SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0001 $01.00/0

EFFICIENT IMPLEMENTATION OF A CLASS OF PRECONDITIONED CONJUGATE GRADIENT METHODS*

STANLEY C. EISENSTAT†

Abstract. The preconditioned conjugate gradient (PCG) method is an effective means for solving systems of linear equations where the coefficient matrix is symmetric and positive definite. The incomplete LDL^t factorizations are a widely used class of preconditionings, including the SSOR, Dupont-Kendall-Rachford, generalized SSOR, ICCG(0), and MICCG(0) preconditionings. The efficient implementation of PCG with a preconditioning from this class is discussed.

Key words. incomplete factorization, preconditioned conjugate gradient methods, preconditioning

1. Introduction.

Consider the system of N linear equations

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

where the coefficient matrix A is symmetric and positive definite. When A is large and sparse, the preconditioned conjugate gradient (PCG) method is an effective means for solving (1) [2], [4], [5], [9], [13]. Given an initial guess x_0, we generate a sequence $\{x_k\}$ of approximations to the solution x as follows:

(2a) $r_0 = b - Ax_0$

(2b) Solve $Mr'_0 = r_0$ and set $p_0 = r'_0$

FOR $k = 0$ STEP 1 UNTIL Convergence DO

(2c) $a_k = (r_k, r'_k)/(p_k, Ap_k)$

(2d) $x_{k+1} = x_k + a_k p_k$

(2e) $r_{k+1} = r_k - a_k A p_k$

(2f) Solve $Mr'_{k+1} = r_{k+1}$

(2g) $b_k = (r_{k+1}, r'_{k+1})/(r_k, r'_k)$

(2h) $p_{k+1} = r'_{k+1} + b_k p_k$.

The effect of the preconditioning matrix M is to increase the rate of convergence of the basic conjugate gradient method of Hestenes and Stiefel [11]. The number of multiply-adds per iteration is just $5N$, plus the number required to form Ap_k, plus the number required to solve $Mr'_k = r_k$.

One widely used class of preconditionings are the incomplete LDL^t factorizations

$$M = (\tilde{D} + L)\tilde{D}^{-1}(\tilde{D} + L)^t, \tag{3}$$

where $A \equiv L + D + L^t$, L is strictly lower triangular and D and $\tilde{D}$ are positive diagonal. This class includes the SSOR [9], Dupont-Kendall-Rachford [7], generalized SSOR [1], ICCG(0) [13] and MICCG(0) [10] preconditionings. If we let NZ(A) denote the number of nonzero entries in the matrix A, a straightforward implementation of

* Received by the editors October 30, 1980. This research was supported in part by the U.S. Office of Naval Research under grant N00014-76-C-0277.

† Research Center for Scientific Computation, Department of Computer Science, Yale University, New Haven, Connecticut 06520.

PCG with a preconditioning from this class[1] would require $6N + 2\mathrm{NZ}(A)$ multiply-adds per iteration.[2]

In this brief note, we show how to reduce the work to $8N + \mathrm{NZ}(A)$ multiply-adds, asymptotically half as many as the straightforward implementation.[3] We give details in § 2 and consider some generalizations in § 3.

2. Implementation. The linear system (1) can be restated in the form

$$[(\tilde{D}+L)^{-1}A(\tilde{D}+L)^{-t}][(\tilde{D}+L)^{t}x] = [(\tilde{D}+L)^{-1}b] \tag{4}$$

or

$$\hat{A}\hat{x} = \hat{b}. \tag{5}$$

But applying PCG to (1) with $M = (\tilde{D}+L)\tilde{D}^{-1}(\tilde{D}+L^{t})$ is equivalent to applying PCG to (5) with $\hat{M} = \tilde{D}^{-1}$ and setting $x = (\tilde{D}+L)^{-t}\hat{x}$.[4] If we update x instead of $\hat{x}$ at each iteration, algorithm (2) becomes:

(6a) $\hat{r}_0 = (\tilde{D}+L)^{-1}(\hat{b} - \hat{A}x_0)$

(6b) $\hat{p}_0 = \hat{r}'_0 = \tilde{D}\hat{r}_0$

FOR $k = 0$ STEP 1 UNTIL Convergence DO

(6c) $\hat{a}_k = (\hat{r}_k, \hat{r}'_k)/(\hat{p}_k, \hat{A}\hat{p}_k)$

(6d) $x_{k+1} = x_k + \hat{a}_k(\tilde{D}+L)^{-t}\hat{p}_k$

(6e) $\hat{r}_{k+1} = \hat{r}_k - \hat{a}_k\hat{A}\hat{p}_k$

(6g) Compute $\hat{r}'_{k+1} = \tilde{D}\hat{r}_{k+1}$

(6g) $\hat{b}_k = (\hat{r}_{k+1}, \hat{r}'_{k+1})/(\hat{r}_k, \hat{r}'_k)$

(6h) $\hat{p}_{k+1} = \hat{r}'_{k+1} + \hat{b}_k\hat{p}_k$

$\hat{A}\hat{p}_k$ can be computed efficiently by taking advantage of the following identity:

