$$

$$-\int_\Omega \mathbf{f}\cdot\mathbf{u}\alpha_1\alpha_2\,dx\,dy$$

is a modified form of the Hellinger–Reissner stationary variational principle.

In addition to the subspaces S_u^h, S_v^h, and S_w^h, one must choose subspaces $S_{\mathbf{N}}^h=(S_N^h)^3$ with $S_N^h\subset L^2(\Omega)$ and $S_{\mathbf{M}}^h=(S_M^h)^3$ with $S_M^h\subset L^2(\Omega)$. The finite element method consists of solving

$$
\begin{aligned}
(5)\quad & \frac{-(1-\nu^2)}{E\delta}\int_\Omega [N_{11}^h\hat{N}_{11}^h - \nu N_{11}^h\hat{N}_{22}^h - \nu N_{22}^h\hat{N}_{11}^h + 2(1+\nu)N_{12}^h\hat{N}_{12}^h + N_{22}^h\hat{N}_{22}^h]\alpha_1\alpha_2\,dx\,dy \\
& + \int_\Omega \Bigg[\left(\frac{1}{\alpha_1}u_x^h + k_{11}w^h + \frac{1}{2}\left(\frac{1}{\alpha_1}w_x^h\right)^2\right)\hat{N}_{11}^h + N_{11}^h\left(\frac{1}{\alpha_1}\hat{u}_x^h + k_{11}\hat{w}^h + \frac{1}{\alpha_1^2}w_x^h\hat{w}_x^h\right) \\
& + \left(\frac{1}{\alpha_2}v_y^h + k_{22}w^h + \frac{1}{2}\left(\frac{1}{\alpha_2}w_y^h\right)^2\right)\hat{N}_{22}^h + N_{22}^h\left(\frac{1}{\alpha_2}\hat{v}_y^h + k_{22}\hat{w}^h + \frac{1}{\alpha_2^2}w_y^h\hat{w}_y^h\right) \\
& + \left(\frac{1}{\alpha_2}u_y^h + \frac{1}{\alpha_1}v_x^h + k_{12}w^h + \frac{1}{\alpha_1\alpha_2}w_x^h w_y^h\right)\hat{N}_{12}^h \\
& + N_{12}\left(\frac{1}{\alpha_2}\hat{u}_y^h + \frac{1}{\alpha_1}\hat{v}_x^h + k_{12}\hat{w}^h + \frac{1}{\alpha_1\alpha_2}\{w_x^h\hat{w}_y^h + w_y^h\hat{w}_x^h\}\right)\Bigg]\alpha_1\alpha_2\,dx\,dy \\
& - \frac{12(1-\nu^2)}{E\delta^3}\int_\Omega [M_{11}^h\hat{M}_{11}^h - \nu(M_{11}^h\hat{M}_{22}^h + M_{22}^h\hat{M}_{11}^h) \\
& \qquad + 2(1+\nu)M_{12}^h\hat{M}_{12}^h + M_{22}^h\hat{M}_{22}^h]\alpha_1\alpha_2\,dx\,dy \\
& - \int_\Omega \Bigg[\frac{1}{\alpha_1^2}\{w_{xx}^h\hat{M}_{11}^h + M_{11}\hat{w}_{xx}^h\} + \frac{2}{\alpha_1\alpha_2}\{w_{xy}^h\hat{M}_{12}^h + M_{12}^h\hat{w}_{xy}^h\} \\
& \qquad + \frac{1}{\alpha_2^2}\{w_{yy}^h\hat{M}_{22}^h + M_{22}^h\hat{w}_{yy}^h\}\Bigg]\alpha_1\alpha_2\,dx\,dy \\
& - \int_\Omega \mathbf{f}\cdot\mathbf{u}^h\alpha_1\alpha_2\,dx\,dy = 0, \qquad \text{all } \hat{u}^h \in S_u^h,\ \hat{v}^h \in S_v^h,\ w^h \in S_w^h,\ \hat{\mathbf{N}}^h \in S_{\mathbf{N}}^h,\ \hat{\mathbf{M}}^h \in S_{\mathbf{M}}^h,
\end{aligned}
$$

for $u^h \in S_u^h$, $v^h \in S_v^h$, $w^h \in S_w^h$, $\mathbf{N}^h \in S_{\mathbf{N}}^h$, $\mathbf{M}^h \in S_{\mathbf{M}}^h$.

In [4] it was shown that optimal orders of convergence are obtained if for all $\hat{u}^h \in S_u^h$, $\hat{v}^h \in S_v^h$, and $\hat{w}^h \in S_w^h$, $\hat{u}_x^h$, $\hat{v}_y^h$, and $\hat{v}_x^h + \hat{u}_y^h$ are contained in S_N^h and $\hat{w}_{xx}^h$, $\hat{w}_{xy}^h$, and $\hat{w}_{yy}^h$ are contained in S_M^h. Although these conditions have not been shown to be absolutely necessary, it is known (see [3]) that other choices, such as choosing all subspaces to consist of the same type of finite elements, do not give optimal rates of convergence. We choose both S_N^h and S_M^h to consist of piecewise linear polynomials with jumps at the edges and vertices of each triangle in the triangulation of Ω. For S_M^h there should be jumps at the edges and vertices of each subtriangle of the macrotriangles associated with the Clough–Tocher elements as well. Thus basis elements can be chosen for S_N^h and S_M^h with support on one triangle only. This fact will enable us to efficiently use the multrigrid algorithm to solve the linear system in each step of Newton's method.

We also compare the minimum potential energy method with the mixed method for the arch problem. For the arch problem the strain energy functional is given by

$$
\begin{aligned}
(6)\quad W(\mathbf{u}) &= \frac{E\delta}{2(1-\nu^2)}\int_0^a \left(\frac{1}{\alpha}u' + kw + \frac{1}{2}\left(ku - \frac{1}{\alpha}w'\right)^2\right)^2 \alpha\,dx \\
&+ \frac{E\delta^3}{24(1-\nu^2)}\int_0^a \left(\frac{k}{\alpha}u' - \frac{1}{\alpha^2}w''\right)^2 \alpha\,dx - \int_0^a \mathbf{f}\cdot\mathbf{u}\alpha\,dx,
\end{aligned}
$$

where $\mathbf{u} = (u, w)$, k denotes the curvature and $\alpha^2 = (r_1')^2 + (r_2')^2$. The finite element method consists of choosing finite dimensional subspaces $S_u^h \subset H_0^1(0, a)$ and $S_w^h \subset H_0^2(0, a)$, and solving

$$\begin{aligned}(7)\quad &\frac{E\delta}{1-\nu^2}\int_0^a \left(\frac{1}{\alpha}u_h' + kw_h + \frac{1}{2}\left(ku_h - \frac{1}{\alpha}w_h'\right)^2\right)\\ &\quad\times\left(\frac{1}{\alpha}\hat{u}_h' + k\hat{w}_h + \left(ku_h - \frac{1}{\alpha}w_h'\right)\left(k\hat{u}^h - \frac{1}{\alpha}\hat{w}_h'\right)\right)\alpha\,dx\\ &+\frac{E\delta^3}{12(1-\nu^2)}\int_0^a \left(\frac{k}{\alpha}u_h' - \frac{1}{\alpha^2}w_h''\right)\left(\frac{k}{\alpha}\hat{u}_h' - \frac{1}{\alpha^2}\hat{w}_h''\right)\alpha\,dx - \int_0^a \mathbf{f}\cdot\hat{\mathbf{u}}_h\alpha\,dx = 0,\\ &\qquad\qquad\text{all } \hat{u}_h \in S_u^h, \quad \hat{w}_h \in S_w^h,\end{aligned}$$

for $u_h \in S_u^h$, $w_h \in S_w^h$.

The mixed method results from solving $S'(\boldsymbol{\phi}) = 0$ where $S(\boldsymbol{\phi})$ is given by

$$\begin{aligned}(8)\quad S(\boldsymbol{\phi}) = &\frac{-(1-\nu^2)}{2E\delta}\int_0^a N^2\alpha\,dx - \frac{6(1-\nu^2)}{E\delta^3}\int_0^a M^2\alpha\,dx\\ &+\int_0^a \left[\left(\frac{1}{\alpha}u' + kw + \frac{1}{2}\left(ku - \frac{1}{\alpha}w'\right)^2\right)N + \left(\frac{k}{\alpha}u' - \frac{1}{\alpha^2}w''\right)M\right]\alpha\,dx\\ &-\int_0^r \mathbf{f}\cdot\mathbf{u}\alpha\,dx,\end{aligned}$$

where $\boldsymbol{\phi} = (u, w, N, M)$. The finite element method consists of choosing in addition to S_u and S_w, finite dimensional subspaces $S_N^h \subset L^2(0, a)$, $S_M^h \subset L^2(0, a)$, and solving

$$\begin{aligned}(9)\quad &\frac{-(1-\nu^2)}{E\delta}\int_0^a N_h\hat{N}_h\alpha\,dx - \frac{12(1-\nu^2)}{E\delta^3}\int_0^a M_h\hat{M}_h\alpha\,dx\\ &+\int_0^a \left[\left(\frac{1}{\alpha}u_h' + kw_h + \frac{1}{2}\left(ku_h - \frac{1}{\alpha}w_h'\right)^2\right)\hat{N}_h\right.\\ &\qquad + N_h\left(\frac{1}{\alpha}\hat{u}_h' + k\hat{w}_h + \left(ku_h - \frac{1}{\alpha}w_h'\right)\left(k\hat{u}_h - \frac{1}{\alpha}\hat{w}_h'\right)\right)\\ &\qquad \left.-\left(\frac{k}{\alpha}u_h' - \frac{1}{\alpha^2}w_h''\right)\hat{M}_h - M_h\left(\frac{k}{\alpha}\hat{u}_h' - \frac{1}{\alpha^2}\hat{w}_h''\right)\right]\alpha\,dx\\ &-\int_0^a \mathbf{f}\cdot\mathbf{u}_h\alpha\,dx = 0,\\ &\qquad\qquad\text{all } \hat{u}_h \in S_u^h, \quad \hat{w}_h \in S_w^h, \quad \hat{N}_h \in S_N^h, \quad \hat{M}_h \in S_M^h,\end{aligned}$$

for $u_h \in S_u^h$, $w_h \in S_w^h$, $N_h \in S_N^h$, $M_h \in S_M^h$.

These equations differ from being one-dimensional versions of the shell equations in that the assumption that derivatives of curvature can be neglected, which is used in the shallow shell theory, is not assumed; therefore, (6–9) give equations of the deep arch.

In our comparison we shall choose S_u^h to consist of C^0-quadratic or linear finite elements, S_w^h to consist of piecewise cubic Hermite polynomials, S_N^h to consist of piecewise linear polynomials with jumps at the grid points or piecewise constants, and S_M^h to consist of piecewise linear polynomials with jumps at the grid points.

3. Method of solution of the equations. Our comparison of the relative efficiencies of the two finite element methods described in the previous section will be based on the following method for the solution of the equations. We take the load to be given by $f = \lambda f_0$ and trace successive solutions for a sequence of values of λ. In the case of an arch or shell, there is a critical load which occurs at what is called a limit point in applied mechanics. At this point λ can no longer be used as a parameter in tracing the load curve. A method for continuing beyond this critical load is given in § 5. Here, for simplicity we shall assume we are in a region where λ is a suitable parameter.

At each load step we solve the nonlinear system of equations $F(x) = 0$, by Newton's method or by a modified Newton's method using the solution at the previous load step as the initial approximation. For $\lambda = 0$ the only solution to the equations in the previous section is $\mathbf{u} = 0$, $\mathbf{N} = 0$, $\mathbf{M} = 0$. At each Newton iteration step we have chosen to solve the linear system of equations

$$J(x_n)(x_{n+1} - x_n) = -F(x_n)$$

by the multigrid method.

We have not made a detailed study of how the multigrid method compares with other iterative methods or with direct methods in this context. This is a subject for further research. Because one always has a good initial guess, iterative methods may have some advantage here. If so, it is important to find a rapidly convergent method to use with the mixed method where the Jacobian is indefinite. We chose to use the multigrid method instead of other possible iterative methods, because of both its fast convergence and the fact that, as we show in this section, it can be modified to give the same fast convergence for the mixed methods where the linear systems are indefinite, as for positive definite systems.

To solve the linear system

$$Kx = y \tag{10}$$

by the multigrid method, it is assumed that one has a sequence of grids parameterized by h_q with associated subspaces $S_q = S_{h_q}$, where $h_q = \rho h_{q+1}$, $\rho > 1$. To solve the linear system

$$K_q x^q = y^q, \tag{11}$$

associated with the grid parameterized by h_q, the multigrid method alternates iteration sweeps, whose purpose is to smooth the error, with corrections obtained from approximating the solution of the residual equation on the coarser grid parameterized by h_{q-1}. This can be formalized by the following algorithm given in [5].

Starting with a given initial approximation $x^{q,0,0}$ to the solution x^q to (11):

Do steps 1, 2, and 3 for $k = 0, 1, \cdots, \mu - 1$:

1. $x^{q,k,i} = x^{q,k,i-1} - \beta(K_q x^{q,k,i-1} - y^q)$, $i = 1, 2, \cdots, n$.

2. With $\varepsilon^{q-1,k,0}$ defined by

$$K_{q-1}\varepsilon^{q-1,k,0} = E_{q-1}^T r^{q,k,n}, \tag{12}$$

where $r^{q,k,n} = y^q - K_q x^{q,k,n}$, compute $\eta^{q-1,k,0}$ such that

$$\|\eta^{q-1,k,0} - \varepsilon^{q-1,k,0}\|_{l^2} \leqq \delta \|\varepsilon^{q-1,k,0}\|_{l^2}.$$

3. Set $x^{q,k+1,0} = x^{q,k,n} + E_{q-1}\eta^{q-1,k,0}$.

To approximate the solution to (12), one uses the same algorithm. It is assumed that for the coarsest grid, the system is small enough to solve efficiently by a direct method or to carry out the iterations in step 1 until convergence. In most situations the Richardson iteration of step 1 may be replaced by the Gauss–Seidel method or an under-relaxed Jacobi method. Let $\mathscr{E}_{q-1}$ denote the embedding of S_{q-1} into S_q. Corresponding to the operator $\mathscr{E}_{q-1}$ there is an operator from $R(N_{q-1})$ to $R(N_q)$. Relative to bases $\{\psi_i^{q-1}\}_{i=1}^{N_{q-1}}$ and $\{\psi_i^q\}_{i=1}^{N_q}$ of the subspaces S_{q-1} and S_q, this operator has a matrix representation E_{q-1}. Its transpose E_{q-1}^T maps $R(N_q)$ onto $R(N_{q-1})$.

Under the assumption that the system (10) came from the minimization of a quadratic functional and that $S_{q-1} \subset S_q$, it was shown in [5] that the multigrid method can solve (10) with $O(N)$ arithmetical operations, where N is the number of unknowns in (10). When the Clough–Tocher elements are used, to satisfy $S_{q-1} \subset S_q$, it is necessary that S_q be obtained by refining each of the subtriangles in each macrotriangle. Although the assumption that $S_{q-1} \subset S_q$ enabled Nicolaides to prove his result, it is undoubtedly not completely necessary. One can obtain his results if one can show that $E_{q-1}x^{q-1}$ is a good approximation to x^q, where x^q is the solution to (11) and x^{q-1} is the solution to the same finite element problem but on the grid parameterized by h_{q-1}. In the case $S_{q-1} \subset S_q$, the embedding $\mathscr{E}_{q-1}$ may be taken to be the identity, and $\|x^q - E_{q-1}x^{q-1}\|$ is easy to estimate using standard finite element convergence results.

If the multigrid algorithm is applied as described above, it will not work to solve the equations resulting from the use of mixed methods. Since the mixed methods result from stationary principles rather than minimum principles, the linear systems are indefinite so that step 1 in the multigrid algorithm will magnify rather than reduce the part of the error associated with the negative eigenvalues. Although the purpose of step 1 is to smooth the error rather than necessarily to reduce it, the fact that the positive and negative eigenvalues are of nearly equal magnitude means that step 1 will magnify the error faster than steps 2 and 3 can reduce it. For the mixed methods, the linear systems have the form

$$\begin{pmatrix} -M & B \\ B^T & C \end{pmatrix} \begin{pmatrix} \boldsymbol{\sigma} \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ \mathbf{g} \end{pmatrix},$$

where the vector $\boldsymbol{\sigma}$ represents those unknowns associated with $\mathbf{M}^h$ and $\mathbf{N}^h$, and $\mathbf{u}$ represents the unknowns associated with $\mathbf{u}^h$. Since elements of our subspaces S_N^h and S_M^h have support only on one triangle, the matrix M is block diagonal with the size of the blocks determined by the number of basis functions with common support on a given triangle.

For mixed methods, we propose that step 1 of the multigrid algorithm be modified to

(13) $\quad$ Solve $M\boldsymbol{\sigma}^{q,k,i} = B\mathbf{u}^{q,k,i-1} - \mathbf{f}^q$ for $\boldsymbol{\sigma}^{q,k,i}$,

(14) $\quad$ Set $\mathbf{u}^{q,k,i} = \mathbf{u}^{q,k,i-1} - \beta(B^T\boldsymbol{\sigma}^{q,k,i} + C\mathbf{u}^{q,k,i-1} - \mathbf{g}^q), \qquad i = 1, 2, \cdots, n.$

Because M is block diagonal, (13) is inexpensive to solve. If $B^TM^{-1}B + C$ is positive definite, β can be chosen to both reduce and smooth the error.

To use (13)–(14), one must determine a suitable value for the parameter β in (14). Note that the parameter β depends on λ. In numerical experiments we have done, we took β to be a vector rather than a scalar, as indicated in (14). We determined the main diagonal of the matrix $B^TM^{-1}B$, and initially took β to be the vector which made (14) equivalent to an under-relaxed Jacobi iteration, to solve

$$B^TM^{-1}Bu = g.$$

Since M is block diagonal the main diagonal of $B^TM^{-1}B$ is not difficult to determine. We found that we were able to maintain satisfactory convergence with our initial β until we got fairly close to λ_{cr}. By this time the matrix $B^TM^{-1}B+C$ had changed enough that we needed to update β. The matrices M and B can be decomposed into submatrices coming from the stretching and bending equations in the system $F(x)=0$. We determined the main diagonal of the most significant part of $B^TM^{-1}B$ and used this to update β. The components of the matrix C were small compared with those of $B^TM^{-1}B$. Throughout, we followed the policy of updating β only when our bound for the maximal allowable number of iterations in the linear system was exceeded.

The multigrid method has the disadvantage that the region must be simple enough for it to be feasible to generate a sequence of grids as described above. Another iterative method which can be used for mixed methods is

(15) Solve $M\boldsymbol{\sigma}^i = B\mathbf{u}^{i-1}-\mathbf{f}$ for $\boldsymbol{\sigma}^i$,

(16) Set $\mathbf{u}^i=\gamma\mathbf{u}^{i-1}+(1-\gamma)\mathbf{u}^{i-2}-\beta(B^T\boldsymbol{\sigma}^i+C\mathbf{u}^{i-1}-\mathbf{g})$.

Equations (15)–(16) represent a second degree iterative process applied to $B^TM^{-1}B+C$ and is suggested in [1].

4. Comparison of efficiency of the two methods. In this section we will compare the efficiency of the minimum potential energy methods with mixed methods with regard to the efficiency of solving the resulting algebraic systems. Our comparison will be based principally on a comparison of the number of arithmetic operations required to evaluate the Jacobian matrices and the systems themselves in order to perform each Newton iteration step,

$$J(x_n)(x_{n+1}-x_n)=-F(x_n).$$

The procedure we follow for evaluating the Jacobian J and the system F is the standard procedure of proceeding element by element and inserting into J and F the contributions from that element. This procedure relies on a table of integrals for each element, which needs to be set up once and stored. In the case of the plate which has constant coefficients, a single table of integrals can be used for all the elements provided that multiplications by factors dependent upon the geometry of the element caused by the transformation of terms like

$$\int_T \frac{\partial^2 w}{\partial x^2}\frac{\partial^2 \hat{w}}{\partial x^2}\,d\mathbf{x}$$

to a standard triangle are done separately. For simplicity we shall assume that these operations are incorporated into the table of integrals for each element, so that even in the case of a plate, a separate table exists for each element.

We have not compared the number of operations required to set up these tables, except for the arch with S_u^h consisting of piecewise linear polynomials in C^0, S_w^h consisting of piecewise cubic hermite polynomials, S_N^h consisting of piecewise constants, and S_M^h consisting of piecewise linear polynomials with jumps at the grid points. Here 70 multiplications were required for the mixed method, while 1102 multiplications were required for the potential energy method. The integrals were computed using a Gauss quadrature rule of sufficient accuracy to carry out all integrations exactly, assuming the k_{ij} and f are constants. This insures that the integrals are computed with the same order of accuracy as the finite element methods themselves (see [7]). Although this comparison may be slightly biased in favor of the mixed method because S_N^h consists of piecewise constants, it certainly seems to indicate that mixed methods are no less efficient than potential energy methods with regard to the calculation of the element integral tables.

The numbers of operations required per element of the finite element mesh are given in Tables 1 and 2 below for the plate and shell problems, for the potential energy method (POT) and mixed method (MXD). The finite element subspaces are as stated in § 2. We have split up the number of operations required to set up $J(x_n)$ into a linear part and a nonlinear part, since only the nonlinear part needs to be updated at each iteration step.

TABLE 1
Number of arithmetic operations per element for plate problem.

	set up $J(x_n)$			set up $F(x_n)$
	linear	nonlinear	total	
POT	780	216,666	217,446	1,370,212
MXD	385	6660	7045	11,868

TABLE 2
Number of arithmetic operations per element for shell problem.

	set up $J(x_n)$			set up $F(x_n)$
	linear	nonlinear	total	
POT	2478	248,658	251,136	1,429,888
MXD	493	6660	7153	12,084

Table 3 gives the numbers of arithmetic operations per element required for the arch problem. POT1 refers to the potential energy method as described in § 2, where S_u^h consists of C^0-piecewise linear polynomials, and POT2 refers to the same method, where S_u^h consists of C^0-piecewise quadratic polynomials. MXD1 and MXD2 are the corresponding mixed methods.

TABLE 3
Number of arithmetic operations per interval for arch problem.

	set up $J(x_n)$			set up $F(x_n)$
	linear	nonlinear	total	
POT1	63	2523	2586	4176
POT2	84	4374	4458	6902
MXD1	4	93	97	322
MXD2	6	308	314	644

Table 1 indicates that for the plate problem 30.9 times more operations per element are required to set up $J(x_n)$ in POT as in MXD and that 31.2 times more operations per element are required to set up the system itself. For the shell problem 35.1 times more operations per element are required in POT to set up the Jacobian and 35.6 times more operations are required to set up the system itself. For the arch problem these numbers are 26.6 and 12.97 for POT1 and 14.2 and 10.7 for POT2.

We give here more details concerning how we calculated the numbers given in Tables 1–3. Consider the system for the potential energy method for the plate problem. For any element K, one has to evaluate

$$\frac{E\delta}{1-\nu^2}\int_K [u_x^h\hat{u}_x^h+\tfrac{1}{2}(w_x^h)^2\hat{u}_x^h+\nu v_y^h\hat{u}_x^h+\frac{\nu}{2}(w_y^h)^2\hat{u}_x^h \tag{17}$$
$$-\tfrac{1}{2}(u_y^h+v_x^h+w_x^hw_y^h)\hat{u}_y^h]\,dx\,dy-\lambda\int_K f_1\hat{u}^h\,dx\,dy,$$

$$\frac{E\delta}{1-\nu^2}\int_K [v_y^h\hat{v}_x^h+\tfrac{1}{2}(w_y^h)^2\hat{v}_y^h+\nu u_x^h\hat{v}_y^h+\frac{\nu}{2}(w_x^h)^2v_y^h \tag{18}$$
$$-\tfrac{1}{2}(v_x^h+u_y^h+w_x^hw_y^h)\hat{v}_x^h]\,dx\,dy-\lambda\int_K f_2\hat{v}^h\,dx\,dy,$$

and

$$\frac{E\delta}{1-\nu^2}\int_K [(u_x^h+\tfrac{1}{2}(w_x^h)^2)w_x^h\hat{w}_x^h+(v_y^h+\tfrac{1}{2}(w_y^h)^2)w_y^h\hat{w}_y^h+\nu u_x^hw_y^h\hat{w}_y^h+\nu v_y^hw_x^h\hat{w}_x^h$$
$$+\tfrac{1}{2}(1-\nu)(u_y^h+v_x^h)(w_x^h\hat{w}_y^h+w_y^h\hat{w}_x^h)+\tfrac{1}{2}(w_x^h)^2w_y^h\hat{w}_y^h+\tfrac{1}{2}(w_y^h)^2w_x^h\hat{w}_x^h]\,dx\,dy \tag{19}$$
$$+\frac{E\delta^3}{12(1-\nu^2)}\int_K [w_{xx}^h\hat{w}_{xx}^h+w_{yy}^h\hat{w}_{yy}^h+\nu w_{xx}^h\hat{w}_{yy}^h+\nu w_{yy}^hw_{xx}^h$$
$$+2(1-\nu)w_{xy}^h\hat{w}_{xy}^h]\,dx\,dy-\lambda\int_K f_3\hat{w}^h\,dx\,dy,$$

where $\hat{u}^h$ and $\hat{v}^h$ range over the six piecewise quadratic polynomials which are nonzero on K, and $\hat{w}^h$ ranges over 12 nonzero Clough–Tocher cubics on K.

Terms like $\int_K u_x^h\hat{u}_x^h\,dx\,dy$ require 6 multiplications and 6 additions to evaluate, while terms like $\int_K (w_x^h)^2\hat{u}_x^h$ require 288 multiplications and 144 additions. Thus (17) and (18) each contribute 6 entries requiring 1349 arithmetic operations. Expression (19) contributes 12 entries requiring 29,502 operations each. Thus 370,212 operations are required to evaluate the system for the potential energy method for the plate problem.