$$\begin{aligned}\hat{A}\hat{p}_k &= (\tilde{D}+L)^{-1}[(\tilde{D}+L)+(\tilde{D}+L)^{t}-(2\tilde{D}-D)](\tilde{D}+L)^{-t}\hat{p}_k \\ &= (\tilde{D}+L)^{-t}\hat{p}_k + (\tilde{D}+L)^{-1}[\hat{p}_k - K(\tilde{D}+L)^{-t}\hat{p}_k],\end{aligned} \tag{7}$$

where $K \equiv 2\tilde{D} - D$. Thus

$$\hat{t}_k = (\tilde{D}+L)^{-t}\hat{p}_k, \tag{8a}$$

$$\hat{A}\hat{p}_k = \hat{t}_k + (\tilde{D}+L)^{-1}(\hat{p}_k - K\hat{t}_k), \tag{8b}$$

which requires $2N + \mathrm{NZ}(A)$ multiply-adds. $\hat{t}_k$ can also be used to update x_k in (6d), so that the total cost for each PCG iteration is just $8N + \mathrm{NZ}(A)$ multiply-adds[5] (plus

---

[1] Writing M as $(\tilde{D}+L)(I+\tilde{D}^{-1}L^{t})$, we solve $Mr'_k = r_k$ by solving the triangular systems

$$(\tilde{D}+L)t_k = r_k, \qquad (I+\tilde{D}^{-1}L^{t})r'_k = t_k.$$

[2] $2N$ (respectively, N) multiply-adds can be saved by symmetrically scaling the problem to make $\tilde{D} = I$ (respectively, $D = I$).

[3] A similar speedup for pairs of linear iterative methods is given in [6].

[4] Both are equivalent to applying the basic conjugate gradient method to the preconditioned system

$$\bar{A}\bar{x} \equiv [\tilde{D}^{1/2}(\tilde{D}+L)^{-1}A(\tilde{D}+L)^{-t}\tilde{D}^{1/2}][\tilde{D}^{-1/2}(\tilde{D}+L)^{t}x] = [\tilde{D}^{1/2}(\tilde{D}+L)^{-1}b] \equiv \bar{b}$$

(see [4, pp. 58–59]).

[5] Again, $3N$ multiply-adds can be saved by symmetrically scaling the problem so that $\tilde{D} = I$.

an additional N words of storage), versus $6N+2\mathrm{NZ}(A)$ for the straightforward implementation.

3. Generalizations. The approach presented in § 2 extends immediately to preconditionings of the form

$$M=(\tilde{D}+L)\tilde{S}^{-1}(\tilde{D}+L)^t, \tag{9}$$

where $\tilde{S}$ is positive diagonal. Moreover, if we take $K\equiv\tilde{D}+\tilde{D}^t-D$ in (7) and (8), then $\tilde{D}$ need not be diagonal or even symmetric. In this case, $\tilde{D}$ would reflect changes to both the diagonal and offdiagonal entries of A in generating an incomplete factorization. If we assume that only the nonzero entries of A are changed, i.e., that $(K)_{ij}$ is nonzero only if $(A)_{ij}$ is nonzero, then the operation count is $7N+\mathrm{NZ}(A)+\mathrm{NZ}(K)$.

Another application is to preconditioning nonsymmetric systems. Let

$$M=(\tilde{D}+L)\tilde{S}^{-1}(\tilde{D}+U) \tag{10}$$

be an incomplete LDU factorization of a nonsymmetric matrix A, where $A\equiv L+D+U$, L (respectively, U) is strictly lower (respectively, upper) triangular and D and $\tilde{S}$ are diagonal. Then a number of authors have proposed solving the linear system $Ax=b$ by using the conjugate gradient method to solve the normal equations for one of the preconditioned systems

$$\hat{A}_1\hat{x}\equiv[\tilde{S}(\tilde{D}+L)^{-1}A(\tilde{D}+U)^{-1}][(\tilde{D}+U)x]=[\tilde{S}(\tilde{D}+L)^{-1}b]\equiv\hat{b} \tag{11a}$$

(see [12]) and

$$\hat{A}_2x\equiv[(\tilde{D}+U)^{-1}\tilde{S}(\tilde{D}+L)^{-1}A]x=[(\tilde{D}+U)^{-1}\tilde{S}(\tilde{D}+L)^{-1}b]\equiv\hat{b} \tag{11b}$$

(see [14], [3]). $\hat{A}_2\hat{p}$ can be computed as

$$\hat{A}_2\hat{p}=(\tilde{D}+U)^{-1}\tilde{S}[\hat{p}+(\tilde{D}+L)^{-1}(D+U-\tilde{D})\hat{p}] \tag{12}$$

in $4N+\mathrm{NZ}(L)+2\mathrm{NZ}(U)$ multiply-adds, whereas $\hat{A}_1\hat{p}$ can be computed as

$$\hat{t}=(\tilde{D}+U)^{-1}\hat{p}, \tag{13a}$$

$$\hat{A}_1\hat{p}=\tilde{S}[\hat{t}+(\tilde{D}+L)^{-1}(\hat{p}-(2\tilde{D}-D)\hat{t})] \tag{13b}$$

in $4N+\mathrm{NZ}(L)+\mathrm{NZ}(U)$ multiply-adds. Thus the preconditioning (11a) would be more efficient per iteration, although more iterations might be required to achieve comparable accuracy [14].[6]

REFERENCES

[1] O. AXELSSON, *A generalized* SSOR *method*, BIT, 13 (1972), pp. 443–467.

[2] ———, *On preconditioning and convergence acceleration in sparse matrix problems*, Technical Report 74-10, CERN, May, 1974.

[3] ———, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1–16.

[4] RATI CHANDRA, *Conjugate Gradient Methods for Partial Differential Equations*, PhD thesis, Department of Computer Science, Yale University, New Haven, CT, 1978.

[5] PAUL CONCUS, GENE H. GOLUB AND DIANNE P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, James R. Bunch and Donald J. Rose, eds., Academic Press, New York, 1976, pp. 309–332.

[6] The same would be true if a generalized conjugate residual method such as Orthomin [15], [8] were used to solve (11a) or (11b).

[6] V. CONRAD AND Y. WALLACH, *Alternating methods for sets of linear equations*, Numer. Math., 32 (1979), pp. 105–108.

[7] TODD DUPONT, RICHARD P. KENDALL AND H. H. RACHFORD JR., *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal., 5 (1968), pp. 559–573.

[8] S. C. EISENSTAT, H. ELMAN, M. H. SCHULTZ AND A. H. SHERMAN, *Solving approximations to the convection diffusion equation*, in Society of Petroleum Engineers of AIME, Proceedings of the Fifth Symposium on Reservoir Simulation, February, 1979, pp. 127–132.

[9] D. J. EVANS, *The analysis and application of sparse matrix algorithms in the finite element method*, in The Mathematics of Finite Elements and Applications, J. R. Whiteman, ed., Academic Press, New York, 1973, pp. 427–447.

[10] IVAR GUSTAFSSON, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[11] MAGNUS R. HESTENES AND EDUARD STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. National Bureau of Standards, 49 (1952), pp. 409–436.

[12] DAVID S. KERSHAW, *The incomplete Cholesky-conjugate gradient method for the solution of systems of linear equations*, J. Comput. Phys., 26 (1978), pp. 43–65.

[13] 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.

[14] M. PETRAVIC AND G. KUO-PETRAVIC, *An ILUCG algorithm which minimizes in the Euclidean norm*, J. Comput. Phys., 32 (1979), pp. 263–269.

[15] P. K. W. VINSOME, *Orthomin, an iterative method for solving sparse sets of simultaneous linear equations*, in Society of Petroleum Engineers of AIME, Proceedings of the Fourth Symposium on Reservoir Simulation, February, 1976, pp. 150–159.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0002 $01.00/0

SPLINE INTERPOLATION AND SMOOTHING ON THE SPHERE*

GRACE WAHBA†

Abstract. We extend the notion of periodic polynomial splines on the circle and thin plate splines on Euclidean d-space to splines on the sphere which are invariant under arbitrary rotations of the coordinate system. We solve the following problem: Find $u \in \mathcal{H}_m(S)$, a suitably defined reproducing kernel (Sobolev) space on the sphere S to, A) minimize $J_m(u)$ subject to $u(P_i) = z_i$, $i = 1, 2, \cdots, n$, and B) minimize

$$\frac{1}{n}\sum_{j=1}^{n}(u(P_i)-z_i)^2+\lambda J_m(u),$$

where

$$J_m(u)=\int_0^{2\pi}\int_0^{\pi}(\Delta^{m/2}u(\theta,\phi))^2\sin\theta\,d\theta\,d\phi, \qquad m \text{ even}$$

$$=\int_0^{2\pi}\int_0^{\pi}\left\{\frac{(\Delta^{(m-1)/2}u)_\phi^2}{\sin^2\theta}+(\Delta^{(m-1)/2}u)_\theta^2\right\}\sin\theta\,d\theta\,d\phi, \qquad m \text{ odd}.$$

Here Δ is the Laplace–Beltrami operator on the sphere and $J_m(u)$ is the natural analogue on the sphere, of the quadratic functional $\int_0^{2\pi}(u^{(m)}(\theta))^2\,d\theta$ on the circle, which appears in the definition of periodic polynomial splines. $J_m(u)$ may also be considered to be the analogue of

$$\sum_{j=0}^{m}\binom{m}{j}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\left(\frac{\partial^m u}{\partial x^j\,\partial y^{m-j}}\right)^2 dx\,dy$$

appearing in the definition of thin plate splines on the plane. The solution splines are obtained in the form of infinite series, which do not appear to be convenient for certain kinds of computation. We then replace J_m in A) and B) by a quadratic functional Q_m which is topologically equivalent to J_m on $\mathcal{H}_m(S)$ and obtain closed form solutions to the modified problems which are suitable for numerical calculation, thus providing practical pseudo-spline solutions to interpolation and smoothing problems on the sphere. Convergence rates of the splines and pseudo-splines will be the same. A number of results established or conjectured for polynomial and thin plate splines can be extended to the splines and pseudo-splines constructed here.