On each element K for the mixed method for the plate problem, one has to evaluate

$$\int_K\left[u_x^h+\frac{1}{2}(w_x^h)^2-\frac{(1-\nu^2)}{E\delta}(N_{11}^h-\nu N_{22}^h)\right]\hat{N}_{11}^h\,dx\,dy, \tag{20}$$

$$\int_K\left[u_y^h+v_x^h+w_x^hw_y^h-2\frac{(1-\nu^2)}{E\delta}(1+\nu)N_{12}^h\right]\hat{N}_{12}^h\,dx\,dy, \tag{21}$$

$$\int_K\left[v_y^h+\frac{1}{2}(w_y^h)^2-\frac{(1-\nu^2)}{E\delta}(N_{22}^h-\nu N_{11}^h)\right]\hat{N}_{22}^h\,dx\,dy, \tag{22}$$

$$\int_K\left[-w_{xx}^h-\frac{12(1-\nu^2)}{E\delta^3}(M_{11}^h-\nu M_{22}^h)\right]\hat{M}_{11}^h\,dx\,dy, \tag{23}$$

$$\int_K\left[-w_{xy}^h-\frac{12(1-\nu^2)}{E\delta^3}(1+\nu)M_{12}^h\right]\hat{M}_{12}^h\,dx\,dy, \tag{24}$$

$$\int_K\left[-w_{yy}^h-\frac{12(1-\nu^2)}{E\delta^3}(M_{22}^h-\nu M_{11}^h\right]\hat{M}_{22}^h\,dx\,dy, \tag{25}$$

$$(26)\quad \int_K (N_{11}^h \hat{u}_x^h + N_{12}^h \hat{u}_y^h)\,dx\,dy - \lambda \int_K f_1 \hat{u}^h\,dx\,dy,$$

$$(27)\quad \int_K (N_{22}^h \hat{v}_y^h + N_{12}^h \hat{v}_x^h)\,dx\,dy - \lambda \int_K f_2 \hat{v}^h\,dx\,dy,$$

$$(28)\quad \begin{aligned}\int_K & [N_{11}^h w_x^h \hat{w}_x^h + N_{12}^h (w_x^h \hat{w}_y^h + w_y^h \hat{w}_x^h) + N_{22}^h w_y^h \hat{w}_y^h \\ & -M_{11}^h \hat{w}_{xx}^h - 2M_{12}^h \hat{w}_{xy}^h - M_{22}^h \hat{w}_{yy}^h]\,dx\,dy - \lambda \int_K f_3 \hat{w}^h\,dx\,dy.\end{aligned}$$

Expressions (20) and (22) each contribute 3 entries with 459 operations each, while (21) contributes 3 entries with 463 operations each. (23) and (25) each contribute 9 entries with 62 operations each, while (24) contributes 9 entries with 45 operations each. Finally (26) and (27) each contribute 6 entries with 9 operations each, while (28) contributes 12 entries with 505 operations each. Thus 11,868 operations are required to evaluate the system for the mixed method for the plate problem.

The numbers of operations required to evaluate the systems for the shell and arch problems are determined in the same manner. To evaluate the Jacobians, one notes that since they are symmetric, only the upper triangular part needs to be set up. The determination of the number of operations required is done similarly to what we have indicated for the evaluations of the systems.

5. Continuation beyond the critical load. For an arch or shell, as the load is increased, a critical load is reached which occurs at what is called a limit point in applied mechanics (see Fig. 1). At a limit point the Jacobian becomes singular, but perhaps more importantly, the load parameter λ is double valued in the neighborhood of the critical load λ_{cr}. This means that λ can no longer be taken to be a parameter in tracing the load curve beyond λ_{cr}.

To continue beyond λ_{cr}, we regard λ as a dependent variable to be determined. We introduce a new parameter s and trace the solution curve using s. Let

$$(29)\qquad G(x) - \lambda b = 0$$

denote the system of equations to be solved from any of the methods described in

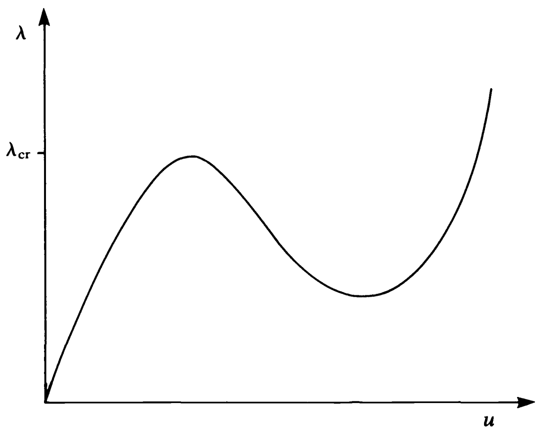


FIG. 1

§ 2, where x denotes all of the unknowns of the problem, and b is the vector obtained by taking the inner product of $\mathbf{f}_0$ with each of the basis functions used to define $\mathbf{u}^h$.

We augment the system (17) by adding the single equation

$$c\int_\Omega f_0\cdot\mathbf{u}^h+\lambda-s=0,$$

or

$$N(x,\lambda,s)=c\mathbf{u}^T\tilde{M}\bar{f}_0+\lambda-s=0, \tag{30}$$

where $\tilde{M}$ is the mass matrix associated with $\mathbf{u}^h$, $\bar{f}_0$ is the set of nodal values of f_0, and c is a positive constant. We solve the combined system (29)–(30), where s is a parameter but λ is to be determined along with x. Other choices for $N(x,\lambda,s)$ are given in [2].

The linear system to be solved at each Newton iteration step is

$$\begin{aligned}&J(x_n,\lambda_n)(x_{n+1}-x_n)-\lambda_{n+1}b=-G(x),\\&c(u^{n+1})^T\tilde{M}\bar{f}_0+\lambda_{n+1}=s.\end{aligned} \tag{31}$$

We have done some numerical experimentation with a shallow circular arch subject to a uniform pressure. We used piecewise linear polynomials to approximate u, piecewise cubic Hermite polynomials to approximate w, piecewise constants to approximate N and piecewise linear polynomials with jumps at the grid points to approximate M. The mixed method proved to be so much more efficient than the potential energy method (in actual computations as well as from the analysis of § 4) that most of the experimentation was done with the mixed method.

In fact, in actual computations we noticed another advantage that mixed methods seem to have over potential energy methods. With the mixed method we were able to take much larger load steps and still get convergence of Newton's method. This shows up in the theoretical analysis of the two methods. In [4] Kantorovich's theorem was used to show the existence and uniqueness of solutions along the load curve. Away from limit points, the size of the ball about the initial guess in which one could guarantee a solution shrank as w increased for the potential energy method, but remained constant for the mixed method.

For a circular shallow arch subject to a uniform pressure, (8) reduces to

$$\begin{aligned}S(\boldsymbol{\phi})=\frac{-(1-\nu^2)}{2E\delta}\int_0^a N^2\,d\theta-\frac{6(1-\nu^2)}{E\delta^3}\int_0^a M^2\,d\theta\\+\int_0^a\left(\frac{u'}{R}-\frac{w}{R}+\frac{1}{2R^2}(w')^2\right)N-\frac{w''}{R^2}M\,d\theta-\lambda\int_0^a w\,d\theta.\end{aligned}$$

In our numerical experiments we took s as the parameter and λ as a dependent variable throughout, not just in the neighborhood of λ_{cr}. To solve the linear system (19) we used the multigrid method where we modified (13)–(14) to

(32) Set $\lambda^{q,k,i}=s-c(\mathbf{w}^{q,k,i-1})^T\tilde{M}\bar{f}_0$,

(33) Solve $M\boldsymbol{\sigma}^{q,k,i}=B\mathbf{u}^{q,k,i-1}-\mathbf{f}^q$ for $\boldsymbol{\sigma}^{q,k,i}$,

(34) Set $\mathbf{u}^{q,k,i}=\mathbf{u}^{q,k,i-1}-\beta(B^T\boldsymbol{\sigma}^{q,k,i}+C\mathbf{u}^{q,k,i-1}-\lambda^{q,k,i}\tilde{M}f_0-\mathbf{g}^q)$,

where $\mathbf{w}$ denotes the unknowns associated with w^h. The updating of λ in (32) preserves the same structure in the linear system as was present when λ was used as the parameter.

In our most extensive experiment, we took the radius of curvature $R = 100$ inches, the thickness of the arch $\delta = 2$ inches, and $a = 1$. We also took $E = 10^7$ psi and $\nu = 0.3$. Initially we took the constant c in (30) to be 1, but ran into some difficulty getting by λ_{cr} with (32)–(33). However, with $c = 10$, we had no difficulty. We determined λ_{cr} to be 749.03958 using a grid size $h = \frac{1}{4}$. This is in good agreement with the analytic determination of λ_{cr} given in [6].

Acknowledgment. The author wishes to thank Ms. Yannjun Hsu Chang for programming assistance.

REFERENCES

[1] G. J. Fix, R. A. Nicolaides and M. D. Gunzburger, *On mixed finite element methods for a class of nonlinear boundary value problems*, in Computational Methods in Nonlinear Mechanics, J. T. Oden, ed., North-Holland, New York, 1980, pp. 245–260.

[2] H. B. Keller, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359–384.

[3] L. Mansfield, *On mixed finite element methods for elliptic equations*, Comp. and Maths. with Appls., 7 (1981), pp. 59–66.

[4] ———, *Finite element methods for nonlinear shell analysis*, Numer. Math., 37 (1981), pp. 121–131.

[5] R. A. Nicolaides, *On the l^2 convergence of an algorithm for solving finite element equations*, Math. Comp., 31 (1977), pp. 892–906.

[6] H. L. Schreyer and E. F. Masur, *Buckling of shallow arches*, J. Eng. Mech. Div., ASCE, 92 (1966), pp. 1–19.

[7] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice-Hall, Englewood Cliffs, N J, 1973.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

DISCRETE WEIGHTED MEAN APPROXIMATION OF A MODEL CONVECTION-DIFFUSION EQUATION*

E. C. GARTLAND, JR.†

Abstract. Five-point finite-difference approximations are considered for a model (linear, constant-coefficient) convection-diffusion equation in two dimensions. Standard difference schemes for such problems behave badly when the convective terms are dominant. A new discretization is derived from a local integral representation of the true solution. This derivation is analogous to the way that the discrete Laplacian can be derived from the mean-value property of harmonic functions, and it generalizes an approach due to Allen and Southwell [Quart. J. Mech. Appl. Math., 8 (1955), pp. 129–45]. Also discussed is how the strong upwind bias of this and other discretizations serves to make more stable some methods of the two-sweep or marching type for the direct solution of the resulting linear algebraic equations.

Key words. Nonstandard difference approximations, convection-diffusion or convective-transport equation, marching or two-sweep algorithms, local integral representations

Introduction. A model elliptic boundary-value problem, which embodies some numerically troublesome aspects, is considered. The problem is nonsymmetric with a strong first-order term, and it possesses a boundary layer.

Attention is restricted to uniform-mesh discretizations that are coarse with respect to the boundary-layer thickness. This is analogous to computing an "outer" solution, the problem of resolving the boundary layer being considered a separate component of a complete solution process. The difficulties to be overcome are twofold: (1) to determine an accurate discretization (here restricted to a five-point stencil) of the nonsymmetric differential operator and (2) to overcome the polluting effect that the boundary layer has on the discretization error.

The discretization is derived by replacing a local integral representation of the true solution with a quadrature rule. This derivation is analogous to the way that the discrete Laplacian can be derived from the mean-value property of harmonic functions, and it generalizes, in a certain sense, an approach attributed to Allen and Southwell. The polluting effect of the boundary layer is overcome, to a degree, by using asymptotic information about the true solution locally to improve the discretization at those points that "see" the boundary layer.

Also discussed is an efficient method for solving directly the resulting linear algebraic system of equations. This method is of the two-pass or marching type. Such schemes are not widely used because they are very unstable with respect to the growth of roundoff errors. For these convection-dominated problems, however, the stability is significantly improved.

1. The problem. Let R denote the rectangle $(0, a)\times(0, b)$. Consider the following model problem:

$$\begin{aligned} -\varepsilon\,\Delta u + u_x &= 0 \quad \text{in } R, \\ u &= \gamma \quad \text{on } \partial R. \end{aligned} \tag{1.1}$$

Here ε is a positive constant that is small compared to a, and γ is a function defined on the boundary ∂R. Such a problem is known as a convection-diffusion or convective-

* Received by the editors August 12, 1981, and in revised form February 19, 1982. A portion of this work was completed in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Mathematics at Purdue University, West Lafayette, Indiana.

† Mathematics Department, Southern Methodist University, Dallas, Texas, 75275.

transport problem. The unknown u can be thought of as a concentration (of a chemical, of heat, of vorticity, etc.) that is convected, by a stream flowing the positive x direction, and diffused. The parameter ε gives the strength of the dispersive effects relative to the convective effects.

The form of the solution to (1.1) is complicated. It has a regular boundary layer of thickness $O(\varepsilon)$ along the downstream edge, $x = a$, and parabolic or shear layers of thickness $O(\varepsilon^{1/2})$ along the top and bottom edges, $y = b$ and $y = 0$. A thorough discussion of this behavior can be found in [8].

When $\varepsilon/a \ll 1$, the problem is said to be convection dominated, and its numerical approximation is difficult. This difficulty is due to the presence of the boundary layers. Approximation in or near these layers is degraded by the rapidly varying nature of the solution there. Even in the parts of the region away from the layers, standard approximations are affected with severe stability restrictions due to the existence of nonpolynomial-like components of the true solution [12, pp. 36–41].

Various approaches for overcoming these difficulties have been proposed for the case of finite-difference approximations. These include upwind differencing of the convective term (credit for this approach is difficult to ascribe, see [12, p. 64]), using exponentially weighted differences (this idea is attributed to Allen and Southwell [2], see also [1] and [9]), and using a modified upwind scheme (which has rational weight functions) [3], [10].

Here I derive a new five-point difference scheme that follows in a natural way from a local integral representation of the true solution. The scheme is a form of discrete weighted mean-value property, and it generalizes, in a certain sense, the approach of Allen and Southwell cited above.

2. One-dimensional problem. For the sake of illustration and motivation, let us consider the one-dimensional analogue of (1.1):

$$-\varepsilon u_{xx} + u_x = 0, \quad 0 < x < a, \quad u(0) = \gamma_0, \quad u(a) = \gamma_a. \tag{2.1}$$

The solution of this problem can easily be found; it is a linear combination of the functions ϕ and ψ defined by $\phi(x) = 1$ and $\psi(x) = \exp(x/\varepsilon)$. More important is the fact that one can find a difference equation that is exact for this problem, that is, a difference equation for which the restriction of u (the solution of (2.1)) provides the unique solution.

For a uniform mesh, this difference equation is

$$\begin{gathered} -\exp\left(\frac{h}{2\varepsilon}\right) u_{j-1} + 2\cosh\left(\frac{h}{2\varepsilon}\right) u_j - \exp\left(\frac{-h}{2\varepsilon}\right) u_{j+1} = 0, \qquad j = 1, \cdots, n, \\ u_0 = \gamma_0 \qquad u_{n+1} = \gamma_a. \end{gathered} \tag{2.2}$$

Here $h = a/(n+1)$, $x_j = jh$, and $u_j = u(x_j)$.

This difference scheme can be derived in various ways. The simplest and most direct way is to require that it be exact on the null vectors (ϕ and ψ above) of the differential operator L defined by $Lv = -\varepsilon v_{xx} + v_x$; this leads to a system of two linear algebraic equations in the three unknown coefficients of the difference scheme. Another way (and one that will help us to generalize this approach to the two-dimensional problem) is to observe that the scheme follows from a local integral representation of the true solution.

Let x_{j-1}, x_j, and x_{j+1} be three consecutive mesh points; let g_j be the local Green's function for L on the subinterval (x_{j-1}, x_{j+1}) with Dirichlet boundary conditions at

x_{j-1} and x_{j+1}; and let ϕ_j and ψ_j span the null space of L and satisfy $\phi_j(x_{j-1})=1$, $\phi_j(x_{j+1})=0$, $\psi_j(x_{j-1})=0$, and $\psi_j(x_{j+1})=1$. Then on $[x_{j-1}, x_{j+1}]$, any sufficiently smooth function v has the representation

$$v(x)=v(x_{j-1})\phi_j(x)+v(x_{j+1})\psi_j(x)+\int_{x_{j-1}}^{x_{j+1}} g_j(x,\xi)Lv(\xi)\,d\xi. \tag{2.3}$$

The difference equation (2.2) follows from this representation (up to normalization) by replacing v by u, which satisfies $Lu=0$, and x by x_j.

An important aspect of the difference scheme (2.2) is its strong upwind bias. While it has the usual $[-1, 2, -1]$ form as $h\to 0$, for values of h large compared to ε it has the form $[-1, 1+\delta, -\delta]$, where $\delta \ll 1$. And if the convection is dominant enough, i.e., if ε is small enough, this situation could persist through the whole range of feasible values for h on a given computer.

3. Two-dimensional problem. Let us return our attention to the two-dimensional problem (1.1). We seek a local representation of the solution that is analogous to the integral representation (2.3). Let Ω and L be a sufficiently nice domain and linear, second-order, elliptic partial differential operator; let g be the Green's function for L on Ω with Dirichlet boundary conditions on $\partial\Omega$, and let B denote the boundary differential operator for the adjoint problem. It is well known (see, for example, [11]) that for (x, y) in Ω, a sufficiently smooth function v has the representation

$$v(x,y)=\int_{\partial\Omega} B_{\xi\eta}g(x,y;\xi,\eta)v(\xi,\eta)\,d\sigma_{\xi\eta}+\iint_{\Omega} g(x,y;\xi,\eta)Lv(\xi,\eta)\,d\xi\,d\eta. \tag{3.1}$$

Here σ denotes arc length. When Ω is taken to be a small neighborhood of the given interior point (x, y), this representation provides the two-dimensional analogue of (2.3).

If we let L be the negative of the Laplacian and Ω the disk of radius h centered at the origin, then we can express the Green's function in terms of the natural logarithm, and we get, on evaluation at the center of the disk,

$$v(0,0)=\frac{1}{2\pi}\int_0^{2\pi} v(h\cos\theta, h\sin\theta)\,d\theta+\frac{1}{2\pi}\int_0^{2\pi}\int_0^h \ln\frac{h}{r}\cdot(-\Delta v)(r,\theta)r\,dr\,d\theta.$$

Here r and θ are the usual polar coordinates, and we have also used the fact that $B_{\xi\eta}$ is equal to the normal derviative $(=\partial_r)$. The above reduces to the familiar mean-value property when v is harmonic. A composite trapezoid rule approximation to this mean-value relation using the values of v at $(h, 0)$, $(0, h)$, $(-h, 0)$, and $(0, -h)$ gives the standard five-point discretization of the Laplacian.

Consider now the same disk domain with L equal to $-\varepsilon\Delta+\partial_x$, the convection-diffusion operator. The Green's function can be expressed in terms of the Bessel functions I_0 and K_0, and when $(x, y)=(0, 0)$, it is given by

$$g(0,0;r,\theta)=\frac{\exp\dfrac{-r\cos\theta}{2\varepsilon}}{2\pi\varepsilon I_0\left(\dfrac{h}{2\varepsilon}\right)}\left[I_0\left(\frac{h}{2\varepsilon}\right)K_0\left(\frac{r}{2\varepsilon}\right)-K_0\left(\frac{h}{2\varepsilon}\right)I_0\left(\frac{r}{2\varepsilon}\right)\right].$$

Substitution in (3.1) and simplification by use of the formula $I_0(z)K_0'(z) - K_0(z)I_0'(z) = -1/z$ [13, pp. 79–80] gives the representation

$$v(0,0) = \frac{1}{2\pi I_0\left(\frac{h}{2\varepsilon}\right)} \int_0^{2\pi} \exp\left(\frac{-h}{2\varepsilon}\cos\theta\right) v(h\cos\theta, h\sin\theta)\,d\theta$$

$$+ \int_0^{2\pi}\int_0^h g(0,0;r,\theta)(-\varepsilon\Delta v + v_x)(r,\theta)r\,dr\,d\theta,$$

valid for sufficiently smooth v. When v is replaced by the solution u (which satisfies $Lu = 0$) and (x, y) is a point in the open rectangle R, this becomes the weighted mean-value relation

$$\text{(3.2)}\qquad u(x,y) = \frac{1}{2\pi I_0\left(\frac{h}{2\varepsilon}\right)} \int_0^{2\pi} \exp\left(\frac{-h}{2\varepsilon}\cos\theta\right) u(x + h\cos\theta, y + h\sin\theta)\,d\theta,$$

valid for all h sufficiently small so that the disk of radius h centered at (x, y) is contained in $\bar{R}$.

Just as the mean-value property of harmonic functions can be used to derive the standard discretization of the Laplacian, the weighted mean-value relation above can be used to derive a discretization of the convection diffusion operator. A five-point stencil is obtained by replacing $u(x + h\cos\theta, y + h\sin\theta)$ in (3.2) by the trigonometric polynomial t_4 defined by $t_4(\theta) = a_0 + a_1\cos\theta + b_1\sin\theta + a_2\cos 2\theta$ that interpolates to it at $\theta = 0$, $\pi/2$, π, and $3\pi/2$. The coefficients a_0, a_1, b_1, and a_2 can be expressed as linear combinations of the values $u(x + h, y)$, $u(x, y + h)$, $u(x - h, y)$, and $u(x, y - h)$; the resulting integrals can be evaluated in terms of the Bessel functions I_n with the help of the formula

$$\frac{1}{2\pi}\int_0^{2\pi} \exp(-z\cos\theta)\cos n\theta\,d\theta = (-1)^n I_n(z)$$

[13, p. 18]; and the stencil coefficients can be expressed in terms of I_1/I_0 by use of the formula $I_{n-1}(z) - I_{n+1}(z) = (2n/z)I_n(z)$ [13, p. 79]. The following five-point stencil results:

$$\text{(3.3)}\qquad \frac{2\varepsilon}{h^2}\cdot \begin{array}{|c|c|c|}
\cline{2-2}
\multicolumn{1}{c|}{} & \frac{-2\varepsilon}{h}\frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) & \multicolumn{1}{c}{} \\
\hline
-1-\left(1-\frac{2\varepsilon}{h}\right)\frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) & 2 & -1+\left(1+\frac{2\varepsilon}{h}\right)\frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) \\
\hline
\multicolumn{1}{c|}{} & \frac{-2\varepsilon}{h}\frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) & \multicolumn{1}{c}{} \\
\cline{2-2}
\end{array}$$

Let R_h denote the set of interior nodes of a uniform rectangular partition of the region R with mesh width h, m vertical mesh lines, and n horizontal mesh lines; let ∂R_h denote the boundary nodes. Let L_h denote the finite difference operator associated with the stencil above, and let U_{ij} and γ_{ij} denote the discrete approximation to u and the value of γ at the ijth nodal point. The difference equation

$$(3.4) \qquad L_h U_{ij} = 0 \quad \text{in } R_h, \qquad U_{ij} = \gamma_{ij} \quad \text{on } \partial R_h$$

I refer to the *discrete weighted mean* approximation of (1.1).

As $h \to 0$, the stencil (3.3) has the usual "diffusion" form

$$\frac{\varepsilon}{h^2}\cdot \begin{array}{|c|c|c|}\hline & -1 & \\ \hline -1 & 4 & -1 \\ \hline & -1 & \\ \hline \end{array} + O\left(\frac{1}{h}\right).$$

When $h/\varepsilon \gg 1$, however, the following asymptotic expansion is valid;

$$\frac{4}{h}\left(\frac{\varepsilon}{h}\right)\cdot \begin{array}{|c|c|c|}\hline & \frac{-\varepsilon}{h} & \\ \hline -1+\frac{3}{2}\frac{\varepsilon}{h} & 1 & \frac{1}{2}\frac{\varepsilon}{h} \\ \hline & \frac{-\varepsilon}{h} & \\ \hline \end{array} + O\left(\left(\frac{\varepsilon}{h}\right)^3\right).$$

Thus, if the mesh size is coarse compared to the boundary-layer thickness, the discretization has a strong upwind bias, and in the limit as $\varepsilon \to 0$, with h held constant, it becomes the backwards difference approximation to $u_x = 0$.

There is a number, call it z^*, approximately equal to $1.545(\pm .005)$ at which the "east" coefficient of the stencil (3.3) vanishes. When $h/2\varepsilon \leqq z^*$, the discrete weighted mean approximation is monotone (that is, $L_h\phi_{ij} \geqq 0$ in R_h and $\phi_{ij} \geqq 0$ on ∂R_h imply $\phi_{ij} \geqq 0$ in R_h), and the discretization error can be bounded in terms of the truncation error (see, for example, [7] and Theorem 2 below). When $h/2\varepsilon > z^*$, however, this coefficient is of the wrong sign (positive), and the approximation is not monotone. This can be viewed as an exhibition of the fact that mesh sizes larger than this critical value are too coarse to show the elliptic nature of the problem.

Nevertheless, the following theorem shows that the discrete weighted mean approximation is uniquely solvable for all positive ε and h.

THEOREM 1. *Let ε and h be positive. Then the difference equation* (3.4) *has a unique solution.*

Proof. It is sufficient to show that zero is not an eigenvalue of L_h in R_h with homogeneous Dirichlet conditions on ∂R_h. When $h/2\varepsilon \leqq z^*$, L_h is monotone, so that $L_h\phi_{ij} = 0$ in R_h and $\phi_{ij} = 0$ on ∂R_h imply ϕ_{ij} is zero.

Let c_n, c_s, c_e, c_w denote the "north", "south", "east", and "west" coefficients of the stencil (3.3). If $h/2\varepsilon > z^*$, then $c_e > 0$ and the eigenvalues and associated eigenfunctions of L_h are given by

$$\lambda_{kl} = \frac{4\varepsilon}{h^2} + 2(c_s c_n)^{1/2}\cos\frac{l\pi}{n+1} + i2(-c_w c_e)^{1/2}\cos\frac{k\pi}{m+1}, \qquad k = 1, \cdots, m \quad \text{and} \quad l = 1, \cdots, n,$$

and

$$\phi_{\mu\nu}^{(kl)} = i^{\mu}\left(-\frac{c_w}{c_e}\right)^{\mu/2}\left(\frac{c_s}{c_n}\right)^{\nu/2}\sin\frac{\mu k\pi}{m+1}\sin\frac{\nu l\pi}{n+1}, \qquad \mu = 1, \cdots, m \quad \text{and} \quad \nu = 1, \cdots, n.$$

From the inequality $2I_1(z) < zI_0(z)$ (valid for z positive), it follows that c_s and c_n are greater than $-\varepsilon/h^2$, and we have that $\operatorname{Re}(\lambda_{kl}) > 4\varepsilon/h^2 - 2(c_s c_n)^{1/2} > 2\varepsilon/h^2$.

Thus zero cannot be an eigenvalue in this case either. □

The discretization error of the discrete weighted mean approximation is appraised in the following theorem.