Key words. splines on the sphere, spherical harmonics, smoothing on the sphere

1. Introduction.

This work is motivated by the following problem. The 500 millibar height (the height above sea level at which the pressure is 500 millibars) is measured (with error) at a large number n of weather stations distributed around the world. It is desired to find a smooth function $u = u(\theta, \phi)$ defined on the surface of the earth (θ = latitude, ϕ = longitude) which is an estimate of the 500 millibar height at position (θ, ϕ). There are many ways that this can be done. In this paper we develop what appears to be the natural generalization to the sphere of periodic interpolating and smoothing splines on the circle (see Golomb [12], Wahba [27]) and thin plate splines on Euclidean d-space (see Duchon [6], Meinguet [18], Wahba [28]).

To obtain a periodic interpolating or smoothing spline on the circle C one seeks the solution to one of the problems: Find $u \in \mathcal{H}_m(C)$ to minimize

A) $\quad J_m(u)$ subject to $u(t_i) = z_i, \quad i = 1, 2, \cdots, n$

or

B) $$\frac{1}{n}\sum_{i=1}^{n}(u(t_i)-z_i)^2+\lambda J_m(u).$$

* Received by the editors November 29, 1979. This research was supported by the U.S. Army Research Office under contract DAAG29-77-0209.

† Department of Statistics, University of Wisconsin, Madison, Wisconsin 53706.

Here

$$J_m(u) = \int_0^{2\pi} (u^{(m)}(t))^2\, dt, \tag{1.1}$$

$t_i \in [0, 2\pi]$ and $\mathscr{H}_m(C) = \{u: u, u', \cdots, u^{(m-1)}$ abs. cont., $u^{(m)} \in \mathscr{L}_2[0, 2\pi]$, $u^{(j)}(0) = u^{(j)}(2\pi), j = 0, 1, \cdots, m-1\}$. To find a thin plate interpolating or smoothing spline on Euclidean d-space E^d, one finds $u \in \mathscr{H}_m(E^d)$ to minimize A) or B) above, where now $t_i = (x_{1i}, x_{2i}, \cdots, x_{di}) \in E^d$ and

$$J_m(u) = \sum_{i_1, i_2, \cdots, i_m = 1}^{d} \int_{E^d} \left(\frac{\partial^m u}{\partial x_{i_1} \partial x_{i_2}, \cdots, \partial x_{i_m}} \right)^2 dx_1\, dx_2 \cdots, dx_d. \tag{1.2}$$

$\mathscr{H}_m(E^d)$is defined in Meinguet [18]. To obtain a thin plate interpolating or smoothing spline it is necessary that $2m - d > 0$, since otherwise the evaluation functionals $u \to u(t_i)$ will not be bounded in $\mathscr{H}_m(E^d)$ and thus will not have representers which are used in the construction of the solution.

Duchon has called the solutions to problems involving J_m in Euclidean d-space thin plate splines, because, in two dimensions with $m = 2$,

$$J_2(u) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \left[\left(\frac{\partial^2 u}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 u}{\partial x_1\, \partial x_2} \right)^2 + \left(\frac{\partial^2 u}{\partial x_2^2} \right)^2 \right] dx_1\, dx_2$$

is the bending energy of a thin plate. Interpolating and smoothing thin plate splines have been computed in a number of examples by Franke, Utreras, Wahba, Wahba and Wendelberger, and Wendelberger for data given in the form of an analytic function which is evaluated by computer at $t_1, t_2, \cdots, t_n$ [9], for function data with simulated errors [26], [28], [30] and for measured 500 millibar height data [31], with very satisfying results. Fisher and Jerome in a classic early paper [8] answered some important questions concerning interpolation problems on Ω a bounded set in R^d associated with general elliptic operators.

For the analysis of meteorological data, we would like to be able to compute smoothing splines on the sphere. To motivate the definition of J_m for the sphere, we first take a look at the Sobolev spaces $\mathscr{H}_m(C)$ of periodic functions on the circle. $\mathscr{H}_m(C)$ is the collection of square integrable functions u on $[0, 2\pi]$ which satisfy

$$a_0^2 + \sum_{\nu=1}^{\infty} \nu^{2m} a_\nu^2 + \sum_{\nu=1}^{\infty} \nu^{2m} b_\nu^2 < \infty, \tag{1.3}$$

where

$$a_\nu = \frac{1}{\sqrt{\pi}} \int_0^{2\pi} \cos \nu\theta u(\theta)\, d\theta, \qquad \nu = 0, 1, \cdots,$$

$$b_\nu = \frac{1}{\sqrt{\pi}} \int_0^{2\pi} \sin \nu\theta u(\theta)\, d\theta, \qquad \nu = 1, 2, \cdots.$$

We have

$$J_m(u) \doteq \int_0^{2\pi} (u^{(m)}(\theta))^2\, d\theta = \sum_{\nu=1}^{\infty} \nu^{2m} a_\nu^2 + \sum_{\nu=1}^{\infty} \nu^{2m} b_\nu^2 \tag{1.4}$$

for $u \in \mathscr{H}_m(C)$. $\mathscr{H}_m(C)$ is thus a space of (periodic, square integrable) functions whose Fourier coefficients $\{a_\nu, b_\nu\}$ decay sufficiently fast to satisfy (1.3). The functions $\{\cos \nu\theta, \sin \nu\theta\}$ are the (periodic) eigenfunctions of the operator D^{2m} ($D^{2m} u = u^{(2m)}$)

which appears when $J_m(u)$ of (1.1) is integrated by parts and u is sufficiently smooth and periodic:

$$J_m(u) = \int_0^{2\pi} u \cdot D^{2m} u \, d\theta.$$

If one formally integrates (1.2) by parts, and u is sufficiently smooth and decreases to 0 at infinity, then one obtains

$$J_m(u) = (-1)^m \int \cdots \int_{E^d} u \cdot \tilde{\Delta}^m u \, dx_1 \cdots dx_d,$$

where $\tilde{\Delta} u$ is the Laplacian,

$$\tilde{\Delta} u = \frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \cdots + \frac{\partial^2 u}{\partial x_d^2}.$$

The analogue of $\tilde{\Delta}$ on the sphere is the Laplace–Beltrami operator defined by

$$\Delta u = \frac{1}{\sin^2 \theta} u_{\phi\phi} + \frac{1}{\sin \theta} (\sin \theta u_\theta)_\theta,$$

where $\theta \in [0, \pi]$ is latitude and $\phi \in [0, 2\pi]$ is longitude. This is the restriction of the Laplacian in 3-space to the surface of the sphere; see Courant and Hilbert [3, Chapt. V, VII], and Whittaker and Watson [32]. The role of the eigenfunctions $\{(1/\sqrt{\pi}) \cos \nu\theta, (1/\sqrt{\pi}) \sin \nu\theta\}$ in $\mathcal{H}_m(C)$ is played in $\mathcal{H}_m(S)$, (S is the sphere) by the normalized spherical harmonics $\{Y_\nu^k(\theta, \phi)\}_{\nu=0}^{\infty}{}_{k=-\nu}^{\nu}$ (defined in § 2), which are the (periodic) eigenfunctions of the Laplace–Beltrami operator Δ^m, and the role of the eigenvalues $\{\nu^{2m}, \nu^{2m}\}_{\nu=1}^{\infty}$ of D^{2m} is played by the eigenvalues of Δ^m. Δ^m has the single square integrable periodic eigenfunction $Y_0^0(\theta, \phi) = 1$, corresponding to the eigenvalue 0. We now define $\mathcal{H}_m(S)$ as the space of square integrable functions u on S with

$$|u_{00}| < \infty, \qquad \sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} \frac{u_{\nu k}^2}{\lambda_{\nu k}} < \infty, \tag{1.5}$$

where

$$u_{\nu k} = \int_S Y_\nu^k(P) u(P) \, dP \tag{1.6}$$

and $\{\lambda_{\nu k}^{-1}\}$, $(\lambda_{\nu k}^{-1} = [\nu(\nu+1)]^k)$ are the eigenvalues of Δ^m corresponding to $\{Y_\nu^k\}$. $\mathcal{H}_m(S)$ is thus a space of square integrable functions whose Fourier Bessel coefficients with respect to the spherical harmonics decay sufficiently fast to satisfy (1.5). Let J_m be defined by

$$\begin{aligned} J_m(u) &= \int_0^{2\pi} \int_0^{\pi} (\Delta^{m/2} u)^2 \sin \theta \, d\theta \, d\phi, \qquad m \text{ even}, \\ &= \int_0^{2\pi} \int_0^{\pi} \frac{(\Delta^{(m-1)/2} u)_\phi^2}{\sin^2 \theta} + (\Delta^{(m-1)/2} u)_\theta^2 \sin \theta \, d\theta \, d\phi, \qquad m \text{ odd}. \end{aligned} \tag{1.7}$$

It is not hard to show that, for $u \in \mathcal{H}_m(S)$,

$$J_m(u) = \sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} \frac{u_{\nu k}^2}{\lambda_{\nu k}}. \tag{1.8}$$