THEOREM 2. *Let τ_{ij} denote the truncation error ($\tau_{ij} = (L_h - L)u_{ij}$) of the discrete weighted mean approximation. Then there exist points ξ_i, ξ_i', and η_j satisfying $x_{i-1} < \xi_i$, $\xi_i' < x_{i+1}$ and $y_{j-1} < \eta_j < y_{j+1}$ such that*

$$\begin{aligned}\tau_{ij} &= -2\varepsilon\frac{I_2}{I_0}\left(\frac{h}{2\varepsilon}\right)u_{xx}(x_i, y_j)\\ &\quad -2\varepsilon\frac{h^2}{12}\left\{\left[1 - \frac{2\varepsilon}{h}\frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right)\right]u_{xxxx}(\xi_i, y_j) + \left[\frac{2\varepsilon}{h}\frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right)\right]u_{yyyy}(x_i, \eta_j)\right\}\\ &\quad +\frac{h^2}{3}\left[\frac{2\varepsilon}{h}\frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right)\right]u_{xxx}(\xi_i', y_j).\end{aligned}$$

Let e_{ij} denote the error $u_{ij} - U_{ij}$, and let $\|\cdot\|$ denote the grid maximum norm. Then there exists a constant C, independent of h, such that

$$\|e_{ij}\| \leqq Ch^2.$$

Proof. The expression for τ_{ij} follows from Taylor's formula, some straightforward manipulations, and the identity $I_0(z) - 2I_1(z)/z = I_2(z)$ [13, p. 79].

The discretization error, e_{ij}, satisfies $L_h e_{ij} = \tau_{ij}$ in R_h and $e_{ij} = 0$ on ∂R_h. Let h^* denote the value $2\varepsilon z^*$, the critical point at which the "east" coefficient changes sign. When $0 < h \leqq h^*$, L_h is monotone, the inverses L_h^{-1} can be bounded uniformly in h as follows. Let ϕ be the comparison function defined by $\phi(x, y) = y(b-y)/2\varepsilon$; then $L_h\phi_{ij} = 1$. Given a mesh function v, observe that

$$L_h(\|L_h v_{ij}\|\phi_{ij}) = \|L_h v_{ij}\| \geqq \pm L_h v_{ij} \quad \text{for all } i \text{ and } j.$$

Thus $|v_{ij}| \leqq \|L_h v_{ij}\|\phi_{ij}$ and $\|v_{ij}\| \leqq \|L_h v_{ij}\|\,\|\phi_{ij}\|$. It follows that $\|L_h^{-1}\| \leqq \|\phi_{ij}\| = b^2/8\varepsilon$.

For values of h greater than h^* and less than $h_{\max}$ (the maximum value permitted by the dimensions of the region), Theorem 1 guarantees that the bounded inverses L_h^{-1} all exist. Let the constant C_1 denote the maximum of the norms of these,

$$C_1 = \max\{\|L_h^{-1}\|: h^* \leqq h \leqq h_{\max}\}.$$

Then

$$\|e_{ij}\| \leqq \max\left(C_1, \frac{b^2}{2\varepsilon}\right)\|\tau_{ij}\|.$$

The inequalities

$$\frac{I_1}{I_0}(z) \leqq \frac{z}{2} \quad \text{and} \quad \frac{I_2}{I_0}(z) \leqq \left(\frac{z}{2}\right)^2,$$

valid for z greater than or equal to zero, are easy to establish and can be used to bound the truncation error. Define C_2 by

$$C_2 = \frac{1}{8\varepsilon}\|u_{xx}\| + \frac{1}{6}\|u_{xxx}\| + \frac{\varepsilon}{12}(2\|u_{xxxx}\| + \|U_{yyy}\|).$$

Straightforward manipulations give $\|\tau_{ij}\| \leqq C_2h^2$.

Thus the inequality of the theorem is established with the constant C given by $C = C_2 \max\{C_1, b^2/2\}$. $\square$

4. Boundary-layer improvement. Several numerical experiments were performed comparing various five-point discretizations of model problems of the form (1.1); these tests are discussed in § 6. The discrete weighted mean approximation was found to be superior to all of the other tested schemes with respect to both the average and maximum norm measures of relative error. Most of the experiments were conducted on meshes that were coarse relative to the boundary-layer thickness ε, and all of the methods, to some extent, suffered from the same problem: as the mesh was refined, the average error was reduced by an acceptable amount, but the maximum relative error was reduced only slightly (if at all).

A consideration of the truncation error associated with the discrete weighted mean approximation suggests a remedy for the problem. The truncation error can be represented as a quadrature error—this is natural considering the derivation of the scheme. Let e_{ij} denote the difference $u_{ij} - U_{ij}$ between the true solution and the discrete approximation to that solution at the ijth mesh point. Some simple manipulations give

$$L^h e_{ij} = \frac{4\varepsilon}{h^2} \cdot \frac{1}{2\pi I_0\left(\frac{h}{2\varepsilon}\right)} \int_0^{2\pi} \exp\left(\frac{-h}{2\varepsilon}\cos\theta\right)\left[u(x_i + h\cos\theta, y_j + h\sin\theta) - t_4(\theta)\right] d\theta.$$

We see that the truncation error is the error created by replacing $u(x + h\cos\theta, y + h\sin\theta)$ in (3.2) by the trigonometric interpolating polynomial t_4.

For convection-dominated approximations, where $h > \varepsilon$, this truncation error can be expected to be small at all points except those along the vertical mesh line $x = a - h$. At these points, the circular integration path goes through the boundary layer, and the trigonometric polynomial t_4 cannot be expected to approximate well $u(x + h\cos\theta, y + h\sin\theta)$.

The first-order term in the asymptotic expansion of the true solution is given by

$$u(x, y) \sim u(0, y) + [u(a, y) - u(0, y)]\exp((x - a)/\varepsilon). \tag{4.1}$$

By using this information locally, we can derive a better discretization for the points along the last mesh line. In the weighted mean-value relation (4.2), replace $u(x + h\cos\theta, y + h\sin\theta)$ by the function q_4 defined by

$$q_4(\theta) = b_0 + \exp((x + h\cos\theta - a)/\varepsilon)(a_0 + a_1\cos\theta + b_1\sin\theta)$$

that interpolates to it at $\theta = 0$, $\pi/2$, π, and $3\pi/2$. As before, the coefficients b_0, a_0, a_1, and b_1 can be expressed as linear combinations of the values $u(x + h, y)$, $u(x, y + h)$, $u(x - h, y)$, and $u(x, y - h)$, and the stencil coefficients can be expressed in terms of Bessel functions. Let ω denote the quantity $\exp(-h/\varepsilon)$. We have the following stencil:

$$\begin{array}{|c|c|c|}
\hline
 & -\frac{1}{2} + \frac{1+\omega}{2(1-\omega)} \frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) & \\
\hline
\frac{-1}{1-\omega} \frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) & 1 & \frac{-\omega}{1-\omega} \frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) \\
\hline
 & -\frac{1}{2} + \frac{1+\omega}{2(1-\omega)} \frac{I_1}{I_0}\left(\frac{h}{2\varepsilon}\right) & \\
\hline
\end{array} \tag{4.2}$$

By using the stencil (4.2) along the boundary-layer mesh line and the stencil (3.3) at all other points, we obtain a dramatic improvement in accuracy and achieve our best results. This stencil can be used in a similar way to markedly improve the accuracy of other approximations, such as upwind differences. Both of these facts are documented in § 6.

5. A direct solution method. The strong upwind bias possessed by the stencil (3.3) can be used to advantage in the direct solution of the difference equations (3.4); this applies as well to other discretizations of our problem that possess this same property. Consider an algorithm of the two-sweep or marching type. Such methods go back at least to von Mises (who discussed them in a seminar at Harvard around 1955 but dismissed them as unstable[1]), and appear at various places in the literature: Roache [12], who refers to these as error vector propagation (EVP) methods, uses them routinely; Birkhoff and George discussed them in [6, pp. 230–6], and Bank and Rose analyzed them in detail from the standpoint of complexity and roundoff-error instability [5].

Let U_i denote $(U_{i1}, \cdots, U_{in})^t$, the column vector of interior values of U_{ij} along the ith vertical mesh line; let T denote the $n \times n$, tridiagonal, Toeplitz matrix $[c_s, 1, c_n]$ (i.e., with lower codiagonal entries equal to c_s, diagonal entries equal to 1, and upper codiagonal entries equal to c_n), and let B_i denote the n-vector containing the boundary data associated with the grid points along the ith vertical mesh line. Were U_m known, the remaining vectors $U_{m-1}, \cdots, U_1$ could be computed by using the stencil (3.3) in a three-term recursion and "sweeping" upstream—the reason for sweeping in this direction will be made clear in the sequel—according to

$$(5.1) \quad U_{m+1}=0, \quad U_m = U_m, \quad -c_wU_{i-1}=TU_i+c_eU_{i+1}-B_i, \quad i=m, m, -1, \cdots, 2.$$

The essence of the two-sweep or marching method is to (1) do a sweep with U_m set equal to zero, (2) use the result of this first sweep together with the unsatisfied boundary condition at the upstream end to compute the correct initial values for U_m, and (3) do another sweep with these correct values to compute the solution on all of the remaining interior grid points.

This method is very simple and very efficient. The marching phases require $O(mn)$ work and storage, and it was shown by Bank and Rose [4], [5] that for separable problems on rectangular domains, the second phase, which requires the solution of a full $n \times n$ linear system for the correct initial values of U_m, can be accomplished with this same complexity. This complexity is of optimal order, i.e., of the same order as the number of unknowns.

The shortcoming of the method is its instability with respect to growth of roundoff error during the marching phases. Let p_k be the polynomial of degree k defined by

$$(5.2) \quad p_{-1}(x)=0, \quad p_0(x)=1, \quad -c_wp_{k+1}(x)=xp_k(x)+c_ep_{k-1}(x), \quad k=0, 1, \cdots.$$

The polynomial p_k is related to the modified Chebyshev polynomial S_k, which satisfies the above recursion when $c_w = c_e = -1$, by the following:

$$p_k(x)=\left(\frac{c_e}{c_w}\right)^{k/2} S_k\left(\frac{x}{(c_wc_e)^{1/2}}\right).$$

Let $\rho(T)$ denote the spectral radius of T. During a sweep, an initial error can be amplified by a factor of $p_m(\rho(T))$—a detailed error analysis for this type of algorithm

[1] Garrett Birkhoff, 1981, personal communication.

can be found in [5]. In the prototypical case of the discrete Laplacian, where $c_n = c_s = c_e = c_w = -\frac{1}{4}$, $p_m(\rho(T))$ is approximately 6^m, and such severe exponential growth of roundoff error is indeed observed.

The presence of strong convection (and its consequent upwind-biased stencil) provides a considerable improvement. When $\varepsilon/h \ll 1$, the "west" stencil coefficient, c_w, is approximately equal to $\rho(T)$ so that the following asymptotic expansion is valid:

$$p_m(\rho(T)) \sim \left[1+\frac{7}{2}\left(\frac{\varepsilon}{h}\right)+O\left(\left(\frac{\varepsilon}{h}\right)^2\right)\right]^m.$$

Table 1 compares the amplification factors $p_m(\rho(T))$ for discrete Laplacian and the discrete weighted mean approximation of the convection-diffusion equation (2.1) with ε taking on the values 1, .1, .01, and .001.

TABLE 1
Amplification factors

		Convection-diffusion			
m	Discrete Laplacian	$\varepsilon = 1$	$\varepsilon = .1$	$\varepsilon = .01$	$\varepsilon = .001$
5	5.5 (3)	3.6 (3)	2.4 (2)	2.7 (0)	1.1 (0)
10	4.0 (7)	2.6 (7)	8.3 (5)	3.7 (1)	1.5 (0)
15	2.8 (11)	1.8 (11)	4.2 (9)	1.7 (3)	2.5 (0)
20	1.9 (15)	1.2 (15)	2.4 (13)	2.2 (5)	5.0 (0)
30	9.0 (22)	5.6 (22)	9.3 (20)	3.3 (10)	3.3 (1)
40	4.1 (30)	2.5 (30)	3.9 (28)	3.2 (16)	4.2 (2)

On a CDC 6600, for example, the single floating-point precision of which is at least fourteen decimal digits, amplification factors of 10^{10} or so can be tolerated for these problems. So when $\varepsilon = .01$, for instance, a 30×30 problem can be solved directly in this way (as opposed to a maximum 13×13 discrete Laplace equation); when $\varepsilon = .001$, this threshold is not reached until $m > 80$. Moreover, $p_m(\rho(T))$ can be easily computed from (5.2) and is a reliable measure of the possible errors due to marching instabilities.

The basic algorithm used to do the calculations in the next section is the following:

$$\begin{aligned}
&V_{m+1} = 0\\
&V_m = 0\\
&-c_wV_{i-1} = TV_i + c_eV_{i+1} - B_i, \quad i = m, m-1, \cdots, 1\\
(5.3)\qquad &p_m(T)U_m = -V_0\\
&U_{m+1} = 0\\
&U_m = U_m\\
&-c_wU_{i-1} = TU_i + c_eU_{i+1} - B_i, \quad i = m, m-1, \cdots, 2.
\end{aligned}$$

The $n\times n$ linear system $p_m(T)U_m = V_0$ is solved by using the factorization

$$p_m(T) = \begin{cases} \displaystyle\prod_{i=1}^{m/2}\left(\frac{1}{c_w^2}T^2 - 4\frac{c_e}{c_w}\cos^2\frac{i\pi}{m+1}\right), & m \text{ even},\\[2ex] \displaystyle\frac{1}{c_w}T\prod_{i=1}^{(m-1)/2}\left(\frac{1}{c_w^2}T^2 - 4\frac{c_e}{c_w}\cos^2\frac{i\pi}{m+1}\right), & m \text{ odd}.\end{cases}$$

The algorithm differs from the algorithm of [5, p. 798] only in the slightly different form of the recursion and in the different factorization of p_m, which is chosen to avoid the need for complex arithmetic in the convection-dominated approximations where c_w and c_e are of opposite signs. It fits essentially into the context of the variable-coefficient discretizations analyzed in [4].

For cases where this marching algorithm is not stable enough to solve the whole problem directly, generalized marching [5], which makes use of this basic algorithm on subproblems and is analogous to multiple shooting, is a viable alternative. A marching (or generalized marching) algorithm can also be used to solve the problems associated with the boundary-layer modified discretizations, which are discussed in the previous section and have the property of using a different stencil along the last mesh line. In this case, the sweeping phases are similar to those of (5.3), but the second phase requires the calculation of the m zeros of the polynomial taking the place of p_m. This can be done by finding the eigenvalues of a tridiagonal $m \times m$ matrix (as in the variable-coefficient implementation of these schemes [4]), which can be accomplished by the QR algorithm, for example, and the complexity of the resulting algorithm is $O(m^2 + mn)$.

6. Numerical results. Numerical experiments were conducted for the model problem (1.1) on the unit square ($a = b = 1$) with various boundary conditions and a wide range of values for the parameter ε. The tests were performed on an $n \times n$ grid with n taking on values given by $n + 1 = 4$, 8, 16 and 32. The following methods were tested: standard central differences; upwind differencing of the convection term [12, p. 24]; exponentially weighted differences [2], [9]; a modified upwind scheme [3], [10]; the discrete weighted mean approximation (3.3); the discrete weighted mean approximation with boundary-layer improvement (§ 4); and upwind differences with the same boundary-layer improvement.

The performance of the standard central difference approximation was extremely poor, as expected, and this scheme can be immediately dismissed. The performances of the exponentially weighted and the modified upwind schemes were very similar: both had more uniform error distribution than the other schemes but low accuracy. Results for the exponentially weighted scheme are reported. The upwind scheme typifies this type of approximation: the accuracy is quite good (even for very coarse meshes) away from the boundary layer (as reflected in the average error column of Table 2), but it is very nonuniform and falls off sharply along the last mesh line adjacent to the boundary-layer. This latter situation is improved by utilizing the boundary-layer modified stencil (4.2). The discrete weighted mean approximation performed the best (with respect to both the maximum and, in particular, the average error) of the unmodified schemes, and the discrete weighted mean approximation with boundary-layer modification performed the best overall.

Two other schemes were tested. Both were derived from the weighted mean-value relation (3.2) in the same way that the discrete weighted mean approximation was derived, with one using piecewise linear splines and the other using periodic cubic splines. The results for neither of these are reported: the former yielded a monotone approximation but was not very accurate and required numerical quadrature to evaluate the stencil coefficients, and the latter was comparable to the discrete weighted mean approximation but also required numerical quadrature.

Table 2 contains representative results for 5 of the methods, along with the relative errors associated with the asymptotic expansion (4.1), for the model problem with ε equal to .01 and with the boundary conditions

$$u(x, 0) = u(x, 1) = 0, \quad u(0, y) = y(1-y), \quad u(1, y) = 2y(1-y).$$

TABLE 2
Discretization errors ($\varepsilon = .01$). *Methods*: (0) *one-term asymptotic expansion,* (1) *exponentially weighted differences,* (2) *upwind differences,* (3) *discrete weighted mean approximation,* (4) *upwind differences with boundary-layer modification, and* (5) *discrete weighted mean approximation with boundary-layer modification.*

Method	$n+1$	Relative error	
		Maximum	Average
0	4	7.7 (−2)	5.1 (−2)
1		7.7 (−2)	5.1 (−2)
2		4.9 (−2)	1.8 (−2)
3		1.8 (−2)	7.4 (−3)
4		1.6 (−2)	6.9 (−3)
5		1.5 (−2)	6.5 (−3)
0	8	9.0 (−2)	5.1 (−2)
1		8.8 (−2)	5.0 (−2)
2		8.9 (−2)	1.4 (−2)
3		4.0 (−2)	6.6 (−3)
4		7.2 (−3)	1.8 (−3)
5		6.5 (−3)	1.5 (−3)
0	16	9.6 (−2)	5.1 (−2)
1		7.0 (−2)	3.7 (−2)
2		1.6 (−1)	1.3 (−2)
3		6.4 (−2)	4.7 (−3)
4		3.2 (−3)	5.3 (−4)
5		2.2 (−3)	3.5 (−4)
0	32	9.6 (−2)	5.1 (−2)
1		3.0 (−2)	1.6 (−2)
2		2.2 (−1)	1.0 (−2)
3		5.2 (−2)	1.8 (−3)
4		1.4 (−2)	1.1 (−3)
5		2.3 (−3)	1.8 (−4)

7. Conclusion. It has been shown that the approach of the discrete weighted mean approximation possesses several advantages for a problem on which standard (polynomial-based) approximations fail. The effectiveness of this approach has been demonstrated by numerical experiments. It has the advantage of treating the differential operator as a whole and, in a sense, generalizes the ideas of Allen and Southwell for the discretization of these types of operators.

The approach can readily be adapted to the slightly more general problem

$$-\varepsilon\Delta u + \mathbf{U}\circ\nabla u + qu = f \quad \text{in } R, \qquad u = \gamma \quad \text{on } \partial R,$$

where $\mathbf{U} = (U\cos\theta_0, U\sin\theta_0)$ and ε, U, θ_0, and q are constants. The local Green's function for the disk of radius h centered at the origin is given by

$$g(0,0;r,\theta) = \frac{\exp\left[-\dfrac{Ur}{2\varepsilon}\cos(\theta-\theta_0)\right]}{2\pi\varepsilon I_0(\sigma h)}[I_0(\sigma h)K_0(\sigma r) - K_0(\sigma h)I_0(\sigma r)]$$

where

$$\sigma=\frac{(U^2+4\varepsilon q)^{1/2}}{2\varepsilon}.$$

And this gives rise to a local integral representation for the solution

$$u(x,y)=\frac{1}{2\pi I_0(\sigma h)}\int_0^{2\pi}\exp\left[-\frac{Uh}{2\varepsilon}\cos(\theta-\theta_0)\right]u(x+h\cos\theta,y+h\sin\theta)\,d\theta$$
$$+\int_0^{2\pi}\int_0^h g(x,y;r,\theta)f(x+r\cos\theta,y+r\sin\upsilon)r\,dr\,d\theta.$$

Replacing the integrals above by appropriate quadrature rules will yield the discrete weighted mean approximation for this problem. Also, variable-coefficient problems, in which U, θ_0, and q above are functions of x and y, can be handled by using at each mesh point a discrete weighted mean approximation based upon the values of the coefficient functions evaluated at that point (and treated as constants).

The approach generalizes to other differential operators and can be expected to provide an improvement in those cases where polynomial-based approximations have difficulty. It also has the advantage of permitting the local use of asymptotic information, as done in § 5, to improve the accuracy of the approximation. This is, of course, rather problem specific, but it does provide a mechanism for using this information (here the first two terms of an asymptotic expansion of the true solution) when it is available.

The two main difficulties of the discrete weighted mean approximation are its complicated derivation and analysis and the necessity of evaluating special functions. The latter point is not too detrimental, since the only nonstandard function that is required is the ratio of two Bessel functions I_1/I_0, which can be approximated to sufficient accuracy (over the pertaining range of values of its argument) by a rational function or asymptotic expansion.

This discretization, like others of such convection-dominated problems possesses a strong upwind bias, and this helps to make more stable direct solution algorithms of the marching or two-sweep variety. This is important because these algorithms are of optimal computational complexity (in sufficiently nice geometries) and can afford significant savings in solving certain nonlinear problems, such as the vorticity transport equations, which are typically approximated by iteratively solving a sequence of linearized problems like ours [12].

Acknowledgment. Research on this problem was initiated while the author was a graduate student at Purdue University, and he gratefully acknowledges the stimulation and contributions of Professor Garrett Birkhoff (who was a visitor there during the fall semester of 1979) and Professor Robert E. Lynch.

REFERENCES

[1] D. N. de G. Allen, *A suggested approach to finite-difference representation of a differential equation, with an application to determine temperature distributions near a sliding contact*, Quart. J. Mech. Appl. Math., 15 (1962), pp. 11–33.

[2] D. N. de G. Allen and R. V. Southwell, *Relaxation methods applied to determine the motion, in two dimensions, of a viscous fluid past a fixed cylinder*, Quart. J. Mech. Appl. Math., 8 (1955), pp. 129–45.

[3] O. AXELSSON AND I. GUSTAFSSON, *A modified upwind scheme for convective transport equations and the use of a conjugate gradient method for the solution of non-symmetric systems of equations*, J. Inst. Math. Appl., 23 (1979), pp. 321–37.

[4] R. E. BANK, *Marching algorithms for elliptic boundary value problems. II: the variable coefficient case*, SIAM J. Numer. Anal., 14 (1977), pp. 950–70.

[5] R. E. BANK AND D. J. ROSE, *Marching algorithms for elliptic boundary value problems. I: the constant coefficient case*, SIAM J. Numer. Anal., 14 (1977), pp. 792–829.

[6] G. BIRKHOFF AND J. A. GEORGE, *Elimination by nested dissection*, in Complexity of Sequential and Parallel Numerical Algorithms, J. F. Traub, ed., Academic Press, New York, 1973, pp. 221–69.

[7] L. COLLATZ, *The Numerical Treatment of Differential Equations*, 2d ed., Springer, Berlin, 1960.

[8] W. ECKHAUS, *Asymptotic Analysis of Singular Perturbations*, North-Holland, Amsterdam, 1979.

[9] A. M. IL'IN, *Differencing scheme for a differential equation with a small parameter affecting the highest derivative*, Math. Notes, 6 (1969), pp. 596–602.

[10] V. A. GUSHCHIN AND V. V. SHCHENNIKOV, *A monotone difference scheme of second order accuracy*, USSR Computational Maths. and Math. Phys., 14 (1974), pp. 252–56.

[11] C. MIRANDA, *Partial Differential Equations of Elliptic Type*, 2d rev. ed. Springer, Berlin, 1970.

[12] P. J. ROACH, *Computational Fluid Dynamics*, rev., Hermosa, Alburquerque, NM, 1976.

[13] G. N. WATSON, *A Treatise on the Theory of Bessel Functions*, 2d ed., Cambridge Univ. Press, Cambridge, 1966.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

PRECONDITIONING AND COARSE GRID CORRECTIONS IN THE SOLUTION OF THE INITIAL VALUE PROBLEM FOR NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS*

P. J. VAN DER HOUWEN† AND H. B. DE VRIES†

Abstract. The numerical solution of nonlinear, time-dependent partial differential equations is discussed. An initial value problem for a system of ODE's is obtained by the method of lines, and an implicit linear multistep method is applied to this initial value problem. Using Newton type iteration the nonlinear implicit relations are replaced by a sequence of linear equations. The linear problems are preconditioned by applying incomplete LU-decomposition and are then solved by iterative refinement. The convergence is accelerated by introducing coarse grid corrections. Numerical examples are given and a comparison is made with other integration techniques.

Key words. Numerical analysis, method of lines, initial-boundary value problems, incomplete LU-decomposition, coarse grid corrections

1. Introduction.

When the method of lines is applied to an initial-boundary value problem for a (nonlinear) hyperbolic or parabolic differential equation, we often obtain a system of ODE's of the form

$$\frac{d^\nu y}{dt^\nu}=f(t, y), \qquad \nu=1,2, \tag{1.1}$$

with prescribed values for y (and dy/dt) at $t=t_0$. By applying a linear multistep method to this equation we are asked to solve at each time step the system of equations

$$y-b_0\tau_n^\nu f(t_{n+1}, y)=\sum_{l=1}^{k}[a_l y_{n+1-l}+b_l\tau_n^\nu f(t_{n+1-l}, y_{n+1-l})], \tag{1.2}$$

where y_n denotes the numerical solution at $t=t_n$, $\tau_n=t_{n+1}-t_n$ and $\{a_l, b_l\}$ are real coefficients. The (approximate) solution of (1.2) is identified with y_{n+1}.

Assuming that f is differentiable with respect to y we may define the iteration process

$$\begin{aligned}
&y^{(0)}=y^{(\text{pred})},\\
&y^{(j)}=[I-b_0\tau_n^\nu\tilde{J}_j]^{-1}[b_0\tau_n^\nu(J-\tilde{J}_j)y^{(j-1)}+\phi(u(j))], \qquad j=1,\cdots,M,\\
&\phi(u(j))=\sum_n+b_0\tau_n^\nu[f(t_{n+1}, y^{(u(j))})-Jy^{(u(j))}],\\
&J=\frac{\partial f}{\partial y}(t_{n+1}, y^{(0)}), \qquad 0\leqq u(j)\leqq u(j+1)\leqq j,
\end{aligned} \tag{1.3}$$

where $\sum_n$ denotes the right-hand side of equation (1.2), $y^{(\text{pred})}$ is some predictor, $\tilde{J}_j$, $j=1,\cdots,M$, are approximations to J and $u(j)$ is a piecewise constant function. This function will be called the *update function* because each time it changes its value a new right-hand side function is to be evaluated.