A number of results which are known or conjectured for polynomial splines on the circle and thin plate splines on E^d will carry over to the thin plate splines and pseudo-splines on the sphere. They include optimality properties of the generalized cross-validation estimate of λ and m [4], [25], convergence rates for smoothing splines with noisy data, properties of associated orthogonal series density estimates, and interpretation of interpolating and smoothing splines as Bayes estimates when u is modeled as the solution to the stochastic differential equation $\Delta^{m/2} u =$ "white noise." Details and further references may be found in Wahba [29]. The corresponding splines when u is modeled as a general stationary autoregressive moving average process on S are also given in Wahba [29], as well as possible models encompassing nonstationarity (anisotropy). The reader interested in meteorological applications may be interested in consulting Stanford [24], where ensembles of $\{u_{\nu k}^2\}$ defined in (1.6) have been computed from measured satellite radiance data and are suggestive of an appropriate choice of m in certain meteorological applications. The results here also show that variational techniques for meteorological data analysis similar to these pioneered by Sasaki [21] and others can be carried out on the sphere; see also Wahba and Wendelberger [30]. Part of the importance of the present work is in its potential applicability to important meteorological problems, some of which are mentioned near the end of § 2.

We seek $u \in \mathscr{H}_m(S)$ to minimize

A) $\quad J_m(u)$ subject to $u(P_i) = z_i, \qquad i = 1, 2, \cdots, n,$

B) $$\frac{1}{n}\sum_{i=1}^{n} (u(P_i) - z_i)^2 + \lambda J_m(u),$$

where $P_i \in S$, and J_m is defined by (1.7).

We cannot solve these problems for $m = 1$ for the same reason they cannot be solved in E^d for $2m - d \leqq 0$, that is, because the evaluation functions are not continuous in $\mathscr{H}_1(S)$, that is, $\mathscr{H}_1(S)$ is not a reproducing kernel space. However, for $m = 2, 3, \cdots$ we will give the explicit solution to those two problems, which we will call thin plate splines on the sphere. It is actually not hard to obtain the solutions, since we can construct a reproducing kernel for $\mathscr{H}_m(S)$, with $J_m(\cdot)$ as a seminorm, from the well-known eigenfunctions and eigenvalues of the Laplace–Beltrami operator. Given the reproducing kernel [2], the solutions to such problems are well known, and in fact problems A) and B) can be solved with $u(P_i)$ replaced by $L_i u$, where L_i is any continuous linear functional on $\mathscr{H}_m(S)$. See, for example, Kimeldorf and Wahba [17] and references cited there.

Unfortunately we only know the aforementioned reproducing kernels in the form of infinite series. It appears that no closed form expression exists which is convenient for computational purposes. Wendelberger [31] has computed the reproducing kernels given below for m from 2 to 10 by evaluating the infinite series, and it is likely that satisfactory computational procedures for interpolation and smoothing splines on the sphere can be developed based on the infinite series. However, for general continuous linear functionals it may be important to have a reproducing kernel in closed form. Furthermore, to compute certain functionals of the solution, for example, derivatives, it may be important to have a closed form solution. For this reason we suggest replacing $J_m(\cdot)$ by another quadratic functional $Q_m(\cdot)$ which is topologically equivalent to $J_m(\cdot)$ in the sense that there exist α and β, $0 < \alpha < \beta < \infty$ such that

$$\alpha J_m(u) \leqq Q_m(u) \leqq \beta J_m(u), \quad \text{all } u \in \mathscr{H}_m(S).$$

We give the reproducing kernel associated with Q_m in closed form. It involves only logarithms and powers of monomials of sines and cosines, and appears quite suitable for the numerical computation of the solutions of A) and B) and related problems with J_m replaced by Q_m. We will call the resulting interpolating and smoothing functions thin plate pseudo-splines on the sphere. Convergence rates for the thin plate pseudo-splines will be the same as those for the thin plate splines on the sphere because of the topological equivalence of J_m and Q_m.

In § 2 we derive the thin plate spline solutions to problems A) and B), and in § 3 we obtain the thin plate pseudo-spline solutions, where J_m is replaced by Q_m.

We remark that the development of § 2 can no doubt be generalized to establish splines associated with the Laplace–Beltrami operator on compact Riemannian manifolds other than the circle and the sphere; see Gine [10], Hannan [14], Yaglom [33] and Schoenberg [23]. However this is not pursued further.

2. Spherical harmonics and the solution to problems A) and B) on the sphere. The spherical harmonics $\{U_\nu^k(\theta, \phi)\}$ are defined by

$$\begin{aligned} U_\nu^k(\theta, \phi) &= \cos k\phi P_\nu^k(\cos\theta), \qquad k = 1, 2, \cdots, \nu \\ &= \sin k\phi P_\nu^k(\cos\theta), \qquad k = -1, -2, \cdots, -\nu \\ &= P_\nu(\cos\theta), \qquad k = 0, \quad \nu = 0, 1, 2, \cdots, \end{aligned}$$

where $P_\nu^k(z)$ are the Legendre functions of the kth order,

$$P_\nu^k(z) = (1-z^2)^{k/2}\left(\frac{d^k}{dz^k}\right)P_\nu(z),$$

and $P_\nu(z)$ is the νth Legendre polynomial. Recursion formulas for generating the P_ν^k may be found in Abramowitz and Stegun [1].