The scheme (1.3) contains several well-known iteration processes as special cases. For instance,

$$\tilde{J}_j=J, \qquad u(j)=j-1, \tag{1.4}$$

yields the *modified Newton–Raphson process.*

* Received by the editors December 16, 1980 and in revised form April 15, 1981.

† Mathematisch Centrum, Kruislaan 413, 1098 SJ Amsterdam, the Netherlands.

A second class of methods is based on a splitting of the Jacobian matrix J. Well-known methods arise if ($f(t, y)$ originating from a 2-dimensional problem)

$$\tilde{J}_j = \begin{cases} J_1, & j \text{ odd}, \\ J_2, & j \text{ even}, \end{cases} \qquad J_1 + J_2 = J, \tag{1.5}$$

where J_1 and J_2 are "simply structured matrices" (e.g., tridiagonal matrices). A slight modification of (1.5) is given by

$$\tilde{J}_j = J_1 + J_2 - b_0 \tau_n^\nu J_1 J_2 = J - b_0 \tau_n^\nu J_1 J_2. \tag{1.5$'$}$$

A third class of iteration methods is based on incomplete LU-decomposition. Let L^*U^* denote an incomplete LU-decomposition of the matrix $I - b_0\tau^\nu J$ (from now on the index n is omitted in the steps τ_n), i.e.,

$$I - b_0 \tau^\nu J = L^*U^* - R, \tag{1.6}$$

where R is the residual matrix with a small matrix norm and L^*, U^* are a lower and upper triangular matrix, respectively. These matrices were chosen as proposed in [8]. Let J be a $(K \times K)$ matrix; then writing $A = I - b_0\tau^\nu J$ and denoting the elements of the matrices A, L^*, U^* and R by a_{ij}, l^*_{ij}, u^*_{ij} and r_{ij}, $1 \leqq i, j \leqq K$, the incomplete LU-decomposition is defined by

$$\begin{aligned} &l^*_{jj} = 1, \qquad j = 1, \cdots, K, \\ &\text{If } (k, j) \in P \text{ then } u^*_{kj} = 0 \Rightarrow r_{kj} := -\left(a_{kj} - \sum_{i=1}^{k-1} l^*_{ki} u^*_{ij}\right) \\ &\qquad \text{else } u^*_{kj} := a_{kj} - \sum_{i=1}^{k-1} l^*_{ki} u^*_{ij} \text{ for } j = k, \cdots, K; \\ &\text{If } (j, k) \in P \text{ then } l^*_{jk} = 0 \Rightarrow r_{jk} := -\left(a_{jk} - \sum_{i=1}^{k-1} l^*_{ji} u^*_{ik}\right) \\ &\qquad \text{else } l^*_{jk} := \left(a_{jk} - \sum_{i=1}^{k-1} l^*_{ji} u^*_{ik}\right) / u^*_{kk} \text{ for } j = k+1, \cdots, K, \end{aligned} \tag{1.7}$$

where $k = 1, \cdots, K$. In all our experiments P is the set of pairs of integers defined by

$$P = \{(i, j) \mid |i - j| \neq 0, 1, b-1, b; 1 \leqq i, j \leqq K\},$$

where b is the half-bandwidth of A; i.e., $a_{ij} = 0$ whenever $|i - j| > b$.

This choice of the set P is suitable when the partial differential equation does not contain mixed derivatives and is semidiscretized by standard symmetric differences. For more details on the L^*U^*-decomposition we refer to [8].

We now define the approximations $\tilde{J}_j$, $j = 1, \cdots, M$ in (1.3) by

$$I - b_0\tau^\nu \tilde{J}_j = L^*U^*, \qquad \tilde{J}_j = \frac{1}{b_0\tau^\nu}[I - L^*U^*]. \tag{1.8}$$

Substitution into (1.3) yields for $y^{(j)}$ the expression

$$y^{(j)} = [L^*U^*]^{-1}[Ry^{(j-1)} + \phi(u(j))]. \tag{1.9}$$

This iteration method can be interpreted as a Newton type method in which the linear systems are first preconditioned and then solved by iterative refinement. To see this we consider the linear system to be solved in the modified Newton–Raphson process for (1.2) in the form

$$(I - b_0\tau^\nu J)y = \phi(u(j)), \tag{1.10}$$

where $y^{(u(j))}$ is the solution of the preceding Newton step. Let L^*U^* be the incomplete LU-decomposition defined by (1.7); then (1.10) can be preconditioned to obtain

$$(L^*U^*)^{-1}(I-b_0\tau^\nu J)y=(L^*U^*)^{-1}\phi(u(j)).$$

Substitution of (1.6) yields

$$y=[L^*U^*]^{-1}[Ry+\phi(u(j))] \tag{1.10'}$$

and applying iterative refinement leads to (1.9).

In the second and third class of iteration methods discussed above the update function $u(j)$ is still free. The most simple choice is $u(j)=j-1$, which requires in each iteration an f-evaluation and is therefore rather expensive in general. For the second class of methods based on a splitting of the Jacobian matrix, this strategy was investigated in [4] (method of successive corrections). In this paper we consider more efficient update strategies. Furthermore, in our numerical experiment we will apply the third class of iteration methods.

In § 2 we derive the iteration error of the general iteration method (1.3) and in § 3 the effect is considered of introducing coarse grid corrections into (1.3). Finally, in § 4 we apply (1.9) in a number of parabolic initial-boundary value problems and show that coarse grid corrections improve the accuracy considerably. Also comparisons are given with other integration techniques.

2. The iteration error. Let η be the solution of equation (1.2) and define the iteration error

$$\varepsilon(j)=\eta-y^{(j)}. \tag{2.1}$$

Then it is easily verified that the iteration error of the scheme (1.3) satisfies the relation

(2.2)
$$[I-b_0\tau^\nu\tilde{J}_j]\varepsilon(j)=b_0\tau^\nu\{[J-\tilde{J}_j]\varepsilon(j-1)-(J\eta-f(t_{n+1},\eta))+(Jy^{(u(j))}-f(t_{n+1},y^{(u(j))}))\}.$$

Since f is assumed to be differentiable, it satisfies an inequality of the form

$$\|f(t,v)-Jv-f(t,w)+Jw\|\le\sup_{y\in Y}\left\|\frac{\partial f}{\partial y}(t,y)-J\right\|\|v-w\|, \tag{2.3}$$

where

$$Y=\{y\,|\,y=\Theta v+(1-\Theta)w,\ 0\le\Theta\le 1\}.$$

Using this inequality we derive from (2.2) the estimate

$$\|\varepsilon(j)\|\le|b_0|\tau^\nu\|(I-b_0\tau^\nu\tilde{J}_j)^{-1}\|\{\|J-\tilde{J}_j\|\|\varepsilon(j-1)\|+C_j\|\varepsilon(u(j))\|\},$$

$$C_j=\sup_{y\in Y_j}\left\|\frac{\partial f}{\partial y}(t_{n+1},y)-\frac{\partial f}{\partial y}(t_{n+1},y^{(0)})\right\|, \tag{2.4}$$

$$Y_j=\{y\,|\,y=\Theta\eta+(1-\Theta)y^{(u(j))},\ 0\le\Theta\le 1\}.$$

From (2.4) it is immediate that the final iteration error

$$\varepsilon(M)=O(\tau^{\nu(\hat{p}+1+m)})\quad\text{as }\tau\to 0 \tag{2.5}$$

where $\hat{p}$ is the order of accuracy of the predictor formula used in (1.3) and m denotes the number of f-evaluations. Thus in order to obtain the same order of accuracy p as the generating k-step method (1.2), at least $(p+1)/\nu-\hat{p}-1$ right-hand side evaluations are required.

We also conclude from (2.4) that for slowly varying Jacobian matrices $\partial f/\partial y$ the contribution of the (Newton) error $\varepsilon(u(j))$ will be small when compared with $\varepsilon(j-1)$. This means that one should keep $u(j)$ sufficiently long on a fixed value in order to compensate the possibly large error constant caused by the factor $\|J-\tilde{J}_j\|$. This may lead to large numbers of matrix-vector multiplications but does not increase the number of function evaluations.

3. Coarse grid correction. In order to accelerate the convergence of (1.3) we add to $y^{(j-1)}$ for $j=M_1, M_2, \cdots$ the correction term (cf. [2], [3])

$$c^{(j)}=P_{hH}[I-b_0\tau^\nu J_H]^{-1}R_{Hh}[\phi(u(j-1))-(I-b_0\tau^\nu J)y^{(j-1)}] \tag{3.1}$$

to obtain

$$y^{(j)}=[I-b_0\tau^\nu \tilde{J}_j]^{-1}[b_0\tau^\nu(J-\tilde{J}_j)(y^{(j-1)}+c^{(j)})+\phi(u(j))], \qquad j=M_1, M_2, \cdots. \tag{3.2}$$

Here, J_H denotes the Jacobian matrix of the right-hand side function corresponding to a coarse grid with grid parameter H. The (rectangular) matrices R_{Hh} and P_{hH} relate the grid functions defined on the coarse grid Ω_H and the grid Ω_h actually used $(h<H)\cdot R_{Hh}$ (the *restrictor*) transforms a grid function defined on Ω_h into a function defined on Ω_H, and P_{hH} (the *prolongator*), and vice versa. These operators are assumed to satisfy the relation (cf. [2])

$$\|P_{hH}R_{Hh}-I\|=O(H^q) \quad \text{as } H\to 0, \quad q\geqq 1. \tag{3.3a}$$

Furthermore, it will be assumed that

$$\|J_H-R_{Hh}J_hP_{hH}\|=O(H^q) \quad \text{as } h<H\to 0. \tag{3.3b}$$

The coarse grid corrections

$$c^{(j)}=P_{hH}c_H^{(j)}, \qquad j=M_1, M_2, \cdots, \tag{3.1'}$$

require the solution of linear systems for $c_H^{(j)}$ with

$$A_H=I-b_0\tau^\nu J_H \tag{3.4}$$

as its matrix of coefficients. Let $\tilde{A}_H$ be an approximation to A_H such that $\tilde{A}_H^{-1}$ is easily evaluated, and write the linear system for $c_H^{(j)}$ in the form

$$A_Hz=\psi_H^{(j)}. \tag{3.5}$$

Then we may define the iteration process (cf. [2])

$$z_0=\tilde{A}_H^{-1}\psi_H^{(j)}, \tag{3.6a}$$

$$z_l=[I-\tilde{A}_H^{-1}A_H]z_{l-1}+\tilde{A}_H^{-1}\psi_H^{(j)}, \qquad l=1,2,\cdots. \tag{3.6b}$$

We shall assume that this process solves (3.5) with negligible error. Then the iteration error of (3.2) satisfies the equation (cf. (2.2))

$$\begin{aligned}[I-b_0\tau^\nu\tilde{J}_j]\varepsilon(j)=b_0\tau^\nu\{&[J-\tilde{J}_j][I-P_{hH}A_H^{-1}R_{Hh}A]\varepsilon(j-1)\\ &-[I+b_0\tau^\nu(J-\tilde{J}_j)P_{hH}A_H^{-1}R_{Hh}]\\ &\cdot[(J\eta-f(t_{n+1},\eta))-(Jy^{(u(j))}-f(t_{n+1},y^{(u(j))}))]\}\end{aligned} \tag{3.7}$$

where we have written $A=I-b_0\tau^\nu J$ and where it is assumed that $u(j)=u(j-1)$ for $j=M_1, M_2, \cdots$.

In a similar way as we derived (2.4) from (2.2), we now derive from (3.7) the estimate

$$\begin{aligned}(3.8)\quad \|\varepsilon(j)\| \leqq |b_0|\tau^\nu \|(I-b_0\tau^\nu \tilde{J}_j)^{-1}\| \{ \|J-\tilde{J}_j\| \|I-P_{hH}A_H^{-1}R_{Hh}A\| \|\varepsilon(j-1)\| \\ + C_j \|I+b_0\tau^\nu (J-\tilde{J}_j)P_{hH}A_H^{-1}R_{Hh}\| \|\varepsilon(u(j))\| \},\end{aligned}$$

where C_j is defined in the same way as in (2.4). The main difference with the estimate (2.4) is the occurrence of the factor $\|I-P_{hH}A_H^{-1}R_{Hh}A\|$ in the error constant of $\varepsilon(j-1)$. In order to see the magnitude of this quantity we substitute A and A_H, and write

$$\begin{aligned}I-P_{hH}A_H^{-1}R_{Hh}A \\ =(I-b_0\tau^\nu J^*)^{-1}[(I-b_0\tau^\nu J^*)-(I-b_0\tau^\nu J^*)P_{hH}(I-b_0\tau^\nu J_H)^{-1}R_{Hh}(I-b_0\tau^\nu J)],\end{aligned}$$

where J^* is a (square) matrix satisfying the relation

$$(3.9)\qquad J^*P_{hH}=P_{hH}J_H.$$

It is easily verified that

$$\begin{aligned}(3.10)\quad I-P_{hH}A_H^{-1}R_{Hh}A \\ =(I-b_0\tau^\nu J^*)^{-1}[(I-P_{hH}R_{Hh})-b_0\tau^\nu P_{hH}(J_H-R_{Hh}JP_{hH})R_{Hh}(P_{hH}R_{Hh})^{-1}].\end{aligned}$$

Hence, by virtue of (3.3)

$$(3.11)\qquad \|I-P_{hH}A_H^{-1}R_{Hh}A\|=O(H^q+\tau^\nu H^q)\quad \text{as } \tau, h<H\to 0.$$

From (3.8) we now derive for $j=M_1, M_2, \cdots$

$$(3.12)\qquad \|\varepsilon(j)\| \leqq C\tau^\nu (H^q\|J-\tilde{J}_j\| \|\varepsilon(j-1)\| + C_j\|\varepsilon(u(j))\|),$$

where C is a uniformly bounded constant as τ and $H\to 0$.

4. Numerical experiments.

4.1. The test examples. All initial-boundary value problems chosen for our numerical experiments are defined on $0\leqq t\leqq 1$ and

$$\Omega=\{(x_1,x_2)|0\leqq x_1, x_2\leqq 1\},$$

and semidiscretized on a uniform grid Ω_h with mesh width h by standard symmetric differences. The grid used to define the coarse grid correction (3.1) has grid parameter $H=2h$. Thus, $h=\frac{1}{10}$ results in 81 equations on the fine grid and 16 equations on the coarse grid. For $h=\frac{1}{20}$ we have 361 and 81 equations, respectively.

The examples were chosen such that the exact solution is available. Therefore, initial and boundary conditions can be prescribed by providing the exact solution.

Our first example is linear and serves to test the effect of the coarse grid correction on the rate of convergence of the iteration scheme:

$$\begin{aligned}(4.1)\quad &U_t=\alpha(U_{x_1x_1}+U_{x_2x_2})-\alpha e^{-t}(4\alpha+x_1^2+x_2^2), \qquad \alpha=1, 100,\\ &U(x_1,x_2,t)=\alpha e^{-t}(x_1^2+x_2^2)+1.\end{aligned}$$

Since the exact solution is quadratic in the space variables x_1 and x_2, the space discretization error vanishes so that the time integration aspect can be tested more or less separately from the effects of space discretization.

The second example is defined by [9]

$$(4.2)\qquad U_t=\left(\frac{\partial^2}{\partial x_1^2}+\frac{\partial^2}{\partial x_2^2}\right)U^5, \qquad U(x_1,x_2,t)=[\tfrac{4}{5}(2t+x_1+x_2)]^{1/4}.$$

This nonlinear problem gives rise to an iteration error $\varepsilon(M)$ where both the error of the inner iteration and the outer (Newton) iteration are present (cf. (2.4) and (3.12)). It therefore can be used to demonstrate the effect of the inner and outer iteration processes.

4.2. The numerical scheme. In this paper the numerical experiments are restricted to parabolic equations, i.e. $\nu = 1$ in (1.1). For the implicit formula (1.2) the fourth order backward differentiation formula (cf. e.g. [6, p. 242]) was chosen, which results in

$$(4.3) \qquad b_0 = \tfrac{12}{25}, \qquad \sum_n = \tfrac{1}{25}[48y_n - 36y_{n-1} + 16y_{n-2} - 3y_{n-3}]$$

in the iteration process (1.3). This formula was chosen because of its excellent stability properties [1], so that (1.3) is also expected to be stable if the iteration error is sufficiently small in each integration step.

In order to apply {(1.3), (4.3)}, four starting values are required which were obtained from the exact solution of the initial-boundary value problems. Furthermore, we put $y^{(\text{pred})} = y_n$, $\tau_n = \tau$ is constant, J was obtained by analytical differentiation, $\tilde{J}_j$ is determined according to (1.8) (for details of the L^*U^*-decomposition used we refer to [11]) and the update function $u(j)$ is defined by

$$(4.4) \qquad u(1) = 0, u(j) = \left[\frac{m(j-1)}{M}\right], \qquad j = 2, \cdots, M,$$

where $[x]$ denotes the integer part of x and m is the number of f-evaluations per integration step to be specified in the tables of results.

The scheme {(1.3), (1.8)} was combined with the coarse grid correction (3.2). The values $M_1, M_2, \cdots$ where this coarse grid correction is inserted are given by

$$(4.5) \qquad M_l = l(p+s+1) - s,$$

where p and s are integers to be specified in the tables of results. From (4.5) it follows that two coarse grid corrections are "separated" by $p+s$ iterations and the first correction is preceded by p iterations. In the experiments M is always a multiple of $p+s$, hence the number of coarse grid corrections per integration step is given by

$$(4.6) \qquad r = \frac{M}{p+s}.$$

The performance of the coarse grid correction C itself requires the solution of the linear system (3.5) which is solved by (3.6) in μ iterations including the initial iteration (3.6a). The matrix $\tilde{A}_H^{-1}$ in (3.6) is obtained by incomplete LU-decomposition (cf. (1.8)). The prolongator and restrictor operators needed in the coarse grid correction can be compactly formulated by introduction of the averaging operators μ_-, $\mu_|$, μ_+ and $\mu_\times$. When applied to a grid function at a point Q, these operators are respectively defined by the average of the values at the two "horizontal", the two "vertical", the four "horizontal" and "vertical" and the four "diagonal" neighbouring points of Q. Furthermore we can divide the grid points into four groups according to Fig. 4.1. The coarse grid with a parameter $H = 2h$ consists of grid points denoted by □. Let v be a grid function defined on the coarse grid, i.e. the points □. Then the prolongator is defined by

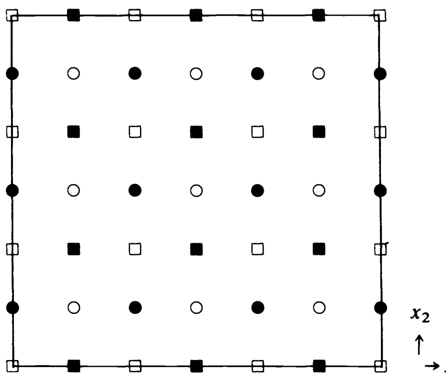


FIG. 4.1. *Grid for* $h=\frac{1}{6}$.

(4.7) $$(P_{hH}v)_\square = v,\quad (P_{hH}v)_\bullet = \mu_| v,\quad (P_{hH}v)_\blacksquare = \mu_- v,\quad (P_{hH}v)_\circ = \mu_\times v.$$

Let u be a grid function defined on the fine grid with grid parameter h. Then

(4.8) $$(R_{Hh}u)_\square = \tfrac{1}{4}u_\square + \tfrac{1}{4}(\mu_\times u)_\square + \tfrac{1}{2}(\mu_+ u)_\square.$$

The numerical scheme specified in this section will be called the PCGC (preconditioning and coarse grid corrections) method.

4.3. Numerical results. In the institute reports [5], [11], a large number of experiments were reported presenting detailed information on the required computational effort for evaluating the function ϕ in (1.3), for performing the iteration steps defined by (1.9) and for computing the coarse grid correction (3.1). Here, we present only a few numerical results in order to demonstrate the behavior of the PCGC method.

In the tables of results we specify the values of

m = number of right-hand side evaluations f per step,

(4.9) M = number of fine grid iterations per step,

r = number of coarse grid corrections per step.

Furthermore, we use the notation:

$A(\tau)$ = accuracy in the end point $t=1$ measured by the minimal number of correct digits, i.e.,

(4.10) $$A(\tau) = -\log_{10}\|y_n - u(t_n)\|_\infty,$$

where $\|\|_\infty$ is the maximum norm and $u(t_n)$ denotes the exact solution of the partial differential equation on the grid Ω_h at $t=t_n$.

$\tilde{p}(\tau, 2\tau)$ = effective order of the scheme {(1.3), (1.8)} in the interval $(\tau, 2\tau)$ defined by

(4.11) $$\tilde{p}(\tau, 2\tau) = \frac{A(\tau)-A(2\tau)}{\log_{10} 2}.$$

In all experiments reported below we chose $p = s$ and performed 4 coarse grid iterations in the calculation of the coarse grid correction (3.1). For those who are interested in the computational effort involved in the experiments we give the expressions for the total number of f-evaluations ($\sum f$), total number of matrix-vector multiplications on Ω_h ($\sum$ matvec), and the total number of equations $L^*U^*y = b$ to be solved on Ω_h ($\sum$ sol):

$$\sum f = (\tau^{-1} - 3)m,$$

$$\sum \text{matvec} = (\tau^{-1} - 3)[m + r(p + s + 1)], \quad \sum \text{sol} = (\tau^{-1} - 3)(p + s)r \qquad \text{for } r \neq 0,$$

$$\sum \text{matvec} = (\tau^{-1} - 3)(m + M), \quad \sum \text{sol} = (\tau^{-1} - 3)M \quad \text{for } r = 0.$$

4.3.1. Coarse grid correction strategy. In order to demonstrate the effect of the number of coarse grid corrections on the accuracy, we first choose the linear problem (4.1) which requires only one f-evaluation per step so that the update function is fixed ($u(j) = 0$). In Tables 4.1 and 4.2 some results are listed showing that inserting coarse grid corrections into the iteration scheme improves the accuracy considerably.

TABLE 4.1
Results for problem (4.1) *with* $\alpha = 1$, $\tau = \frac{1}{4}$ *and* $h = \frac{1}{10}$.

Number of fine grid iterations M	5	10	2	4	6
Number of f-evaluations m	1	1	1	1	1
Number of coarse grid corrections r	0	0	1	2	3
Number of correct digits A	3.20	4.93	2.90	5.26	4.84

TABLE 4.2
Results for problem (4.1) *with* $\alpha = 1$, $\tau = \frac{1}{4}$ *and* $h = \frac{1}{20}$.

Number of fine grid iterations M	5	10	20	2	4	6	8
Number of f-evaluations m	1	1	1	1	1	1	1
Number of coarse grid corrections r	0	0	0	1	2	3	4
Number of correct digits A	1.62	2.27	3.56	2.55	4.46	4.86	4.83

TABLE 4.3
Results for problem (4.1) *with* $\alpha = 100$, $\tau = \frac{1}{4}$ *and* $h = \frac{1}{20}$.

Number of fine grid iterations M	2	8	16	2	6	8	4	12	16
Number of f-evaluations m	1	1	1	1	1	1	1	1	1
Number of coarse grid corrections r	0	0	0	1	3	4	1	3	4
Number of correct digits A	−0.85	−0.27	0.46	0.40	3.13	4.83	0.56	3.71	4.80

In Table 4.3 the results are given for the highly stiff problem (4.1) with $\alpha = 100$. Without coarse grid corrections ($r = 0$) the convergence is extremely slow, whereas a minimum number of iterations with only a single coarse grid correction is sufficient to obtain some accuracy. Additional experiments have shown that the accuracy gradually increases if the number of coarse grid corrections increases from 1 until 4, and remains constant for larger values of $r(A(\frac{1}{4}) \sim 4.7)$.

TABLE 4.4
Number of correct digits A obtained without coarse grid corrections ($r = 0$) *for problem* (4.2) *with* $\tau = \frac{1}{4}$ *and* $h = \frac{1}{20}$.

M	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$
4	1.47				
6	1.68				
8	1.93	1.96			
10	1.96	2.26			
12	1.95	2.44	2.48		
16			2.95	3.00	
20				3.48	3.55
24	⋮	⋮	⋮		3.90
30				⋮	⋮
40	1.95	2.44	2.95	3.48	3.90

In Tables 4.4 and 4.5 the effect of the number of f-evaluations m and coarse grid corrections is illustrated for the nonlinear problem (4.2). If no coarse grid corrections are inserted then we see from Table 4.4 that roughly $M = 4 + 4m$ iterations are required in order to reduce the iteration error in the solution of the linear systems to a negligible value. The method with $r > 0$, however, only needs $M = 2m$ iterations to achieve the same result (see Table 4.5). Notice that this is also the lowest possible number of iterations because $r \geqq m$ and $M = 2r$.

TABLE 4.5
Number of correct digits A obtained with coarse grid corrections for problem (4.2) *with* $\tau = \frac{1}{4}$ *and* $h = \frac{1}{20}$.

$r = M/2$	$m = 1$	$m = 2$	$m = 3$	$m = 4$	$m = 5$
1	1.96				
2	1.95	2.44			
3			2.95		
4				3.48	
5	⋮	⋮	⋮		3.90
6				⋮	⋮
8	1.95	2.44	2.95	3.46	3.90

4.3.2. The effective order of the iteration scheme. From experiments with the method of successive corrections {(1.3), (1.5)} reported in [4], it follows that often the order of accuracy is considerably less than the asymptotic order of accuracy, particularly for small values of m and large values of $\tau\sigma$ where σ denotes the spectral radius of the Jacobian matrix J. Therefore, we are interested in the effective order (4.11) of the scheme {(1.3), (1.8)}.

In Table 4.6 the effective orders are given for problem (4.2) together with the asymptotic order p derived from (2.5). The integration steps are performed without coarse grid corrections. Each three iterations the function f is updated, hence $M = 3m$. For $m > 3$ and $\tau \geqq \frac{1}{20}$ the space discretization error becomes dominant in the error $y_n - u(t_n)$ so that (4.11) does not give the order of the time discretization error (indicated by —). The results in this table indicate that the asymptotic order p is not reached. A possible explanation might be the effect of the space discretization error or the still relatively large values of τ. Since we cannot decrease the value of τ (as the space discretization error would become dominant), we decrease the value of h.

TABLE 4.6
Effective orders $\tilde{p}$ obtained without coarse grid corrections for problem (4.2) *with $h=\frac{1}{10}$ and $M=3m$.*

τ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$
1/5					
	0.6	3.6	4.6	5.9	5.8
1/10					
	0.9	1.3	2.3	—	—
1/20					
	1.3	1.2	1.6	—	—
1/40					
$\tau\to 0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=4$

TABLE 4.7
Effective orders $\tilde{p}$ obtained by m coarse grid corrections for problem (4.2) *with $h=\frac{1}{20}$ and $M=2m$.*

τ	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$
1/5					
	1.8	3.7	4.5	5.9	6.7
1/10					
	2.0	3.3	4.3	4.8	—
1/20					
	1.0	2.9	4.0	—	—
1/40					
$\tau\to 0$	$p=1$	$p=2$	$p=3$	$p=4$	$p=4$

In Table 4.7 the results are listed obtained for $h=\frac{1}{20}$ and by inserting $r=m$ coarse grid corrections. Although the asymptotic order is still not shown, we see a convergence to the correct values for decreasing τ-values. It is also evident from these results that the introduction of coarse grid corrections increases the effective order considerably.

4.4. Comparison with other integration methods. The PCGC method has been compared with two other integration methods. The first one is the second order one-step Runge–Kutta–Chebyshev method (RKC method) described in [10]. The second method is also based on the preconditioned linear equation (1.10′), but instead of accelerating the process of iterative refinement by coarse grid corrections as is done in the PCGC method, the convergence is accelerated by applying Chebyshev iteration (Richardson's method). By virtue of the property that the matrix $I-(L^*U^*)^{-1}R$ has its eigenvalues in the right half-plane (provided that certain mild conditions are satisfied [8]), Manteuffel's analysis of Richardson's method can be applied [7] and the optimal values of the iteration parameters can be evaluated. The generating method is identical to that of the PCGC method. This method will be called the preconditioned Richardson method (PR method); the number of f-evaluations is denoted by m, the number of iterations for solving the linear systems is denoted by q.

Our test examples are again problems (4.1) and (4.2), both with $h=\frac{1}{20}$.

In order to compare the three methods the $A(\tau)$-values and the computational effort required are listed in one table. The computational effort is measured by the number N of computational units which are defined differently for each method. For *nonlinear* problems one may choose

$$\begin{array}{lll} & \text{PCGC } (r=m): & f+4\text{ matvec}+2\text{ sol}+C_4+\dfrac{1}{m}[(L^*U^*)_h+(L^*U^*)_H], \\ (4.12) & \text{RKC:} & 10f, \\ & \text{PR:} & \frac{1}{2}\left[f+(q+1)\text{ matvec}+\dfrac{1}{m}(L^*U^*)_h+q\text{ sol}\right]. \end{array}$$

Here, L^*U^* denotes the computational effort to perform the incomplete LU-decomposition on Ω_h and Ω_H. For the definition of the other quantities we refer to the preceding subsections.

For *linear* problems one may choose the computational units

$$\begin{array}{lll} & \text{PCGC:} & \dfrac{1}{r}[f+\text{matvec}]+C_4+3\text{ matvec}+2\text{ sol}, \\ (4.13) & \text{RKC:} & 4f, \\ & \text{PR:} & \frac{1}{5}[f+\text{matvec}+q(\text{matvec}+\text{sol})]. \end{array}$$

Notice that the computational work involved to perform the incomplete LU-decompositions is neglected in these units because for linear problems these calculations are required only once and the decompositions can be used in all integration steps.

In (4.12) and (4.13) not all calculations performed by the various methods are taken into account. In the PCGC method the evaluations of the Jacobian matrices are neglected and are in fact provided in closed form in our experiments; in the RKC method the evaluation of the spectral radius of the Jacobian matrix is neglected and in the PR method all initial work for estimating the iteration parameters and the evaluation of the Jacobian matrices as well are not taken into account. For a more detailed discussion of the computational units (4.12) and (4.13) and for a comparison on the basis of arithmetic operations we refer to [11].

Another important aspect in interpreting the results obtained by the three methods is the storage requirement. The RKC method requires only a few vector arrays whereas especially the PCGC method needs considerably more storage.

In Fig. 4.2 the $A(\tau)$ values and the corresponding computational work N (expressed in terms of the units defined in (4.13)) are illustrated. These values were obtained by performing the integration with a number of integration steps (RKC with $\tau=1, \frac{1}{12}, \frac{1}{35}$ and $\frac{1}{70}$, the other methods with $\tau=\frac{1}{5}, \frac{1}{10}$ and $\frac{1}{20}$.

In Fig. 4.3 the $A(\tau)$ and N values for problem (4.2) are illustrated, as obtained by the RKC, the PR and the PCGC method. The results of the RKC method correspond to $\tau=1, \frac{1}{2}, \frac{1}{5}, \frac{1}{10}, \frac{1}{20}, \frac{1}{40}$ and $\frac{1}{80}$. For the PR and PCGC methods the integration step τ and the value of m are indicated in the plots.

From Figs. 4.2 and 4.3 we conclude that *relative to the units* (4.12) and (4.13) the PCGC method is the most efficient one and the RKC method the most expensive one. However, are the units (4.12) and (4.13) comparable? This is both problem- and computer-dependent, so that we shall not try to answer the question. Moreover, the aspect of storage may be as important as the computational effort which places the RKC at the first place.

Finally, we remark that the PCGC method analysed in this paper should be implemented as a full multigrid method as described in [2] and [3] if one decides to base a software package on preconditioning and coarse grid corrections. Also a more suitable predictor formula $y^{(\text{pred})}$ might be considered.

Acknowledgment. The authors are grateful to Mr. J. Kok for programming the incomplete LU-decomposition.

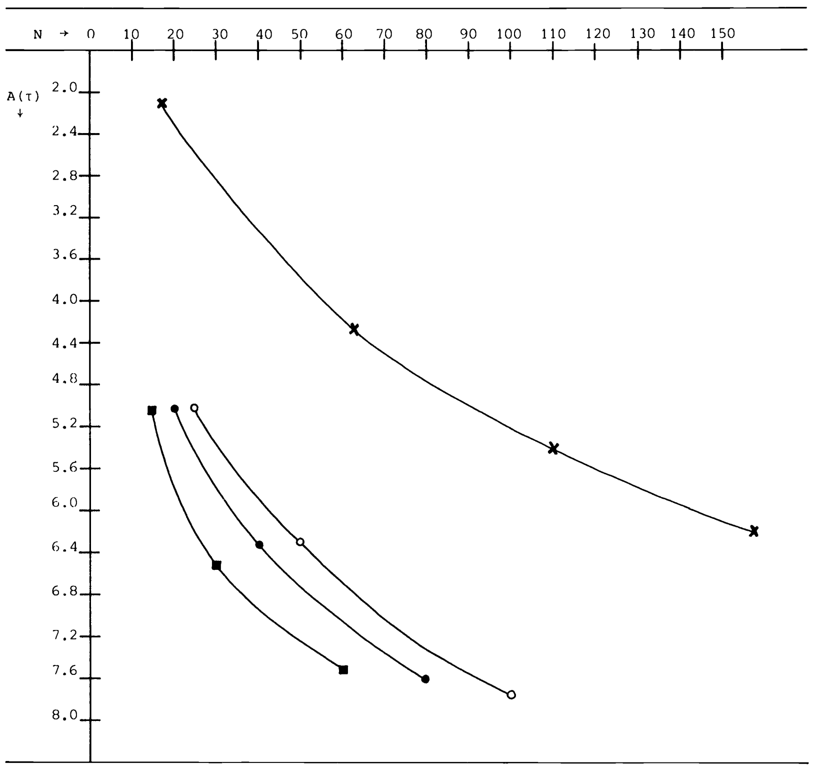


FIG. 4.2. *Number of correct digits A and the computational effort N for problem* (4.1) *with* $h = \frac{1}{20}$. × RKC; ■ PCGC *with* $M = 2r = 6$ *and* $m = 1$; ● PCGC *with* $M = 2r = 8$ *and* $m = 1$; ○ PR *method with* $q = 14$ *and* $m = 1$.

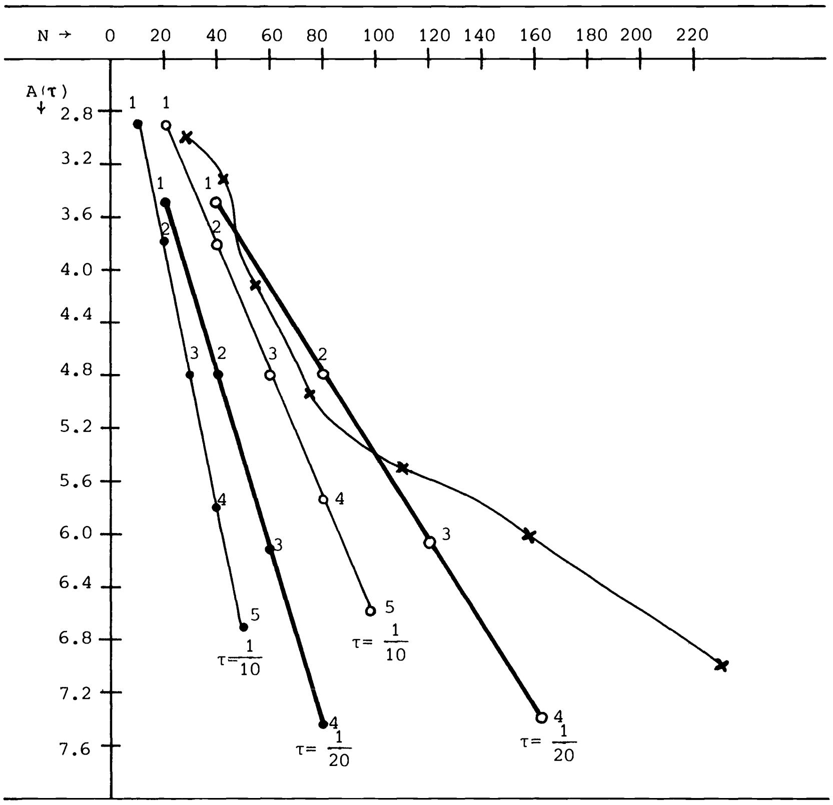


FIG. 4.3. *Number of correct digits A and computational effort N for problem* (4.2) *with* $h=\frac{1}{20}$. × RKC; ● PCGC *with* $M=2m=2r$; ○ PR *with* $q=9$. *The value of m is indicated in the plots.*

REFERENCES

[1] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
[2] P. W. HEMKER, *Introduction to multi-grid methods*, Nieuw Arch. Wisk. (3), XXIX (1981), pp. 71–101.
[3] ———, *On the structure of an adaptive multi-level algorithm*, Nordisk. Tidskr. Informationsbehandling (BIT), 20 (1980), pp. 289–301.
[4] P. J. VAN DER HOUWEN, *Multistep splitting methods of high order for initial value problems*, SIAM J. Numer. Anal., 17 (1980), pp. 410–427.
[5] P. J. VAN DER HOUWEN AND H. B. DE VRIES, *Preconditioning and coarse grid corrections in the solution of the initial value problem for nonlinear partial differential equations*, Report NW 95, Mathematisch Centrum, Amsterdam, 1980.
[6] J. D. LAMBERT, *Computational Methods in Ordinary Differential Equations*, John Wiley, London, 1973.
[7] T. A. MANTEUFFEL, *The Tchebychev iteration for non-symmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327.
[8] 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.
[9] R. D. RICHTMYER AND K. W. MORTON, *Difference Methods for Initial Value Problems*, Interscience, New York, 1967.
[10] B. P. SOMMEIJER AND P. J. VAN DER HOUWEN, *On the economization of stabilized Runge-Kutta methods with applications to parabolic initial value problems*, Z. Angew. Math. Mech., 61 (1981), pp. 105–114.
[11] H. B. DE VRIES, *The two-level algorithm in the solution of the initial value problem for partial differential equations*, Report NW, Mathematisch Centrum, Amsterdam, in preparation.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

COMPARISON OF TWO ALGORITHMS FOR SOLVING LARGE LINEAR SYSTEMS*

ZAHARI ZLATEV,† JERZY WASNIEWSKI‡ AND KJELD SCHAUMBURG§

Abstract. Assume that the coefficient matrix A in the system $Ax = b$ is large and sparse. Consider the following two algorithms: DS (where the system is solved by a direct use of Gaussian elimination) and IR (where the use of Gaussian elimination is combined with the use of a large drop tolerance and followed by iterative refinement). Assume that some sparse technique is implemented with both DS and IR. The performance of two codes, the NAG subroutines (which are based on DS) and the RECKU subroutines (which are based on IR) are compared on a wide set of test matrices. The comparison shows that the second algorithm, IR, performs better in general. The computing time and/or the storage needed may be reduced considerably when the IR algorithm is used. Moreover, this algorithm normally provides a reliable estimate of the accuracy of the computed solution. When the problems are time and storage consuming, IR is much better (it gives a reduction in the computing time of up to 10 times and a reduction in the storage of up to 2–3 times). It is shown that IR is very efficient when linear least-squares problems are solved by the use of augmented matrices.

Key words. Sparse matrices, direct solution, iterative refinement, drop-tolerance, accuracy requirements, nonzero elements, fill-ins, pivotal strategy, test-matrices, matrix generators, least-squares problems, augmented matrices, storage schemes for sparse matrices.

1. Introduction.

Consider the problem

$$Ax = b, \quad n \in \mathbb{N}, \quad A \in \mathbb{R}^{n\times n}, \quad b \in \mathbb{R}^{n\times 1}, \quad \operatorname{rank}(A) = n. \tag{1.1}$$

Let $x \in \mathbb{R}^{n\times 1}$ be the exact solution of (1.1). An approximate solution, $\bar{x} \in \mathbb{R}^{n\times 1}$, of (1.1) can be found by the following algorithm:

(i) Use the Gaussian elimination to decompose matrix A into two triangular matrices L and U so that

$$LU = PAQ + E \tag{1.2}$$

where P and Q are permutation matrices, E is a perturbation matrix.

(ii) Calculate a first approximation, x_1, by

$$x_1 = Hb, \qquad H = QU^{-1}L^{-1}P. \tag{1.3}$$

(iii) Attempt to improve the accuracy of x_1 by iterative refinement:

$$r_i = b - Ax_i, \quad d_i = Hr_i, \quad x_{i+1} = x_i + d_i \qquad (i = 1(1)p-1). \tag{1.4}$$

The third step is optional. If it is not carried out, then we set $\bar{x} = x_1$ and will call the algorithm DS (direct solution). If all three steps are applied, then the algorithm will be called IR (iterative refinement). In this case $\bar{x} = x_p$ is normally more accurate than x_1. Moreover, $\|d_{p-1}\|_\infty$ is a reliable estimate for $\|x - \bar{x}\|_\infty$.

Consider the case where A is dense. Then the use of IR is connected with requirements for both extra storage (about 100%; a copy of matrix A should be

* Received by the editors January 29, 1980, and in revised form January 19, 1981.

† Institute for Numerical Analysis, The Technical University of Denmark, DK-2800 Lyngby, Denmark. Present affiliation: National Agency of Environmental Protection, Air Pollution Laboratory, DK-4000 Roskilde, Denmark.

‡ The Regional Computing Center at the University of Copenhagen, Vermundsgade 5, DK-2100 Copenhagen Ø, Denmark.

§ Department of Chemical Physics, University of Copenhagen, The H. C. Ørsted Institute, Universitetsparken 5, DK-2100 Copenhagen, Denmark.

made) and extra computing time (for the calculations in (1.4)). Therefore the third step of the above algorithm is usually not carried out when A is dense. In this situation double precision is often recommended in order to meet the accuracy requirements. Moreover, a cheap estimate of the condition number of matrix A can be calculated (without using any copy of matrix A) and used as a measure of sensitivity of the results to errors in the computations (see Cline et al. [7] and Forsythe et al. [15]).

Assume now that matrix A is large and sparse. Also in this case the use of IR will give both an approximation $\bar{x} = x_p$ which is normally more accurate than x_1 and a reliable error estimate. Moreover, *if the sparsity of A is exploited by some sparse matrix technique* (SMT), *then it is possible to develop an algorithm where the use of* IR *will often lead to a reduction in storage and/or computing time* (compared with the storage and the computing time needed when DS is applied). Such an algorithm has been implemented in package Y12M (Zlatev [31], Zlatev et al. [34], Zlatev and Wasniewski [36] and Zlatev et al. [37]). The SMT implemented in Y12M will be sketched in the next section. It is necessary to emphasize here that, roughly speaking, by the application of any SMT, one attempts to store and use in the computations only the nonzero elements of A (see Brayton et al. [5], Curtis and Reid [8], Duff [9], Reid [23] and Tewarson [26]). In the algorithm implemented in Y12M we go further: We keep and use in the computations only the elements of A which are larger (in absolute value) than a special parameter T, the drop tolerance (Clasen [6], Reid [22], Tewarson [26] and Wolfe [30]). This means that any element, $a_{ij}^{(s+1)}$, is removed (and thus not used in the further computations) when

$$|a_{ij}^{(s+1)}| < T, \tag{1.5}$$

where $s = 1(1)n-1$ and

$$a_{ij}^{(s+1)} = a_{ij}^{(s)} - a_{is}^{(s)} a_{sj}^{(s)} / a_{ss}^{(s)}, \qquad a_{ss}^{(s)} \neq 0, \quad a_{ij}^{(1)} = a_{ij} \in A. \tag{1.6}$$

It is obvious that the use of a large drop tolerance (LDT) may lead to a reduction of both the storage and the computing time. However, the approximation x_1 so found will often be inaccurate. Therefore LDT should not be used with DS. This leads to the necessity of using IR when T is large.

In Y12M both the use of $T > 0$ and the use of IR are optional. *If* IR *is applied*, then the components of the residual vector r are accumulated in double precision and then rounded to single precision (see Björck [4], Stewart [25], Wilkinson [27], [28] and Wilkinson and Reinsch [29]). The IR process is terminated when any of the following stopping criteria is satisfied:

$$\|d_{i-1}\|_\infty < \varepsilon \|x_i\| \qquad (\varepsilon \text{ is the machine accuracy}), \tag{1.7}$$

$$\|d_i\|_\infty > \|d_{i-1}\|_\infty \wedge i > 2, \tag{1.8}$$

$$i = IMAX \qquad (IMAX \text{ is prescribed in advance}). \tag{1.9}$$

The use of IR *should be combined with a careful choice of* T. Some practical rules for choosing T are given below.

Assume that: (i) matrix A is not very ill-conditioned, and (ii) all numbers

$$a_i = \max_{1 \leq j \leq n} (|a_{ij}|), \qquad i = 1(1)n, \quad a_{ij} \in A, \tag{1.10}$$

have the same magnitude. Denote

$$a = \min_{1 \leq i \leq n} (a_i). \tag{1.11}$$

Then $T \in [10^{-5}a, 10^{-3}a]$ will normally be a good choice for T. Note that even if some very crude estimate of the magnitude of a is known, then this estimate can be used instead of a. If no information about the magnitude of the nonzero elements of A is available, then row scaling of matrix A can be carried out before the beginning of the decomposition (1.2). The cost of this process is $O(NZ)$. This is 3–5 times cheaper than the cost of one iteration in (1.4) and is negligible compared to the total computational time needed for IR. We hope that the user will have some information about the magnitude of the nonzero elements of matrix A and, therefore, will be able to use the above rules. If this is not the case, then a special option in Y12M can be used. In this option, one can specify some T in the interval $[-10^{-3}, -10^{-5}]$. The code Y12M will perform row scaling automatically and use in the computations a drop tolerance $T_1 = -T$. Note that row scaling has no essential effect on the numerical solution (Forsythe and Moler [16]). It is carried out only to facilitate the choice of T. It should also be mentioned that the algorithm advocated in Duff et al. [13] is implemented in our package Y12M. An additional rule, which can be successfully applied in the solution of linear systems of ordinary differential equations by implicit time-discretization schemes, is described in Schaumburg et al. [24].

If T is chosen according to any of the above rules, then we shall call it a *large drop tolerance* (LDT). Thus, the value of the LDT is related to the magnitude of the nonzero elements of matrix A.

We shall refer to the option in Y12M where LDT and IR are used as RECKU subroutines in this paper. This means that, unless the opposite is emphasized in the text, *the use of* LDT *and* IR *is always assumed in the* RECKU *subroutines.*

Our purpose is to show that the use of LDT and IR is *often* more efficient than the simple use of DS. The numerical results in Zlatev [31], Zlatev et al. [34], [37] show that usually the Y12M option with LDT + IR (the RECKU subroutines) performs better than the Y12M option with DS only. The same conclusion can be drawn from Schaumburg et al. [24], where some large chemical problems are solved. In this paper the performance of the RECKU subroutines is compared with a good code oriented to the users. The subroutines F01BRE and F04AXE from the NAG Library[1] have been chosen as such a code. We shall refer to these subroutines as NAG *subroutines.* The NAG subroutines have been chosen because (i) these subroutines perform best among the subroutines available at RECKU[2] (where the NAG Library is implemented), (ii) the basic ideas used in these subroutines are well described (Duff [10] and Duff and Reid [12]). It is very important (but this was not decisive for our choice) that the RECKU subroutines and the NAG subroutines are based on similar principles (e.g., the ideas proposed in Gustavson [18], [19] are applied in the storage scheme in both packages; see § 2). This makes the comparison and the discussion of the results easier. However, it must be emphasized here that the results depend not only on the fact that LDT + IR is used or not used. The influence of some other factors (e.g., the pivoting strategy) can also be traced, sometimes. Unfortunately, such influence cannot be avoided when two different codes are used. Nevertheless, we believe that the main reason for the difference in the results is the use of LDT and IR in the RECKU subroutines. This is important because our purpose is not to compare the two codes, but to show that an algorithm based on SMT + LDT + IR is often more efficient than a good user oriented algorithm based on SMT and DS.

[1] NAG: Numerical Algorithms Group, Banbury Road 7, Oxford, England.

[2] RECKU: The Regional Computing Center at the University of Copenhagen.

2. On the storage schemes and the pivotal strategies used. The storage schemes used in both packages are very similar. They are based on ideas proposed in Gustavson [18], [19]. A brief presentation of the storage schemes is given below; for more details about the RECKU subroutines see Zlatev [31], Zlatev et al. [34] and Zlatev and Wasniewski [36]; about the NAG subroutines, Duff [10] and Duff and Reid [12]. Only the facts needed to better explain the numerical results are described in this section.

Denote by NZ the number of nonzero elements in A. The main arrays (which have length larger than NZ) are three in both packages: real array A and integer arrays SNR and RNR (here and below the notation used in the RECKU subroutines is given). On entry the nonzero elements of matrix A must be stored (in an arbitrary order) in the first NZ locations of array A so that if $A(K)=a_{ij}$ $(K=1(1)NZ)$ then $RNR(K)=i$ and $SNR(K)=j$. The RECKU subroutines order the elements by rows and store them in the first NZ locations of arrays A and SNR so that if $A(K)=a_{ij}$ $(K=1(1)NZ)$, then $SNR(K)=j$. A and SNR form the row ordered list. The row numbers of the nonzero elements ordered by columns are stored in the first NZ locations of array RNR. This array forms the column ordered list. The nonzero elements are ordered in a slightly different way within the lists by the NAG subroutines; however, the basic ideas are the same. Some additional information (e.g., about the row starts and the row ends) is also needed and is stored in an integer array HA (with length $13n$; two arrays are used in the NAG subroutines but their total length is also $13n$). After stage s, $s=1(1)n-1$, of the elimination the row numbers of the nonzero elements in column s are not needed and are therefore removed from array RNR. In this way the length of array RNR may be smaller than the length of the arrays in the row ordered list (about 40%, even if the matrix is such that the number of fill-ins is large; the new nonzero element created when $a_{ij}^{(s)}=0$ in (1.6) while neither $a_{is}^{(s)}=0$ nor $a_{sj}^{(s)}=0$ is called fill-in). The possibility of using a smaller length for array RNR has been proposed in Zlatev and Thomsen [35] and Zlatev and Barker [33], and after that was also used in Duff [10], Duff and Reid [12], Munksgaard [21] and Zlatev and Wasniewski [36]. It is difficult to give recommendations about the length of arrays A and SNR. If IR+LDT is used, then this length is normally in the interval $[2NZ, 3NZ]$.

In the RECKU subroutines, two extra large arrays are needed (a real array $A1$ and an integer array SN; both of length NZ). A copy of the nonzero elements of matrix A (ordered by rows) is made in array $A1$ so that if $A1(K)=a_{ij}$ $(K=1(1)NZ)$, then $SN(K)=j$. Some additional storage (of length $4n$) is also needed when the RECKU subroutines are used (e.g., to store the residual vector r_i in the ith iteration; see Zlatev et al. [34]).

Let COUNT (RECKU) be the largest number of nonzero elements kept in array A during any stage of the elimination process (1.2) when the RECKU subroutines are used. Denote the corresponding number for the NAG subroutines by COUNT (NAG). Then the relation

$$\text{COUNT (RECKU)}+NZ<\text{COUNT (NAG)} \tag{2.1}$$

will indicate that it is possible to use less storage with the RECKU subroutines. The parameter COUNT (without any indication when the meaning is clear) will be used in the storage comparisons.

The pivotal strategies used in both packages can be found as special cases in the GMS (generalized Markowitz strategy) described in Zlatev [32]. However, these strategies are different. Moreover, the influence of the pivotal strategy on the results is sometimes apparent. Therefore, a brief discussion of the pivotal strategies is needed before the presentation of the numerical results.

Assume that the sth pivotal element has to be found ($s = 1(1)n-1$). Consider

(2.2) $$A_s = \{a_{ij}^{(s)}/s \leqq i \leqq n, s \leqq j \leqq n\},$$

(2.3) $$I_s = \{i_m/m = 1(1)p(s), 1 \leqq p(s) \leqq n-s+1, s \leqq i_m \leqq n\},$$

(2.4) $$(i_k \in I_s) \wedge (i_q \in I_s) \wedge (k < q) \Rightarrow r(i_k, s) \leqq r(i_q, s),$$

(2.5) $$(i \notin I_s) \wedge (s \leqq i \leqq n) \Rightarrow r(i_{p(s)}, s) \leqq r(i, s),$$

(2.6) $$B_s = \{a_{ij}^{(s)} \in A_s/|a_{ij}^{(s)}| \cdot u \geqq \max_{s \leqq k \leqq n} (|a_{ik}^{(s)}|), i \in I_s, u \geqq 1\},$$

(2.7) $$M_{ijs} = [r(i,s)-1][c(j,s)-1],$$

(2.8) $$M_s = \min_{a_{ij}^{(s)} \in B_s} (M_{ijs}),$$

(2.9) $$C_s = \{a_{ij}^{(s)} \in B_s/M_{ijs} = M_s\},$$

where (i) $r(i,s)$ and $c(j,s)$ are the numbers of the nonzero elements in row i and column j which belong to A_s (A_s is called the active part of matrix A at stage s), (ii) u is called a stability factor and B_s is called a stability set, (iii) M_{ijs} is called the Markowitz cost of element $a_{ij}^{(s)}$, and (iv) the elements of set C_s are candidates for pivotal elements at stage s of Gaussian elimination.