It is well known that the $\{U_\nu^k, k = -\nu, \cdots, \nu, \nu = 0, 1, \cdots\}$ form an $\mathscr{L}_2(S)$-complete set of eigenfunctions of the Laplace–Beltrami operator of (1.4) satisfying

$$\Delta U_\nu^k = -\nu(\nu+1)U_\nu^k, \qquad k = -\nu, \cdots, \nu, \quad \nu = 0, 1, \cdots.$$

See Courant and Hilbert [3], Sansone [22]. Let

$$Y_\nu^0 = \sqrt{\frac{2\nu+1}{4\pi}}U_\nu^0, \qquad \nu = 0, 1, \cdots,$$

$$Y_\nu^k = 2\sqrt{\frac{2\nu+1}{4\pi}\frac{(\nu-k)!}{(\nu+k)!}}U_\nu^k, \qquad k = 1, 2, \cdots, \quad \nu = 1, 2, \cdots.$$

Then (Sansone [22, p. 264, 268])

$$\int_S (Y_\nu^k(P))^2\, dP = 1,$$

and we have the addition formula

$$\sum_{k=-\nu}^{\nu} Y_\nu^k(P)Y_\nu^k(P') = \frac{2\nu+1}{4\pi}P_\nu(\cos\gamma(P, P')),$$

where $\gamma(P, P')$ is the angle between P and P'. The $\{Y_\nu^k\}$ form an orthonormal basis for $\mathscr{L}_2(S)$. Jones [16] has used a finite set of spherical harmonics to estimate 500 millibar heights by regression methods. The spherical harmonics are also utilized in several numerical weather prediction models [11].

Let $\mathscr{H}_m^0(S)$ be the subset of $\mathscr{L}_2(S)$ with an expansion of the form

$$u(P) \sim \sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} u_{\nu k} Y_\nu^k(P), \tag{2.1}$$

where

$$u_{\nu k} = \int_S u(P) Y_\nu^k(P)\, dP,$$

satisfying

$$\sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} \frac{u_{\nu k}^2}{\lambda_{\nu k}} < \infty,$$

where

$$\lambda_{\nu k} = [\nu(\nu+1)]^{-m}. \tag{2.2}$$

Functions in $\mathscr{H}_m^0(S)$ satisfy

$$\int_S u(P)\, dP = 0,$$

since the 0, 0th term $Y_0^0 \equiv 1$ has been omitted from the expansion (2.1).

$\mathscr{H}_m^0(S)$ is clearly a Hilbert space with the norm defined by

$$\|u\|_m^2 = \sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} \frac{u_{\nu k}^2}{\lambda_{\nu k}} \tag{2.3}$$

for any $m \geqq 0$. For $m > 1$, define $K(P, P')$, $(P, P') \in S \times S$ by

$$\begin{aligned} K(P, P') = K_m(P, P') &= \sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} \lambda_{\nu k} Y_\nu^k(P) Y_\nu^k(P') \\ &\equiv \frac{1}{4\pi} \sum_{\nu=1}^{\infty} \frac{2\nu+1}{\nu^m(\nu+1)^m} P_\nu(\cos \gamma(P, P')). \end{aligned} \tag{2.4}$$

Since $|P_\nu(z)| \leqq 1$ for $|z| \leqq 1$ (Sansone [22, p. 187]), the series converges uniformly for any $m > 1$ and $K(P, P')$ is a well-defined positive definite function on $S \times S$ with

$$\langle K(P, \cdot), K(P', \cdot) \rangle_m = K(P, P'),$$

where $\langle \cdot, \cdot \rangle_m$ is the inner product induced by (2.3). Furthermore, it is easily verified that, for m an integer >1,

$$\int_S K(P, R) \Delta_{(R)}^m K(P', R)\, dR = K(P, P'),$$

where $\Delta_{(R)}^m$ means the operator Δ^m applied to the variable R. This follows since $\Delta^m Y_\nu^k \equiv \lambda_{\nu k}^{-1} Y_\nu^k$. Thus, for m an integer >1, $K(\cdot, \cdot)$ reproduces under the inner product induced by the norm $J_m^{1/2}(\cdot)$, and

$$J_m(u) = \sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} \frac{u_{\nu k}^2}{\lambda_{\nu k}},$$

with

$$u_{\nu k} = \int_S u(P) Y_\nu^k(P)\, dP \quad \text{for any } u \in \mathscr{H}_m^0.$$

$\mathscr{H}_m^0(S)$ is therefore the reproducing kernel Hilbert space $(r\,k\,h\,s)$ with reproducing kernel $(rk)K(\cdot, \cdot)$.