The pivotal strategies defined by (2.2)–(2.9) depend on two parameters: u and $p(s)$. In the original Markowitz strategy (Markowitz [20]), $u = \infty$ and $p(s) = n-s+1$ are used. In the NAG subroutines $u = 10$ is recommended and $p(s) = n-s+1$ is used (Duff [10] and Duff and Reid [12]). Moreover, any element of C_s can be chosen as a pivotal element in the NAG subroutines. Zlatev [32] proposed choosing as pivotal the element of C_s with the largest absolute value. This strategy is called an IGMS (improved generalized Markowitz strategy) in Zlatev [32] and is implemented in the RECKU subroutines (with recommendations $u \in [4, 16]$ and $p(s) \leqq 3$). Since the elements of C_s are also elements of the stability set B_s, the change made to obtain an IGMS does not seem to be very important for the accuracy of the results. However, it can be verified both theoretically and experimentally that this is not so. In Zlatev [32] it is proved that there exist classes of matrices for which any IGMS will ensure stable results, while the GMSs *may* cause instability. In Zlatev [31], [32] there are some examples given where the codes based on GMSs produce much poorer results than the codes based on IGMSs (see also the results in Table 3 of this paper).

3. Test matrices. Three matrix generators have been used in our experiments. These generators are described in Zlatev et al. [38]. Each generator can produce arbitrarily many matrices depending on some or all of the following parameters: m, n, c, r, α. The numbers of rows and columns in the desired matrices can be changed by m and n respectively (we shall mainly use $m = n$). The positions of certain nonzero elements in A are determined by the choice of c. The number of nonzero elements can be changed by r (so that $NZ = rm + 110$). The magnitude of the nonzero elements can be varied by α so that $\max_{1 \leqq i,j \leqq n}(|a_{ij}|)/\min_{1 \leqq i,j \leqq n}(|a_{ij}|) = 10\alpha^2$, $a_{ij} \in A$, $a_{ij} \neq 0$. Only the parameters n and c can be varied in the first two matrix generators, the subroutines MATRD and MATRE. These subroutines generate square matrices which are called matrices of class $D(n,c)$ and class $E(n,c)$, respectively. Two illustrations for matrices of these classes are given in Figs. 1 and 2; more details can be found in Zlatev [31], [32], Zlatev et al. [34], Zlatev and Wasniewski [36] and Zlatev et al. [38].

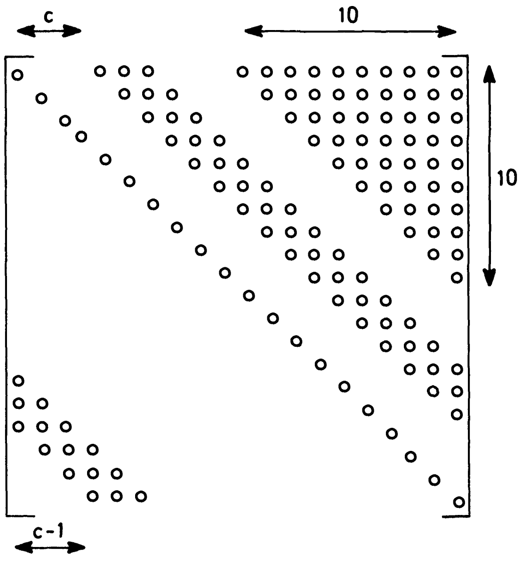


FIG. 1. *Sparsity pattern of matrix $A = D(20, 4)$. The nonzero elements are indicated by* o.

$$\xleftrightarrow{c}$$

$$\begin{bmatrix}
4 & -1 & & & -1 & & & & & \\
-1 & 4 & -1 & & & -1 & & & & \\
& -1 & 4 & -1 & & & -1 & & & \\
& & -1 & 4 & -1 & & & -1 & & \\
-1 & & & -1 & 4 & -1 & & & -1 & \\
& -1 & & & -1 & 4 & -1 & & & -1 \\
& & -1 & & & -1 & 4 & -1 & & \\
& & & -1 & & & -1 & 4 & -1 & \\
& & & & -1 & & & -1 & 4 & -1 \\
& & & & & -1 & & & -1 & 4
\end{bmatrix}$$

FIG. 2. *Matrix* $A = E(10, 4)$.

All parameters can be changed in the third matrix generator, subroutine MATRF2, which produces matrices of class $F2(m, n, c, r, \alpha)$ (see Zlatev et al. [38]). The sparsity pattern of a matrix of this class is given in Fig. 3.

All Harwell test matrices (36 in number; see Duff [10], Duff and Reid [11], [12]) have also been tested.

4. Numerical experiments. All numerical experiments have been carried out at RECKU (the Regional Computing Center at the University of Copenhagen) on a UNIVAC 1100/82. The FTN compiler and single precision ($\varepsilon \approx 1.49 \times 10^{-8}$) have been used. The computing time is always given in seconds. The right-hand side vector has always been generated so that all components of the exact solution, x, are equal to 1. $T = 10^{-2}$, $u = 4$ and $p(s) = \min\{n - s + 1, 3\}$ were normally used with the RECKU subroutines, while the recommended value of parameter u ($u = 10$) was chosen for the NAG subroutines. Whenever another value of some of these parameters was applied, this value is explicitly given in the corresponding table.

4.1. Solving systems with different numbers of equations. Test matrices of classes $D(n, c)$ and $E(n, c)$ have been used in order to investigate the performance of the two codes in the situation where n is varied ($n = 650(50)1000$; moreover, for each n

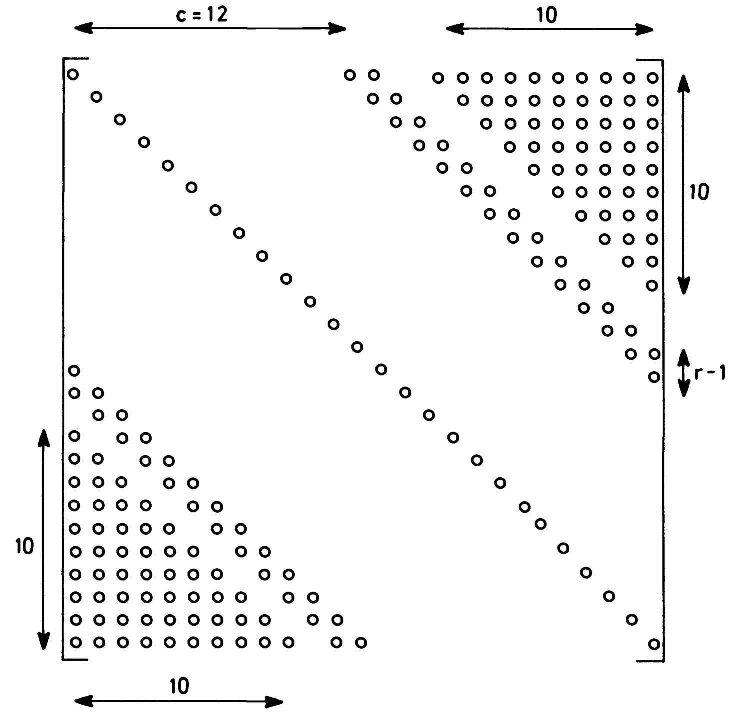


FIG. 3. *Sparsity pattern of matrix* $A = F2(26, 26, 12, 3, \alpha)$. *The nonzero elements are indicated by* o.

six systems with $c = 4(40)204$ are solved). *A tendency for the efficiency* (with regard to the computing time used) *to increase when n becomes large* has been observed (see Table 1). For many examples, COUNT (RECKU) < COUNT (NAG)/3 has been found (see Tables 2 and 3). The accuracy of the approximations found by the RECKU subroutines is close to the machine accuracy for all 96 systems. The accuracy of the approximations found by the NAG subroutines is normally $O(10^{-4})$ for matrices of class $D(n, c)$ and varies from $O(10^{-1})$ to $O(10^{-6})$ for matrices of class $E(n, c)$.

TABLE 1
For each n, the sum of the computing times (in seconds), found in the solution of 6 *systems with different values of c* ($c = 4(40)204$), *is given.*

	Matrices of class $D(n, c)$		Matrices of class $E(n, c)$	
n	NAG subroutines	RECKU subroutines	NAG subroutines	RECKU subroutines
650	46.52	13.16 (28.3%)	51.55	16.34 (31.9%)
700	58.63	13.00 (22.2%)	71.33	19.10 (26.8%)
750	56.38	13.77 (31.5%)	85.45	21.55 (25.2%)
800	64.50	14.55 (22.6%)	107.31	23.06 (21.5%)
850	67.74	15.16 (22.4%)	127.42	25.33 (19.9%)
900	75.89	16.00 (21.1%)	137.55	27.45 (20.0%)
950	88.99	16.28 (18.3%)	224.30	29.51 (13.2%)
1000	80.85	17.85 (22.1%)	251.55	31.79 (12.6%)
Total	559.50	119.77 (22.2%)	1056.46	192.22 (18.4%)

4.2. Solving systems whose matrices have different sparsity patterns. The experiment described in § 4.1 has been considered for $n = 800$. The results are given in Tables 2 and 3. It is seen that both the storage parameter COUNT and the computing time vary very much with c when the NAG subroutines are used.

TABLE 2
Comparison of some characteristics obtained for matrices $D(800, c)$, $c = 4(40)204$, $NZ = 4n + 55 = 3255$ for all values of c.

	NAG subroutines		RECKU subroutines	
c	Time	COUNT	Time	COUNT
4	3.41	8431	2.37 (69.5%)	4573 (54.2%)
44	11.61	17668	2.09 (18.0%)	5843 (33.1%)
84	13.86	20340	2.43 (17.6%)	5979 (29.4%)
124	13.98	18416	2.44 (17.5%)	5978 (32.5%)
164	12.63	18497	2.75 (21.8%)	5921 (32.0%)
204	9.01	16257	2.47 (27.4%)	5937 (36.4%)
Average results	10.75	16608.2	2.42 (22.6%)	5705.2 (34.4%)

TABLE 3
Comparison of some characteristics obtained for matrices $E(800, c)$, $c = 4(40)204$, $NZ = 5n - 20 - 2$.

	NAG subroutines			RECKU subroutines		
c	Time	COUNT	Accuracy	Time	COUNT	Accuracy
4	2.71	9420	5.49 E-4	2.45 (90.4%)	6504 (69.0%)	0.0
44	53.67	30424	4.13 E-1	6.88 (12.8%)	9882 (32.5%)	5.96E-8
84	24.73	22868	1.31 E-2	4.43 (17.9%)	8793 (38.5%)	1.12 E-6
124	12.45	16778	2.36 E-3	3.67 (29.5%)	7849 (46.8%)	7.45 E-8
164	7.69	13951	1.29 E-3	3.15 (41.0%)	7218 (51.7%)	1.49 E-8
204	6.06	12166	7.05 E-5	2.48 (40.9%)	6443 (54.0%)	2.98 E-8
Average results	17.89	17.601.2	7.17 E-2	3.84 (21.5%)	7781.5 (44.2%)	2.16 E-7

Denote by t_1 the ratio of the maximal and the minimal computing times when the NAG subroutines are used with $n = 800$ and $c = 4(40)204$. Let t_2 be the corresponding number for the RECKU subroutines. For the matrices $D(800, c)$, $t_1 \approx 4.1$ and $t_2 \approx 1.3$. For the matrices $E(800, c)$, $t_1 \approx 19.8$ and $t_2 \approx 2.8$. The same tendency holds for all other values of n. Moreover, the same is also true for parameter COUNT. However, for this parameter it is more important to emphasize that the whole test described in Table 2 can be carried out if the length of arrays A and SNR is larger than $6.2 * NZ$ for the NAG subroutines, while $2 * NZ$ will be sufficient for the RECKU subroutines. For the test in Table 3, the corresponding numbers are $7.7 * NZ$ and $3 * NZ$. The same tendency holds for all other values of n. The recommended value for the length of A and SNR is $[2 * NZ, 4 * NZ]$ for the NAG subroutines, while the interval for the RECKU subroutines is $[2 * NZ, 3 * NZ]$. The values of COUNT are not known in advance. Therefore, in general, we cannot reserve the optimal storage needed. Nevertheless the results given in Tables 2 and 3, together with those of many other runs (see [34]), allow us to draw the following conclusion, which is very important for the practical use of sparse packages: *It seems to be easier to give a reliable interval for the length of arrays* A *and* SNR *when* SMT+LDT+IR *is used.*

The accuracy of the approximations for $n = 800$ is: $O(10^{-4})$ for the NAG subroutines and for matrices of class $D(800, c)$; from $O(10^{-1})$ to $O(10^{-5})$ for the NAG subroutines and for matrices of class $E(800, c)$; from $O(10^{-6})$ to $O(10^{-8})$ for the RECKU subroutines for both classes. The accuracy obtained when matrices of class $E(800, c)$ are run is given in Table 3. The poor accuracy found with the NAG subroutines for some matrices of class $E(n, c)$ can be explained by the fact that the pivotal strategy used in the NAG subroutines is only a GMS (see, for more details, [31], [32]). All 48 matrices of class $E(n, c)$ have been run with the DS option of our package and $T = 0$. The accuracy was from $O(10^{-4})$ to $O(10^{-6})$. This shows that the use of an IGMS sometimes can give much greater accuracy.

4.3. Solving systems whose matrices have different densities of the nonzero elements. Some test matrices of class $F2(m, n, c, r, \alpha)$, $m = n = 500$, $c = 200$, $\alpha = 100$ and $r = 5(5)40$, have been run. Since $NZ = rm + 110$, the density of the nonzero elements of matrix A, NZ/n^2, has been varied in this experiment. The computing times found are given in Table 4. The accuracy of the approximations was from $O(10^{-2})$ to $O(10^{-4})$ for the NAG subroutines and from $O(10^{-6})$ to $O(10^{-7})$ for the RECKU subroutines. The relation COUNT (NAG) $\approx 3 *$ COUNT (RECKU) holds for all 8 examples (e.g., for $r = 5$ we have COUNT (NAG) = 13450 and COUNT (RECKU) = 5391).

TABLE 4
The computing times when the density, NZ/n^2, of the nonzero elements is varied. The matrices are $F2(500, 500, 20, r, 100)$.

r	NZ	NZ/n^2	NAG subroutines	RECKU subroutines
5	2610	0.01	9.91	2.22 (22.4%)
10	5110	0.02	32.96	6.16 (18.7%)
15	7610	0.03	56.84	11.60 (20.4%)
20	10110	0.04	59.32	14.84 (25.0%)
25	12610	0.05	131.39	25.59 (19.5%)
30	15110	0.06	97.69	34.32 (35.1%)
35	17610	0.07	117.16	50.76 (35.2%)
40	20110	0.08	288.03	62.81 (21.8%)

Note that the computing time for $r = 25$ is larger than that for $r = 30$ when the NAG subroutines are used. It is difficult to explain this phenomenon (this is not caused by a bad preservation of sparsity; the values of COUNT (NAG) are 49150 for $r = 25$ and 55515 for $r = 30$). The only possible explanation is based on the implementation of the pivotal strategy in the NAG subroutines. The pivotal element at stage s is searched among all nonzero elements of A_s ($p(s) = n - s + 1$) (see § 2). The rows and the columns of A_s are searched in order of increasing number of nonzero elements (using rows in preference to columns in cases of tie). The search is terminated if $M_{ijs} < [r(i, s) - 1]^2$ when a row is searched and if $M_{ijs} < c(j, s)[c(j, s) - 1]$ when a column is searched (see Duff and Reid [12, p. 28]). It is stated that normally the process is terminated rather quickly (Duff [10, p. 25]). However, from Duff [10, Table 5], it is seen that the average number of the searched rows and columns can be considerably large (up to 14 even if n is small, $n = 199$). We believe that the large computing time for $r = 25$ is caused by a large average number of searched rows and columns, i.e., the pivotal strategy fails to determine quickly the pivotal elements during the decomposition (1.2).

4.4. Solving badly scaled systems. Matrices of class $F2(m, n, c, r, \alpha)$ with $m = n = 50$, $c = 20$, $r = 4$ and $\alpha = 10^k$, $k = 0(1)6$, have been tested. The results are given in Table 5. The estimations of the condition numbers of the matrices used in this test have been found by a subroutine given in Forsythe et al. [15]. Note that for the next value of $\alpha, \alpha = 10^7$, the iterative process used in the RECKU subroutines is not convergent. Only the accuracy of the approximations is of interest in this experiment. Therefore, small matrices have been run, and neither computing times nor the values of COUNT are given in Table 5. However, it should be mentioned that the NAG subroutines performed better than the RECKU subroutines in this run (with regard to both the computing time and the storage).

TABLE 5
Test with badly scaled matrices $F2(50, 50, 20, 4, \alpha)$ ($NZ = 310$, *estimates of the condition numbers of the matrices are given under* COND).

α	COND	NAG subroutines	RECKU subroutines
10^0	$2.81E+01$	$9.69E-7$	0.0
10^1	$6.52E+03$	$3.04E-5$	$1.04E-7$
10^2	$2.83E+04$	$5.74E-5$	$2.53E-7$
10^3	$4.80E+05$	$9.41E-5$	$3.87E-7$
10^4	$7.80E+06$	$4.42E-5$	$8.94E-7$
10^5	$6.72E+09$	$5.46E-2$	$6.75E-7$
10^6	$6.26E+12$	$1.38E+1$	$2.49E-3$

4.5. Tests with Harwell matrices. All Harwell matrices (36 matrices with n in the range [32, 1250]) have been run. Only the sparsity pattern for some of these matrices is given. If this is so, then we generated the nonzero elements as follows. Let us assume that (according to the prescribed pattern) there is a nonzero element in position (i, j). Then a_{ij} is generated either by

$$a_{ij} = \{\underline{1.0} \text{ if } i = j, \underline{i+j} \text{ if } i > j, \underline{i-j} \text{ if } i < j\} \tag{4.1}$$

or by

$$a_{ij} = \{\underline{2i+1} \text{ if } i = j, \ \underline{i} \text{ if } i > j, \ \underline{j} \text{ if } i < j\}. \tag{4.2}$$

Some of the matrices are rectangular. If this is so, then the augmented system, [1, 2, 3], is formed in the following way:

$$By = c \tag{4.3}$$

where

$$B = \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}, \quad y = \begin{bmatrix} 0 \\ x \end{bmatrix}, \quad c = \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{4.4}$$

When the nonzero elements are given we do not know their magnitude (and we did not try to obtain any information about it). The option which performs row scaling automatically (see § 1) has been used with $T = -10^{-5}$ (the computing time spent in performing scaling is included in the computing time needed to obtain an approximation to the solution, but scaling is a very cheap process). In Table 6 the total results are given. In Table 7 more detailed results for some of the tests are presented.

The NAG subroutines detected numerical singularity twice (when WILL 199 and ABB 313 were solved and the nonzero elements were created by (4.2); some information about the matrices and their names is given in [11]). For matrix GENT 113 the NAG subroutines gave no error message and large errors ($\|x-\bar{x}\|_\infty \approx 10^{+10}$) when both (4.1) and (4.2) were used to generate the nonzero elements. The growth factor (an estimation of $\max_{1\le i,j\le n}(|a_{ij}^{(s^*)}|)/\max_{1\le i,j\le n}(|a_{ij}|)$, $s^* = 1(1)n-1$, found by a device from Erisman and Reid [14]) is large, $2.4E+10$; however, for another matrix, ASH608, the growth factor is $8.5E+12$, but $\|x-\bar{x}\|_\infty \approx 4.5E-8$; an estimation of the condition number of the matrix (Cline et al. [7] and Forsythe et al. [15]) may be helpful in this situation.

TABLE 6
Some results obtained in the experiment with the Harwell test matrices.

Compared characteristics	NAG subroutines	RECKU subroutines
Total time	225.46	118.39
Total COUNT	217871	181906
Number of failures	3	3
Failures detected by the code	2	3
Number of back substitutes	33	106

TABLE 7
Numerical results for some of the Harwell matrices.

Name of the matrix	*NZ*	NAG subroutines Time	NAG subroutines COUNT	RECKU subroutines Time	RECKU subroutines COUNT
ASH 292	2208	1.86	5346	1.78	3766
ARC 130	1282	1.90	1978	1.23	1711
BP 600	4172	4.28	5632	2.94	5321
BP 1400	4790	6.18	7199	4.33	7084
FS 541-1	4285	4.86	12571	1.61	4285

The RECKU subroutines returned with an error message that a row or column without nonzero elements was found in 6 examples. This is an indication that the matrix is singular when $T=0$. If $T>0$ then this is not certain. Therefore, after such information the computations have been repeated with $T=10^{-10}$. In three cases (WILL 199, ABB 313, GENT 113), the second run has not been successful either. In the other three cases (FS 541-2, FS 541-3, FS 541-4), the second run has been successful; see Table 8. When a matrix is run both with $T=10^{-5}$ and $T=10^{-10}$, the sum of the computing times for the two runs is taken into account in Table 6.

Finally, some remarks concerning the experiment with the Harwell matrices are needed. For the successful runs, the accuracy obtained by the NAG subroutines has been in the range $[10^{-2}, 10^{-8}]$, while that obtained by the RECKU subroutines in the range $[10^{-4}, 10^{-8}]$. Some of the Harwell matrices can be represented in block-triangular form. There is an option in the NAG subroutines where this property can be exploited (i.e., the matrix can be transformed into block-triangular form when the matrix allows this). This option has not been used in our runs. We plan to develop a similar subroutine at RECKU.

TABLE 8
Numerical results for the matrices which have been run twice with the RECKU *subroutines* ($T=10^{-5}$ *and* $T=10^{-10}$).

Name of the matrix	NAG subroutines Time	NAG subroutines COUNT	RECKU subroutines Time	RECKU subroutines COUNT
FS 541-2	9.22	15059	0.83+6.07	11085
FS 541-3	10.14	15053	0.73+8.11	12961
FS 541-4	10.44	15666	0.73+6.77	11838

4.6. Solving least-squares problems by augmented matrices. Many experiments performed with rectangular matrices (from the Harwell set and generated by subroutine MATRF2 with $m>n$) show that the RECKU subroutines perform very well in the solution of least-squares problems by the use of augmented matrices. Some results obtained by the use of test matrices from the Harwell set are given in Table 9. In Table 10, examples generated by MATRF2 are given. More numerical results are given in Zlatev [31].

TABLE 9
Solving least-squares problems by augmented $(m+n)*(m+n)$ *matrices* (*NZ is the number of the nonzero elements in the augmented matrix*).

Name of the matrix	Dimensions m	n	NZ	NAG subroutines Time	NAG subroutines COUNT	RECKU subroutines Time	RECKU subroutines COUNT
ASH 219	219	85	1095	3.50	3943	1.47	3004
ASH 958	958	292	4790	77.96	23579	9.70	16356
ASH 331	331	104	1655	8.47	6764	2.44	4913

TABLE 10
Matrices $A=F2\ (m,400,20,2,10)$, $m=1000\ (100)\ 1500$, *are solved. The values of NZ given in the table are the numbers of nonzero elements in the augmented* $(m+n)*(m+n)$*-matrix used in the solution.*

Matrix identifiers		NAG subroutines			RECKU subroutines		
m	NZ	Time	COUNT	Accuracy	Time	COUNT	Accuracy
1000	5220	47.24	10618	4.47 $E-8$	2.73	5333	5.96 $E-8$
1100	5720	65.00	12125	1.49 $E-7$	2.94	5913	7.75 $E-8$
1200	6220	82.04	13648	5.96 $E-8$	3.20	6690	5.96 $E-8$
1300	6720	95.37	14158	4.47 $E-8$	3.51	7379	0.0
1400	7220	109.59	14922	4.47 $E-8$	3.70	7960	1.49 $E-8$
1500	7720	123.36	15874	4.47 $E-8$	3.91	8220	1.49 $E-8$

5. Some concluding remarks.

5.1. When will IR perform better than DS? The examples in § 4 show that IR with a large $T may$ perform better than DS when sparse matrices are solved. However, there is no guarantee that this will always be so. Three examples where the RECKU subroutines perform more poorly than the NAG subroutines are given in Table 11.

In the first example, the RECKU subroutines perform poorly because (i) the matrix is small (and the computational cost of the iteration process (1.4) is a consider-

TABLE 11
Numerical results obtained in the tests with some Harwell matrices (the numbers of iterations are given in brackets).

Name of the matrix	n	NZ	NAG subroutines Time	NAG subroutines COUNT	RECKU subroutines Time	RECKU subroutines COUNT
IBM 32	32	126	0.09	209	0.17 (3)	192
SHL 0	663	1687	0.61	1687	1.11 (3)	1687
STR 600	363	3279	1.64	4409	2.80 (3)	4699

able part of the total computational cost), (ii) extra work is done to perform row scaling, to check if (1.5) is satisfied and to carry out 3 iterations, (iii) COUNT (RECKU) $\approx$ COUNT (NAG).

In the second example, SHL 0, the same explanation for the poor performance of the RECKU subroutines, without (i), can be given. Note that COUNT (RECKU) = COUNT (NAG) = NZ. There are 5 matrices (among the 36) which give the same relation. Moreover, the matrices SHL are permutations of triangular matrices; see Duff and Reid [11].

In the third example, STR 600, an explanation similar to that in the first example, but without (i), can be given. Moreover, COUNT (RECKU) > COUNT (NAG), which means that in this case the pivotal strategy used in the NAG subroutines performs better (the pivotal search is carried out in the whole A_s, while in the RECKU subroutines $p(s)=3$ is used). The possibility that the pivotal strategies with large $p(s)$ could sometimes preserve the sparsity better than the pivotal strategies with small $p(s)$ is noted in Zlatev [32]. However, note that (i) COUNT (RECKU) is small, COUNT (RECKU) < 1.5 NZ, (ii) the pivotal search with large $p(s)$ may be time consuming (this has already been noted in § 4.3; see also the result for matrix BP 1400 in Table 7 where COUNT (NAG) $\approx$ COUNT (RECKU) but the computing time for the NAG subroutines is larger than that for the RECKU subroutines).

Infinitely many matrices which produce few fill-ins can be constructed. For example, any matrix of class $E(n, 2)$ will produce no fill-in. In this situation and if COUNT > NZ but only a few elements are removed by check (1.5), the RECKU subroutines will perform more poorly than the NAG subroutines (with regard both to storage and computing time; however, the IR option may still be preferable because it normally gives better accuracy and reliable error estimation).

The situation described above is not typical. In many examples COUNT $\gg NZ$ has been found. If COUNT $\gg NZ$, then the use of SMT+LDT+IR is normally very efficient. Some numerical results are given in Tables 12 and 13 to illustrate this.

We can conclude that *for "cheap" problems* (where COUNT $\approx NZ$ or COUNT is not much larger than NZ), *the* RECKU *subroutines may perform more poorly than the* NAG *subroutines* (with regard to the storage and computing time used). *However, for "expensive" problems the* RECKU *subroutines will normally give much better results* (a reduction in computing time of up to 10 times and a reduction in the storage of up to 2–3 times have been observed; note too that the IR process will usually produce more accurate approximations, and reliable error estimations are available).

5.2. On the convergence of IR. It is possible that the iterative process (1.4) will not converge [25], [27], [28]. This is the case when matrix A is too ill-conditioned and/or if the drop tolerance T is too large. Normally, the code provides a clear indication in this situation. Smaller values of T may give good results when this

TABLE 12

Matrices which generate many fill-ins (COUNT $\gg NZ$). *The matrices are* $E(n, 44)$, $NZ = 5n - 2c - 2$. *Note that if the length of arrays A and SNR is* $3*NZ$, *then the* RECKU *subroutines will succeed in the solution, while even if this length is* $9*NZ$ *the* NAG *subroutines will not.*

n	NAG subroutines		RECKU subroutines	
	Time	COUNT	Time	COUNT
650	23.47	22246	4.55 (19.4%)	7697 (34.6%)
700	28.46	24286	5.27 (18.5%)	8453 (34.8%)
750	38.29	26932	6.12 (16.0%)	9174 (34.1%)
800	53.67	30424	6.88 (12.8%)	9882 (32.5%)
850	58.54	35290	6.97 (11.9%)	11646 (33.0%)
900	57.35	37230	7.47 (13.0%)	12360 (31.2%)
950	115.58	40488	8.07 (7.0%)	12551 (31.0%)
1000	152.31	45850	8.50 (5.6%)	14082 (30.7%)

TABLE 13

Performance of two options of the RECKU *subroutines for some large chemical problems* [24], *where* $n = 255$, $NZ = 7715$.

Option	Tolerance	Time	COUNT
DS	10^{-14}	41.25	23837
IR	10^{-2}	16.77	16808

happens. If matrix A is extremely ill-conditioned, the iterative process will not converge even with $T = 0$. DS with double precision computation "might give an answer of acceptable (but unknown) accuracy" (Golub and Wilkinson [17, p. 148]).

5.3. On the rate of convergence. Normally the iterative process (1.4) is terminated after 3–4 iterations. However, for some matrices (e.g., these of class $E(n, c)$ with $n = c^2$), the rate of convergence can be very slow. Therefore some rules which accelerate the convergence may be useful sometimes. Unfortunately, the rules normally used in practice, work only for systems with symmetric and positive definite matrices. Some attempts to accelerate the rate of convergence for systems of general matrices have been carried out. We are continuing the experiments in this direction.

5.4. Main conclusion. In the previous studies (Schaumburg et al. [24], Zlatev [31], Zlatev et al. [34], Zlatev and Wasniewski [36]), it has been verified that the IR option of our package Y12M performs better than the DS option. Of course, this will not be very valuable if our DS option is a bad one. Therefore, in this paper we picked out a good code, oriented to the user, the NAG subroutines, which performs DS only. Then we compared the two algorithms, iterative refinement with a large drop tolerance as implemented in Y12M, and the DS as implemented in the NAG subroutines. It has been shown that not only are a higher degree of accuracy and a reliable error estimation normally found when IR is used, but also the storage or the computing time, or both, can often be reduced. Moreover, the reduction is sometimes very considerable. This is an unexpected result (if the matrix is dense, then both more storage and more computing time are *always* needed with IR). This shows that IR is a useful option for any package based on the use of some sparse matrix technique.

However, we must also emphasize that IR should only be an option in the package. For some problems the DS option can be used successfully (e.g., if only a few fill-ins are produced or if many systems with the same matrix have to be solved).

REFERENCES

[1] Å. Björck, *Iterative refinement of linear least squares problems I*, BIT, 7 (1967), pp. 257–278.

[2] ———, *Iterative refinement of linear least squares problems II*, BIT, 8 (1968), pp. 1–30.

[3] ———, *Methods for sparse linear least squares problems*, in Sparse Matrix Computations, J. Bunch and D. Rose, eds., Academic Press, New York, 1976, pp. 179–199.

[4] ———, *Comment on the iterative refinement of least squares problems*, J. Amer. Statist. Assoc., 73 (1979), pp. 161-166.

[5] R. K. Brayton, F. G. Gustavson and R. A. Willoughby, *Some results on sparse matrices*, Math. Comp., 24 (1970), pp. 937–954.

[6] R. J. Clasen, *Techniques for automatic tolerance in linear programming*, Comm. ACM, 9 (1966), pp. 802-803.

[7] A. K. Cline, C. B. Moler, G. W. Stewart and J. M. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368–375.

[8] A. R. Curtis and J. K. Reid, *The solution of large sparse unsymmetric systems of linear equations*, J. Inst. Math. Appl., 8 (1971), pp. 344–355.

[9] I. S. Duff, *A survey of sparse matrix research*, Report CSS 28, A.E.R.E., Harwell, England, 1976.

[10] ———, MA28—*a set of FORTRAN subroutines for sparse unsymmetric matrices*, Report R8730, A.E.R.E., Harwell, England, 1977.

[11] I. S. Duff and J. K. Reid, *Performance evaluation of codes for sparse matrix problems*, Report CSS 66, A.E.R.E., Harwell, England, 1978.

[12] ———, *Some design features of a sparse matrix code*, ACM Trans. Math. Software, 5 (1979), pp. 18–35.

[13] I. S. Duff, J. K. Reid, N. Munksgaard and H. B. Nielsen, *Direct solution of sets of linear equations whose matrices are sparse, symmetric and indefinite*, J. Inst. Math. Appl., 23 (1979), pp. 235–250.

[14] A. M. Erisman and J. K. Reid, *Monitoring the stability of the triangular factorization of a sparse matrix*, Numer. Math., 22 (1974), pp. 183–186.

[15] G. E. Forsythe, M. A. Malcolm and C. B. Moler, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[16] G. E. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[17] G. H. Golub and J. H. Wilkinson, *Note on the iterative refinement of least squares solution*, Numer. Math., 9 (1966), pp. 139–148.

[18] F. G. Gustavson, *Some basic techniques for solving sparse systems of linear equations*, in Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972, pp. 41–52.

[19] ———, *Two fast algorithms for sparse matrices: Multiplication and permuted transposition*, ACM Trans. Math. Software, 4 (1978), pp. 250–269.

[20] H. M. Markowitz, *The elimination form of the inverse and its application to linear programming*, Management Sci., 3 (1957), pp. 255–269.

[21] N. Munksgaard, *FORTRAN SUBROUTINES for direct solution of sets of sparse and symmetric linear equations*, Report 77-05, Institute for Numerical Analysis, Technical Univ. of Denmark, Lyngby, Denmark, 1977.

[22] J. K. Reid, *Fortran subroutines for handling sparse linear programming bases*, Report R8269, A.E.R.E., Harwell, England, 1976.

[23] ———, *Solution of linear systems of equations: Direct methods (general)*, in Sparse Matrix Techniques, V. A. Barker, ed., Lecture Notes in Mathematics 572, Springer, Berlin, 1977, pp. 102–129.

[24] K. Schaumburg, J. Wasniewski and Z. Zlatev, *The use of sparse matrix technique in the numerical integration of stiff systems of linear ordinary differential equations*, Comput. Chem., 4 (1980), pp. 1-12.

[25] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[26] R. P. Tewarson, *Sparse Matrices*, Academic Press, New York, 1973.

[27] J. M. Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

[28] ———, *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, London, 1965.

[29] J. M. WILKINSON AND C. REINSCH, *Handbook for Automatic Computation, Vol.* 2, *Linear Algebra*, Springer, Berlin, 1971.

[30] P. WOLFE, *Error in the solution of linear programming problems*, in Error in Digital Computations, Vol. 2, L. B. Rall, Ed., John Wiley, New York, 1965, pp. 271–284.

[31] Z. ZLATEV, *Use of iterative refinement in the solution of sparse linear systems*, Report 1/79, Institute for Mathematics and Statistics, Royal Veterinary and Agricultural Univ., Copenhagen, 1979; SIAM J. Numer. Anal., 19 (1982), pp. 381–399.

[32] ———, *On some pivotal strategies in Gaussian elimination by sparse technique*, SIAM J. Numer. Anal., 17 (1980), pp. 18–30.

[33] Z. ZLATEV AND V. A. BARKER, *Logical procedure SSLEST—an Algol W procedure for solving sparse systems of linear equations*, Report 76–13, Institute for Numerical Analysis, Technical Univ. of Denmark, Lyngby, Denmark, 1976.

[34] Z. ZLATEV, K. SCHAUMBURG AND J. WASNIEWSKI, *Implementation of an iterative refinement option in a code for large and sparse systems*, Comput. Chem., 4 (1980), pp. 87–99.

[35] Z. ZLATEV AND P. G. THOMSEN, *ST—a FORTRAN IV subroutine for the solution of large systems of linear algebraic equations with real coefficients by use of sparse technique*, Report 76–05, Institute for Numerical Analysis, Technical Univ. of Denmark, Lyngby, Denmark, 1976.

[36] Z. ZLATEV AND J. WASNIEWSKI, *Package* Y12M—*solution of large and sparse systems of linear algebraic equations*, Preprint Series 24 1978, Mathematics Institute, Univ. of Copenhagen, Copenhagen, 1978.

[37] Z. ZLATEV, J. WASNIEWSKI AND K. SCHAUMBURG, Y12M—*solution of large and sparse systems of linear algebraic equations* (*documentation of subroutines*), Lecture Notes in Computer Science, Springer, Berlin, 1981.

[38] ———, *A testing scheme for subroutines for solving large linear problems*, Comput. Chem., 5 (1981), pp. 91–100.

SIAM J. SCI. STAT. COMPUT.
Vol. 3, No. 4, December 1982

THE SIMULATION OF GENERALIZED INVERSE GAUSSIAN AND HYPERBOLIC RANDOM VARIABLES*

A. C. ATKINSON†

Abstract. Computer algorithms are described for simulation of the generalized inverse Gaussian, generalized hyperbolic and hyperbolic distributions. The efficiencies of the algorithms are found. Timing comparisons with the best available algorithms for sampling the gamma distribution show the new algorithms to be acceptably fast. The extension to sampling multivariate generalized hyperbolic distributions is described. Listings of Fortran implementations of the algorithms are available.

Key words. generalized inverse Gaussian random variable, hyperbolic random variable, pseudorandom variable, rejection algorithm, simulation

1. Introduction.

The hyperbolic distribution was introduced by Barndorff-Nielsen (1977) to model the log size distribution of aeolian, that is wind blown, sand deposits. The purpose of the present paper is to describe algorithms for the simulation of this and the related generalized hyperbolic and generalized inverse Gaussian distributions. Since these distributions are not well known, even to statisticians, we begin with a short description of their properties, uses and interrelationships.

The generalized inverse Gaussian distribution has probability density function

$$f(x)=\frac{(\psi/\chi)^{\lambda/2}}{2K_\lambda(\sqrt{\psi\chi})}e(x;\lambda,\chi,\psi) \tag{1}$$

where

$$e(x;\lambda,\chi,\psi)=x^{\lambda-1}e^{-(1/2)(\chi x^{-1}+\psi x)} \qquad (x>0)$$

and K_λ is the modified Bessel function of the third kind with index λ. The distribution is denoted $N^-(\lambda,\chi,\psi)$. The domain of variation for the parameters is

$$\begin{aligned} &\chi>0, \quad \psi\geqq 0 \quad \text{if } \lambda<0,\\ &\chi>0, \quad \psi>0 \quad \text{if } \lambda=0,\\ &\chi\geqq 0, \quad \psi>0 \quad \text{if } \lambda>0.\end{aligned}$$

One special case is the gamma distribution ($\chi=0$), which follows from the Bessel function limit

$$K_\nu(x)\sim\Gamma(\nu)2^{\nu-1}x^{-\nu} \quad \text{as } x\downarrow 0 \quad (\nu>0).$$

Other special cases are the distribution of the reciprocal of a gamma random variable ($\psi=0$) and the inverse Gaussian distribution ($\lambda=-\frac{1}{2}$).

The density (1) can be reparameterized by setting $\omega=\sqrt{\chi\psi}$, a shape parameter, and $\eta=\sqrt{\chi/\psi}$, a scale parameter. The two parameters are therefore λ and ω. Figure 1 shows the shapes of three members of this two-parameter family when $\eta=1$, so that $\chi=\psi=\omega$. As would be expected from the special cases, the distributions are, in shape, like an enriched family of gamma distributions.

The primary importance of the generalized inverse Gaussian lies in its use not for the description of data, but as a means of simulating other distributions via mixing. If σ^2 is sampled from an appropriately parameterized form of (1), the random variable

* Received by the editors November 9, 1979, and in revised form May 15, 1981.

† Imperial College, London SW7 2BZ.

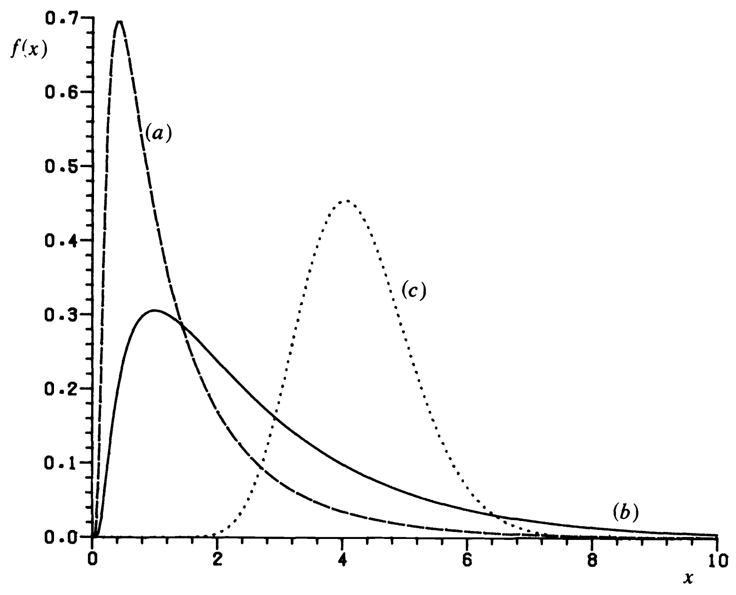


FIG. 1. *The density function of the generalized inverse Gaussian distribution*: (a) $\lambda = 0$, $\omega = 1$, (b) $\lambda = 1$, $\omega = 1$ *and* (c) $\lambda = 20$, $\omega = 10$.

Y with conditional normal distribution of mean $\xi = \mu + \beta\sigma^2$ and variance σ^2 has the generalized hyperbolic distribution with probability density function

$$a_\lambda(\alpha, \beta, \delta)\{\sqrt{\delta^2 + (x-\mu)^2}\}^{\lambda - 1/2} K_{\lambda - 1/2}(\alpha\sqrt{\delta^2 + (x-\mu)^2})\, e^{\beta(x-\mu)}. \tag{2}$$

Here λ, α, β, μ and δ are parameters and the normalizing constant is

$$a_\lambda(\alpha, \beta, \delta) = (\alpha^2 - \beta^2)^{\lambda/2} / \{\sqrt{2\pi}\alpha^{\lambda - 1/2}\delta^\lambda K_\lambda(\delta\sqrt{\alpha^2 - \beta^2})\}$$

with K_ν the modified Bessel function of the third kind with index ν. An algorithm which uses the mixing property is described in § 5.

An important special case of this distribution occurs when $\lambda = 1$. The resulting hyperbolic distribution has density

$$f(x) = \frac{\zeta}{2\alpha K_1(\zeta)} e^{-\alpha\sqrt{1+x^2} + \beta x} \tag{3}$$

where $\zeta = \sqrt{\alpha^2 - \beta^2}$ and, in (2), the location and scale parameters μ and δ have been set to 0 and 1 respectively. In addition to sand particles, the distribution has been found (Barndorff-Nielsen (1977), (1978), (1979)) to apply to sizes of diamonds, to incomes and to some distributions arising in turbulence.

The graph of the logarithm of the density (3) is a hyperbola with asymptotes ϕx and $-\gamma x$, where $\phi = \alpha + \beta$ and $\gamma = \alpha - \beta$. For $\beta = 0$ the distribution is symmetric. The logarithmic plot of Fig. 2 shows a symmetrical hyperbolic distribution together with a normal distribution of the same mean and variance which plots as a parabola. The asymptotes of the hyperbolic distribution are proportional to the density of a double exponential, or Laplace, distribution. Thus the hyperbolic distribution has a near normal center but exponential tails, a combination of importance in robust statistical

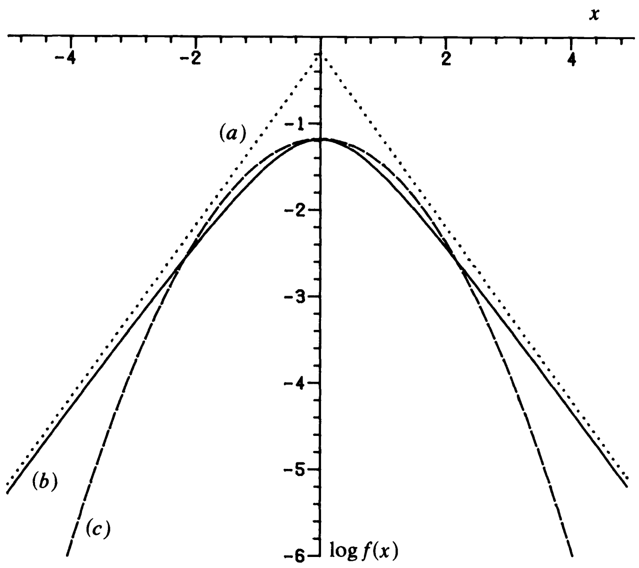


FIG. 2. *Log density functions:* (a) *asymptotes of hyperbolic distribution,* (b) *hyperbolic distribution* $\alpha = 1$, $\beta = 0$ *and* (c) *normal distribution.*

procedures. One reason for the development of the algorithms described here was interest in the robustness properties of estimates based on the hyperbolic distribution. An important asset of this approach is that when β in (3) is nonzero, asymmetric distributions result. Customary robust procedures are concerned with symmetrical situations.

The paper begins in § 2 with a review of rejection algorithms. Section 3 applies the method to sampling the generalized inverse Gaussian distribution. A more efficient method for values of λ in the range $0 \leqq \lambda < 1$ is derived in § 4. The use of these algorithms to produce generalized hyperbolic distributions by mixing is described in § 5. The next section considers the special case of the inverse Gaussian distribution with $\lambda = 1$. The resulting algorithm is used in § 7 to sample the hyperbolic distribution by mixing. Comparisons are made in the same section with a method which uses rejection on the hyperbolic distribution itself.

Section 8 compares the algorithm of § 3 in the special case of the gamma distribution with some of the better known algorithms for the gamma distribution. Section 9 uses the generalized inverse Gaussian as a mixing distribution with multivariate normal distributions to yield multivariate generalized hyperbolic distributions.

From a practical point of view the most useful of these distributions are probably the generalized inverse Gaussian distributions with index $\lambda = 0$ and 1 and the family of r-dimensional hyperbolic distributions produced by mixing multivariate normal distributions with generalized inverse Gaussian distributions of index $(r+1)/2$. The family has the property that the log density forms an r-dimensional hyperboloid.

2. Rejection algorithms. The algorithms for sampling the generalized inverse Gaussian and hyperbolic distributions use the envelope rejection technique with first stage sampling from an envelope $g(x)$, which is sampled by inversion. If U is uniformly distributed on (0, 1), the method is to set $U = g(X)$ and invert analytically to obtain $X = g^{-1}(U)$.

Let the density to be sampled be $f(x) = ce(x)$. We divide the range of x into two parts. Since generalized inverse Gaussian variates are nonnegative we take these as $(0, t]$ and (t, ∞). If we let

(4) $$g(x) = k_i d_i(x), \qquad i = \begin{cases} 1 & (0 \leqq x \leqq t), \\ 2 & (x > t), \end{cases}$$

with $h_i(x) = e(x)/d_i(x)$ and $S_i = \sup h_i(x)$ over $(0, t]$ or (t, ∞) as indicated by i, the generated value X is now accepted if

(5) $$U^* \leqq h_i(X)/S_i,$$

where U^* is uniformly distributed on $(0, 1)$, independently of U. The probability of accepting a given value x is $k_i e(x)/S_i$. To ensure that $f(x)$ is sampled correctly, we require

(6) $$\frac{k_1}{S_1} = \frac{k_2}{S_2} = F,$$

and then the efficiency of the algorithm is $E = F/c$. A second requirement is that $g(x)$ be a density, so that, in an obvious notation,

(7) $$1 = \int_0^\infty g(x) = k_1 \int_0^t d_1(z)\,dz + k_2 \int_t^\infty d_2(z)\,dz = k_1\Delta_1 + k_2\Delta_2,$$

which defines the Δ_i.

We now solve for k_1 and k_2. If (6) is rewritten as

(8) $$k = k_1 S_2 = k_2 S_1,$$

combination with (7) leads to

(9) $$k^{-1} = \frac{\Delta_1}{S_2} + \frac{\Delta_2}{S_1}.$$

The efficiency of the algorithm is, from (6),

(10) $$E = \frac{F}{c} = \frac{1}{c(S_1\Delta_1 + S_2\Delta_2)}.$$

The algorithm with highest efficiency is found by choosing the envelopes $d_i(x)$ to minimize $S_1\Delta_1 + S_2\Delta_2$. If t is fixed, the choices in the two regions are therefore independent of each other.

The method can be extended by dividing the range of x into three or more parts. Figure 3 shows a three-part envelope for the hyperbolic distribution which is described in § 7. Algorithms with two-part envelopes have been used for the gamma distribution with index less than one (Ahrens and Dieter (1974, Algorithm GS)) and in the switching algorithm for the beta distribution (Atkinson and Whittaker (1976)). We now apply the method to sampling the generalized inverse Gaussian distribution.

3. The generalized inverse Gaussian distribution. Properties of the generalized inverse Gaussian distribution (1) are given by Blæsild (1978). One property we shall use is that the distribution is unimodal with mode

(11) $$t = m(\lambda, \chi, \psi) = \begin{cases} \dfrac{(\lambda - 1) + \sqrt{(1-\lambda)^2 + \chi\psi}}{\psi} & (\psi > 0), \\ \dfrac{\chi}{2(1-\lambda)} & (\psi = 0). \end{cases}$$

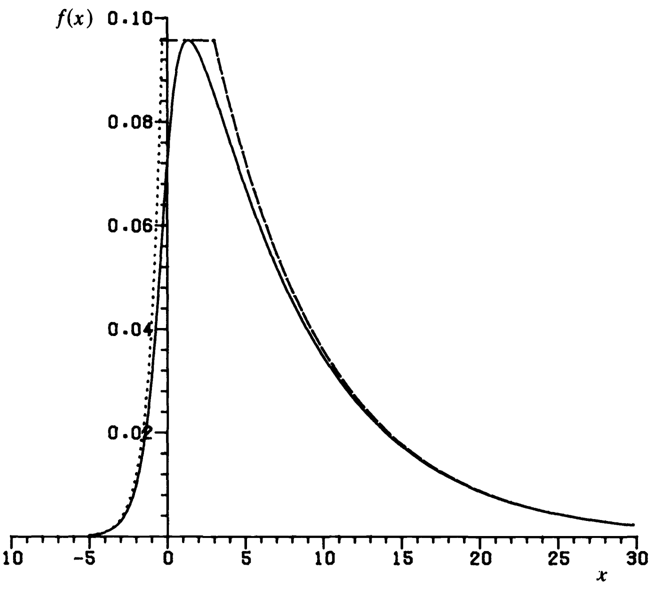


FIG. 3. *Density function of the hyperbolic distribution* $\alpha = 0.7$, $\beta = 0.56$ *showing three-part envelope for Algorithm* HYP.

The distribution is simulated by dividing the range of x at the mode, a convenient, if nonoptimal, choice. The envelopes are the exponentials

$$d_1(x) = e^{sx} \quad \text{and} \quad d_2(x) = e^{-px} \tag{12}$$

so that in (7)

$$\Delta_1 = \frac{(e^{st} - 1)}{s} \quad \text{and} \quad \Delta_2 = \frac{e^{-pt}}{p}. \tag{13}$$

One advantage of this choice is that $g(x)$ is readily sampled. A second is that the rejection functions $h_i(x)$ are proportional to generalized inverse Gaussian densities and are, in the notation of (1),

$$h_1(x) = e(x; \lambda, \chi, \psi + 2s) \quad \text{and} \quad h_2(x) = e(x; \lambda, \chi, \psi - 2p). \tag{14}$$

This representation makes the point that we must have $p \leqq \psi/2$ for the right-hand tail of $f(x)$ to be covered by the envelope. It follows from (11) and (14) that the suprema of $h_i(x)$ are given by

$$\begin{aligned} S_1 &= h_1(x_L) \quad \text{where } x_L = m(\lambda, \chi, \psi + 2s), \\ S_2 &= h_2(x_H) \quad \text{where } x_H = m(\lambda, \chi, \psi - 2p). \end{aligned} \tag{15}$$

It remains to find values of s and p. The optimum values minimize $\Delta_1 S_1 + \Delta_2 S_2$. It is not possible to find such values analytically except for the special case of the gamma distribution. One possibility which was not explored was to perform a numerical optimization to find p and s. Instead crude step searches were used which are described in Atkinson [1979b, § 3]. Evidence to be given in § 8 suggests that the neighbourhood of the optimum is sufficiently flat for this procedure to be negligibly suboptimum.

The algorithm involves one further constant. If the distribution function of the envelope is $G(x)$, let $r = G(t) = k_1\Delta_1$, where k_1 is given by (8) and (9).

```
ALGORITHM GIG ($\lambda$ arbitrary).
Set $r = k_1\Delta_1$.
1. Generate $U$ and $U^*$. If $U > r$, go to 2.
   $x = \frac{1}{s}\log\left(1+\frac{sU}{k_1}\right)$.
   If $\log U^* > \log\{h_1(x)/S_1\}$ go to 1.
   Otherwise return $x$.
2. $x = -\frac{1}{p}\log\left\{\frac{p}{k_2}(1-U)\right\}$.
   If $\log U^* > \log\{h_2(x)/S_2\}$ go to 1.
   Otherwise return $x$.
```

In the program listings in Atkinson (1979b, Appendix) these statements were rewritten to increase speed of execution.

The results of efficiency calculations for Algorithm GIG are presented in Table 1, for parameters λ and ω with $\eta = 1$. As an alternative to the calculation of the Bessel

TABLE 1
Estimated % efficiency of algorithm GIG *for generating generalized inverse Gaussian random variables.*

	λ									
$\omega\ (=\chi=\psi)$	−1	−0.5	0	0.5	1	2	3	5	10	20
0.1	0.4	0.8	3.2	18.6	91.6	87.0	83.8	79.4	77.7	76.1
0.5	10.3	16.3	32.0	63.4	93.1	85.5	83.5	78.7	76.1	76.6
1	29.3	41.6	61.0	84.1	89.7	85.2	82.5	79.5	76.3	76.8
5	81.3	81.4	79.5	80.8	79.9	79.8	78.7	78.0	75.5	75.4
10	77.1	76.6	76.1	76.8	76.6	75.3	76.6	75.8	75.6	76.0

function in the density (1) the efficiency was found by simulation. The values given are the percentage of 10,000 calls to Step 1 that resulted in the generation of a generalized inverse Gaussian random variable. For values of $\lambda \geqq 1$ the efficiency is greater than 75%, and the algorithm can be considered satisfactory. The behaviour for $\lambda < 1$ is, on the other hand, not good enough. The timings in Table 2 were carried out on the Cyber 174 at Imperial College in the manner described by Atkinson and Pearce (1976). Ten thousand variables were generated in a DO loop, and the time

TABLE 2
Average time, in μsec *on the Cyber* 174, *to generate one generalized inverse Gaussian random variable using Algorithm* GIG.

	λ							
$\omega\ (=\chi=\psi)$	0	0.5	1	2	3	5	10	20
0.1	6663	1186	255	273	284	297	307	308
0.5	698	359	253	276	286	296	304	307
1	376	278	261	278	282	296	308	306
5	291	290	289	293	297	300	310	308
10	305	307	303	307	307	308	308	311

for the empty loop subtracted. The CDC random number generator RANF was used with the MNF5 compiler. For values of $\lambda \geqq 1$ the timings are roughly 250–310 μsec per random variable. The results of § 8 show that times of this order are comparable with those for the best algorithms for the gamma distribution.

The next section describes an algorithm for λ in the region $0 \leqq \lambda < 1$. Negative values of λ do not need to be considered since if $X \sim N^{-}(\lambda, \chi, \psi)$, then $1/X \sim N^{-}(-\lambda, \psi, \chi)$. Thus for $\lambda < 0$ one simulates a variable with index $-\lambda$ and with χ and ψ interchanged, and then returns the reciprocal of the value.

4. An improved algorithm for the generalized inverse Gaussian distribution with $0 \leqq \lambda < 1$. For values of λ greater than one, the two-part algorithm of § 3 is efficient because the right-hand tail of the log density is concave so that a value of $p < \psi/2$ can be used. As a result, the envelope provides a good approximation to the distribution where the density is appreciable. But, for $\lambda < 1$, the right-hand tail of the distribution is, in log form, asymptotically $(\lambda - 1) \log x - \psi x/2$, which is convex with asymptotic slope $-\psi/2$. In the two-part algorithm a value of $\psi/2$ has therefore to be used for p, and this can give a poor fit to the heavy-tailed distribution. In order to accommodate the upper tail of the distribution, the region above the mode can be broken into two parts. The general form of the resulting algorithm is a direct extension of that of § 2.

Instead of (4) we now let

$$g(x) = k_i d_i(x), \qquad i = \begin{cases} 1 & (0 \leqq x \leqq t), \\ 2 & (t < x \leqq w), \\ 3 & (x > w). \end{cases}$$

The general notation remains the same. The extensions of (7) and (8) are that

$$1 = k_1 \Delta_1 + k_2 \Delta_2 + k_3 \Delta_3$$

and

$$k = k_1 S_2 S_3 = k_2 S_1 S_3 = k_3 S_1 S_2.$$

The replacement for (9) is that

$$k^{-1} = \frac{\Delta_1 S_1 + \Delta_2 S_2 + \Delta_3 S_3}{S_1 S_2 S_3},$$

and the efficiency of the algorithm is

$$E = \frac{1}{c(S_1 \Delta_1 + S_2 \Delta_2 + S_3 \Delta_3)}. \tag{16}$$

The three envelopes are

$$d_1(x) = e^{sx}, \quad d_2(x) = e^{-px}, \quad d_3(x) = e^{-qx},$$

so that (13) becomes

$$\Delta_1 = \frac{e^{st} - 1}{s}, \quad \Delta_2 = \frac{e^{-pt} - e^{-pw}}{p}, \quad \Delta_3 = \frac{e^{-qw}}{q}.$$

Because of the convexity of the upper tail of the log density, we know that $q = \psi/2$. Given w, the values of p and s are found as for Algorithm GIG. To find w an extra dimension of search is needed, the details of which are given in Atkinson (1979b). The resulting algorithm is otherwise similar to that of § 3.

```
ALGORITHM GIGLT1 ($0 \leqq \lambda < 1$).
Set $r = k_1\Delta_1$ and $v = 1 - k_3\Delta_3$.
1. Generate $U$ and $U^*$. If $U > r$, go to 2.
   $x = \frac{1}{s}\log\left(1 + \frac{sU}{k_1}\right)$.
   If $\log U^* > \log\{h_1(x)/S_1\}$ go to 1.
   Otherwise return $x$.
2. If $U > v$, go to 3.
   $x = -\frac{1}{p}\log\left\{\frac{p}{k_2}(v - U) + e^{-pw}\right\}$.
   If $\log U^* > \log\{h_2(x)/S_2\}$ go to 1.
   Otherwise return $x$.
3. $x = -\frac{1}{q}\log\left\{\frac{q}{k_3}(1 - U)\right\}$.
   If $\log U^* > \log\{h_3(x)/S_3\}$ go to 1.
   Otherwise return $x$.
```

Efficiencies for this algorithm, again calculated from 10^4 calls to Step 1, are given in Table 3. These show a distinct improvement over those for the two-part algorithm given in Table 1. For all parameter values except $\omega = 0.1$ and $0 \leqq \lambda \leqq 0.5$, the efficiency is greater than 50%. For the worst case, $\omega = 0.1$ and $\lambda = 0$, the efficiency is 21.5% compared with 3.2% for Algorithm GIG, a nearly sevenfold improvement. Timings which are in line with these efficiencies, are given in Table 4.

TABLE 3
Estimated % efficiency of algorithm GIGLT1 *for generating generalized inverse Gaussian random variables, parameter* $\lambda < 1$.

	λ						
ω $(=\chi=\psi)$	0	0.1	0.3	0.5	0.7	0.9	0.99
0.1	21.5	24.5	32.8	47.5	67.0	90.4	99.0
0.5	55.9	59.6	69.1	77.6	88.6	95.1	95.7
1	74.9	77.6	83.4	88.1	91.2	93.0	90.5
5	84.6	84.2	82.2	81.5	80.4	80.9	80.6
10	78.6	77.5	76.9	76.8	76.6	75.7	76.7

TABLE 4
Average time, in μsec *on the Cyber* 174, *to generate one generalized inverse Gaussian random variable, parameter* $\lambda < 1$, *using algorithm* GIGLT1.

	λ						
ω $(=\chi=\psi)$	0	0.1	0.3	0.5	0.7	0.9	0.99
0.1	1054	926	672	490	351	266	242
0.5	416	388	342	304	273	250	252
1	318	309	287	270	261	257	265
5	283	284	285	294	297	296	297
10	305	306	309	311	312	310	311

The algorithm GIGLT1 thus provides a useful complement to Algorithm GIG for simulation of the generalized inverse Gaussian distribution. For $\lambda > 1$ the efficiency of GIGLT1 is not markedly better than that of the two-part algorithm because of the restriction in the upper tail that $q = \psi/2$.

5. The generalized hyperbolic distribution. Since the argument x enters the generalized hyperbolic density (2) through a Bessel function, there is no hope, in the general case, of a fast rejection algorithm for simulation of the distribution. Exceptions are the special cases $\lambda = 0$ and $\lambda = 1$. However, simulation is possible via the representation, mentioned in § 1, of the generalized hyperbolic distribution as a conditional normal distribution mixed with the generalized inverse Gaussian. A schematic representation of the algorithm follows:

```
Set $\chi = \delta^2$ and $\psi = \alpha^2 - \beta^2$.
Sample $X$ from $N^-(\lambda, \chi, \psi)$ and set $\sigma^2 = X$.
Sample $Z$ from $N(0, 1)$.
Return $Y = \mu + \beta\sigma^2 + \sigma Z$.
```

The efficiency of this method depends upon the parameters of the generalized inverse Gaussian distribution. The results of § 7 indicate that, depending upon the algorithm used to sample the normal distribution, 100–150 μsec per variate should be added to the timings in Tables 2 and 4 to give timings for generating generalized hyperbolic random variables.

6. The generalized inverse Gaussian distribution with λ = 1. The special case of the generalized inverse Gaussian distribution with $\lambda = 1$ is important in sampling the hyperbolic distribution, a topic which is discussed in § 7. When λ equals 1, Algorithm GIG described in § 3 can be used without modification, when the rejection steps require the calculation of $h_i(x) \propto (\lambda - 1)\log x - \frac{1}{2}(\chi x^{-1} + \psi x)$. If λ equals 1, calculation of $\log x$ is not required and the algorithm can be speeded up by omitting this expression. Timing comparisons in Atkinson (1979b, Table 5) show that omission of the logarithm leads to a saving in time of just over 25%.

One other special case merits comment: the inverse Gaussian distribution (Tweedie (1957)), for which $\lambda = -\frac{1}{2}$. Algorithms for sampling this distribution are given by Michael, Schucany and Haas (1976) and, independently, by Padgett (1978). Both algorithms include arithmetic operations, including a square root, on a generated normal random variable. They are therefore unlikely to be appreciably faster than use of the algorithm of § 4 to generate the reciprocal of the required value.

7. The hyperbolic distribution. The hyperbolic distribution (3) can be sampled by the mixing method of § 5 or by rejection. We first describe an algorithm with three-part envelope analogous to Algorithm GIGLT1 of § 4.

At the mode of the distribution, if the normalizing constant is ignored, $\log f(x) = \theta = -\sqrt{\phi\gamma} = -\sqrt{\alpha^2 - \beta^2}$. The three envelopes employed are the two asymptotes and a uniform distribution over the central region. These three intersect at

$$t = -\sqrt{\gamma/\phi} \quad \text{and} \quad w = \sqrt{\phi/\gamma} = -\frac{1}{t}.$$

The three-part envelope is thus

$$g(x) = \begin{cases} ke^{\phi x} & (x \leqq t), \\ ke^{\theta} & (t < x \leqq w), \\ ke^{-\gamma x} & (x > w), \end{cases}$$

where, since $S_1 = S_2 = S_3 = 1$, all k_i in the analogue of (8) equal k. Figure 3 shows one example of the resulting envelope.

The remaining quantities required for the algorithm are

$$\Delta_1 = \frac{e^{\phi t}}{\phi} = \frac{e^{\theta}}{\phi}, \quad \Delta_2 = (w-t)e^{\theta}, \quad \Delta_3 = \frac{e^{-\gamma w}}{\gamma}.$$

In the following description of the resulting algorithm, which is similar in structure to Algorithm GIGLT1 of § 4, E is an exponential random variable with unit parameter.

```
ALGORITHM HYP.
Set $r = k\Delta_1$ and $v = 1 - k\Delta_3$.
1. Generate $U$ and $E$. If $U > r$, go to 2.
   $x = \frac{1}{\phi}\log\left(\frac{\phi U}{k}\right)$.
   If $E < \alpha\{\sqrt{1+x^2}+x\}$ go to 1.
   Otherwise return $x$.
2. If $U > v$, go to 3.
   $x = t - \frac{1}{\phi} + Ue^{-\theta}/k$.
   If $E < \alpha\sqrt{1+x^2} - \beta x + \theta$ go to 1.
   Otherwise return $x$.
3. $x = \frac{1}{\gamma}\log\frac{k}{\gamma} - \frac{1}{\gamma}\log(1-U)$.
   If $E < \alpha\{\sqrt{1+x^2}-x\}$ go to 1.
   Otherwise return $x$.
```

The exponential variables arise from replacement of the rejection condition $U^* \geqq h(x)/S$ by the equivalent condition $E \leqq -\log\{h(x)/S\}$. In the implementation of the algorithm the exponential random variables are generated by von Neumann's method (Ahrens and Dieter (1972, Algorithm NE)).

To compare this algorithm with the mixing algorithm of § 5, we take $\mu = 0$ and $\delta = 1$. The parameters of the generalized inverse Gaussian with $\lambda = 1$ are thus $\chi = \delta^2 = 1$ and $\psi = \alpha^2 - \beta^2$.

Table 5 can be used to obtain comparative timings for a range of α values from 0.1 to 10 and β values from 0 to 0.8. Table 5(a) gives times for the special version of GIG for $\lambda = 1$ which was described in § 6. Timings for the mixing algorithm are obtained by adding the time to generate a normal random variable and to form the hyperbolic variable. The results in Atkinson (1979b, Table 6) show that if normal variables are generated by the polar method, Algorithm PO (Marsaglia and Bray (1964)), about 145 μsec has to be added. The polar algorithm is an improved version of the Box–Muller method which does not require trigonometric functions. When the normal variables are generated using the convenient algorithm[1] of Marsaglia and Bray

[1] There is a misprint in the description of this algorithm in Atkinson and Pearce (1976 p. 438) where the last line of the algorithm should read "until either $|S|$ or $|T| > 3$". (In the printed version the absolute signs have been omitted.)

(1964) an average of an additional 104 μsec is required, provided that, in both cases, the generalized inverse Gaussian and normal random variables are generated in the same subroutine.

In Table 5(b) timings are given for the direct rejection algorithm HYP. For values of $\psi = \alpha^2 - \beta^2$ below the range 15–20, this rejection algorithm is faster than either mixing method and is faster than the mixing method using Algorithm PO up to a value of 36 for ψ.

TABLE 5

Average time, in μsec on the Cyber 174, to generate one hyperbolic random variable.

(a) Generalized inverse Gaussian: Algorithm GIG without logarithm

	α						
β	0.1	0.2	0.5	1	2	5	10
0	193	191	192	197	209	218	230
0.2α	193	192	191	197	206	218	227
0.8α	194	192	191	191	199	214	220

(b) Hyperbolic: rejection algorithm HYP

β							
0	218	218	222	236	263	339	438
0.2α	220	219	223	239	263	339	440
0.8α	224	222	222	229	245	295	366

These results show that either mixing or rejection leads to a satisfactory algorithm for sampling the hyperbolic distribution provided that α is less than 10. For larger values of α the mixing method is preferable, with little to be gained from the extra length of the convenient algorithm for generating normal random variables. There remains the possibility of other envelopes for the rejection method, in particular an algorithm similar to GIG in which the values of the parameters are found by crude searches.

8. The gamma distribution. When $\chi = 0$ and $\psi = 2$, the generalized inverse Gaussian distribution reduces to the gamma distribution

$$f(x) = \frac{x^{\lambda-1}e^{-x}}{\Gamma(\lambda)} \qquad (x \geqq 0).$$

For $\lambda \geqq 1$ Algorithm GIG of § 3 therefore provides yet another algorithm for generating gamma random variables. In order to calibrate the timings given in Tables 2, 4 and 5 we compare this new gamma algorithm with some currently in the literature.

Algorithm GIG can be improved in at least three ways for the specific purpose of generating gamma random variables. One, following Cheng (1977), is to use the concavity of $\log x$ to provide a quick test for rejection. Unfortunately, what is required is a quick test for acceptance, and this procedure makes the algorithm about 5 μsec slower.

A second and more effective way of improving the algorithm is to use the optimum value of p in (12), which can be found analytically and is $p^* = \{\sqrt{1+4t} - 1\}/2t$ where $t = \lambda - 1$, the mode (see Atkinson (1977) for this derivation in the context of the gamma distribution). The third improvement is to use the logarithmic form of the rejection condition, as in Algorithm HYP, combined with exponential random variables generated by von Neumann's method, Algorithm NE.

The comparisons in Table 6 show these modifications to have surprisingly little effect. For the exponential form of rejection to be an improvement it is necessary that the exponential variables be generated in the subroutine where they are used. The importance of these results is that the negligible improvement from using the optimum value p^* indicates that Algorithm GIG sacrifices little by the crudeness of the search for optimum parameter values.

TABLE 6

Average time, in μsec on the Cyber 174, *to generate one gamma random variable.*

	λ										
Algorithm	1.5	2	2.5	3	3.5	4	4.5	5	7	10	20
GIG	260	269	277	282	286	291	291	290	302	304	306
GIG with p^*	257	268	273	280	282	289	291	290	298	302	307
GIG with p^*, exponential rejection	247	256	265	268	273	276	275	280	283	288	290
Ahrens and Dieter GO	–	–	–	274	271	266	260	260	245	231	205
Cheng GB	290	277	269	268	260	261	257	256	253	253	250
Atkinson	240	247	254	257	262	267	271	275	286	306	352

The remainder of Table 6 gives the results of timing comparisons with three gamma algorithms. For large λ the fastest is Algorithm GO (Ahrens and Dieter (1974)), a rejection algorithm with first stage sampling from the normal distribution, in this case the convenient method of Marsaglia and Bray. This algorithm is valid only for $\lambda > 2.53 \cdots$. One algorithm for all $\lambda \geqq 1$ is that due to Cheng (1977) which is a rejection algorithm with a log-logistic envelope. A third algorithm (Atkinson (1977)) complements GO and is efficient in the region $1 \leqq \lambda \leqq 4$. This is similar to Algorithm GIG above the mode, but below the mode the envelope is a constant.

The results in Table 6 lead to conclusions similar to those of Atkinson (1977). For $\lambda \leqq 3$ Atkinson's algorithm is fastest; in the region $\lambda < 3 \leqq 5$ Cheng's is fastest; above this, GO predominates. But the important result for the present investigation is that the timings for GIG are comparable with those for the best gamma algorithms. For $\lambda = 1.5$ the unimproved version of GIG is only 8% slower than Atkinson's algorithm and for $\lambda = 5$ the same algorithm is 13% slower than Cheng's algorithm.

To some extent the choice of gamma algorithms for comparison was arbitrary. Similar timings can be expected from the algorithms of Best (1978) and Tadikamalla (1978). A further algorithm, based on the general method of Kinderman and Monahan (1977), is described by Cheng and Feast (1979). For $\lambda = 10$ their composite algorithm is about 5% faster than Cheng's algorithm, with the advantage that the time does not increase appreciably as λ decreases.

9. Multivariate distributions. The distributions described so far are all univariate. Barndorff-Nielsen (1978, § 7) has described a multivariate generalized hyperbolic distribution. This is obtained by mixing as in § 5 except that an r-dimensional normal distribution is mixed by $N^{-}(\lambda, \chi, \psi)$ such that the mean ξ and variance Σ are related by $\xi = \mu + \beta\Sigma$ with μ and β vector parameters. The variance $\Sigma = \sigma^2\Delta$, where Δ is $r \times r$ with determinant $|\Delta| = 1$ and σ^2 has the generalized inverse Gaussian distribution.

To simulate this distribution, let $\Delta = LL^T$, where L is a lower-triangular matrix. Then if Z is an $r \times 1$ vector of independent standard normal random variables, the vector random variable LZ has an r-dimensional normal distribution with covariance matrix Δ. The algorithm follows.

```
Set $\Delta = LL^T$ and $\kappa = \beta\Delta$.
Sample $X$ from $N^-(\lambda, \chi, \psi)$ and set $\sigma^2 = X$.
Sample $Z$ from $N(0, I_r)$.
Return $Y = \mu + \sigma^2\kappa + \sigma ZL$.
```

It remains to calculate L, and the Choleski decomposition can be used (Seber (1977, pp. 304–6)). The NAG algorithm FO1BQF yields the decomposition $\Delta = TDT^T$ where T is a lower-triangular matrix with unit diagonal and $D = \text{diag}(d_{11}, \cdots, d_{rr})$. If $D^{1/2} = \text{diag}(d_{11}^{1/2}, \cdots, d_{rr}^{1/2})$, the required matrix L equals $TD^{1/2}$. The algorithm given by Fishman (1978, p. 465, Algorithm LTM) does not include a check that the pivots are nonzero.

10. Discussion. The timings of the algorithms described in this paper show that they are satisfactory when compared with those for the related gamma distribution. However, this comparison serves as a reminder that many potential enveloping distributions, other than the exponentials used here, will give broadly similar results and can indeed be expected to give better results for at least some special cases. One such case is the generalized inverse Gaussian distribution when ω is small and λ is in the range 0 to 0.5. Another special case is the hyperbolic distribution for large values of $\psi = \alpha^2 - \beta^2$. One obvious candidate algorithm is an analogue of Algorithm GIG in which the parameters for a two-part exponential envelope would be found by grid searches.

It appears that such improvements are unlikely to be needed for general use. However, they may be necessary in applications of the hyperbolic distribution to studies of turbulence, where tens of thousands of samples from heavy-tailed distributions are needed (Petersen (1976), Barndorff-Nielsen (1979)).

An extension is to simulation of the compound Poisson distribution introduced by Sichel (1974), (1975) in which the Poisson parameter is mixed by the generalized inverse Gaussian distribution. Algorithms for sampling the Poisson distribution with variable parameter are described by Fishman (1976) and Atkinson (1979a).

Listings of Fortran versions of the algorithms for the generalized inverse Gaussian and hyperbolic distributions are given in an appendix to Atkinson (1979b), copies of which are available from the author.

Acknowledgments. I am grateful to Professor Ole Barndorff-Nielsen for the opportunity of working at Aarhus University, where Algorithm GIG was developed. During my visit I profited greatly from the unfailing helpfulness of Geert Schou, whom I thank.

REFERENCES

J. H. Ahrens and U. Dieter (1972), *Computer methods for sampling from the exponential and normal distributions*, Comm. ACM, 15, pp. 873–882.

——— (1974), *Computer methods for sampling from gamma, beta, Poisson and binomial distributions*, Computing (Vienna), 12, pp. 223–246.

A. C. Atkinson (1977), *An easily programmed algorithm for generating gamma random variables*, J. Roy. Statist. Soc. Ser. A, 140, pp. 232–234.

——— (1979a), *Recent developments in the computer generation of Poisson random variables*, Appl. Statist. 28, pp. 260–263.

——— (1979b), *The simulation of generalised inverse Gaussian, generalised hyperbolic, gamma and related random variables*, Research report 52, Dept. Theoretical Statistics, Aarhus Univ., Aarhus, Denmark.

A. C. ATKINSON AND M. C. PEARCE (1976), *The computer generation of beta, gamma and normal random variables* (*with discussion*), J. Roy. Statist. Soc. Ser. A, 139, pp. 431–461.

A. C. ATKINSON AND J. WHITTAKER (1976), *A switching algorithm for the generation of beta random variables with at least one parameter less than* 1, J. Roy. Statist. Soc. Ser. A, 139, pp. 462–467.

O. BARNDORFF-NIELSEN (1977), *Exponentially decreasing distributions for the logarithm of particle size*, Proc. Roy. Soc. London Ser. A, 353, pp. 401–419.

——— (1978), *Hyperbolic distributions and distributions on hyperbolae*, Scand. J. Statist., 5, pp. 151–157.

——— (1979), *Models for non-Gaussian variation with applications to turbulence*, Proc. Roy. Soc. London Ser. A, 368, pp. 501–520.

D. J. BEST (1978), *Letter to the editor*, Appl. Statist., 27, p. 181.

P. BLÆSILD (1978), *The shape of the generalized inverse Gaussian and hyperbolic distributions*, Research report 37, Dept. Theoretical Statistics, Aarhus Univ., Aarhus, Denmark.

R. C. H. CHENG (1977), *The generation of gamma variables with non-integral shape parameter*, Appl. Statist., 26, pp. 71–75.

R. C. H. CHENG AND G. M. FEAST (1979), *Some simple gamma variate generators*, Appl. Statist., 28, pp. 290–295.

G. S. FISHMAN (1976), *Sampling from the Poisson distribution on a computer*, Computing (Vienna), 17, pp. 147–156.

——— (1978), *Principles of Discrete Event Simulation*, Wiley-Interscience, New York.

A. J. KINDERMAN AND J. F. MONAHAN (1977), *Computer generation of random variables using the ratio of uniform deviates*, ACM Trans. Math. Software, 3, pp. 257–260.

G. MARSAGLIA AND T. A. BRAY (1964), *A convenient method for generating normal variables*, SIAM Rev., 6, pp. 260–264.

J. R. MICHAEL, W. R. SCHUCANY AND R. W. HAAS (1976), *Generating random variables using transformations with multiple roots*, Amer. Statist., 30, pp. 88–90.

W. J. PADGETT (1978), *Comment on inverse Gaussian random number generation*, J. Statist. Comput. Simulation, 8, pp. 78–79.

E. L. PETERSEN (1976), *A model for the simulation of atmospheric turbulence*, J. Appl. Meteorol., 15, pp. 571–587.

G. A. F. SEBER (1977), *Linear Regression Analysis*, John Wiley, New York.

H. S. SICHEL (1974), *On a distribution representing sentence length in written prose*, J. Roy. Statist. Soc. Ser. A., 137, pp. 25–34.

——— (1975), *On a distribution law for word frequencies*, J. Amer. Statist. Assoc., 70, pp. 542–547.

P. R. TADIKAMALLA (1978), *Computer generation of gamma random variables*—II, Comm. ACM, 21, pp. 925–928.

M. C. K. TWEEDIE (1957), *Statistical properties of inverse Gaussian distributions*. I, Ann. Math. Statist., 28, pp. 362–377.