The space $\mathcal{H}_m(S)$ in which one wants to solve problems A) and B) is

$$\mathcal{H}_m(S) = \mathcal{H}_m^0(S) \oplus \{1\},$$

where $\{1\}$ is the one-dimensional space of constant functions. $\mathcal{H}_m^0(S)$ and $\{1\}$ will be orthogonal subspaces in $\mathcal{H}_m(S)$ if we endow $\mathcal{H}_m(S)$ with the norm defined by

$$\|u\|^2 = J_m(u) + \frac{1}{4\pi}\left(\int_S u(P)\, dP\right)^2.$$

The following theorem is an immediate consequence of these facts and Kimeldorf and Wahba [17, Lemmas 3.1, 5.1].

THEOREM 1. *The solutions $u_{n,m}$ and $u_{n,m,\lambda}$ to problems* A) *and* B) *on the sphere are given by*

$$u_{n,m,\lambda}(P) = \sum_{i=1}^{n} c_i K(P, P_i) + d, \tag{2.5}$$

where $\mathbf{c} = (c_1, \cdots, c_n)'$ and d are given by

$$\mathbf{c} = (K_n + n\lambda I)^{-1}[I - T(T'(K_n + n\lambda I)^{-1}T)^{-1}T'(K_n + n\lambda I)^{-1}]\mathbf{z}, \tag{2.6}$$

$$d = (T'(K_n + n\lambda I)^{-1}T)^{-1}T'(K_n + n\lambda I)^{-1}\mathbf{z}, \tag{2.7}$$

where K_n is the $n \times n$ matrix with j, kth entry $(K_n)_{ij}$ given by

$$(K_n)_{ij} = K(P_i, P_j), \tag{2.8}$$

$$T = (1, \cdots, 1)' \tag{2.9}$$

and

$$\mathbf{z} = (z_1, \cdots, z_n)'.$$

Also

$$u_{n,m} \equiv u_{n,m,0}.$$

The continuous linear functionals $L_i u = u(P_i)$ may be replaced in the problem statements by any set of n linearly independent continuous linear functionals on $\mathcal{H}_m(S)$ which are not all identically 0 on $\{1\}$. Then, as is usual in rk theory, to obtain the solution one replaces $K(P, P_i)$ in (2.5) by $L_i K(P, \cdot)$, $K(P_i, P_j)$ in (2.8), by $L_{i(P)} L_{j(P')} K(P, P')$, and the ith component of T in (2.9) by $L_i(1)$. (See Kimeldorf and Wahba [17].) One example of useful L_i is $L_i u = \int_{S_i} u(P)\, dP$; i.e., the data functionals are regional averages (see Dyn and Wahba [5]). Furthermore, if $z_i = L_i u + \varepsilon_i$, where u is fixed, unknown function in $\mathcal{H}_m(S)$ and the $\{\varepsilon_i\}$ can be modeled as i.i.d. $\mathcal{N}(0, \sigma^2)$ random variables, then (provided λ is chosen properly; see [4], [30]) an estimate of Lu for L any continuous linear functional on $\mathcal{H}_m(S)$ is provided by $Lu_{m,n,\lambda}$. $Lu = \sin\theta u_\theta(P)$ and $Lu = u_\phi(P)$ are continuous linear functionals on $\mathcal{H}_m(S)$ for $m \geqq 3$. Therefore, this provides a technique for estimating meteorological properties of interest involving the derivatives of u, for example, the geostrophic wind; see [30]. Other potential applications are to the estimation of budgets (Johnson and Downey [15]), and the geostrophic vorticity (Haltiner and Martin [13]).

We remark that the equations (2.6) and (2.7) for $\mathbf{c}$ and d can be readily verified to be equivalent to

$$(K_n + n\lambda I)\mathbf{c} + dT = \mathbf{z}, \tag{2.10}$$

$$T'\mathbf{c} = 0. \tag{2.11}$$

If we assume K_n and T are given, then (2.10) and (2.11) lend themselves more readily to numerical solution than the computation of (2.6) and (2.7). See Paihua Montes [19], Wahba [28], Wendelberger [31].

In order to have a closed form expression for $K(P_i)$ it is necessary to sum the series

$$k_m(z)=\sum_{\nu=1}^{\infty}\frac{2\nu+1}{\nu^m(\nu+1)^m}P_\nu(z). \tag{2.12}$$

A closed form expression for $m=1$ $(z\neq 0)$ can be obtained but does not interest us here.

To attempt to sum (2.12) for $m=2$, we note that

$$\frac{2\nu+1}{\nu^2(\nu+1)^2}\equiv\frac{1}{\nu^2}-\frac{1}{(\nu+1)^2}\equiv\int_0^1\log h\left(1-\frac{1}{h}\right)h^\nu\,dh,\qquad \nu=1,2,\cdots. \tag{2.13}$$

Using the generating formula for Legendre polynomials (Sansone [22, p. 169]),

$$\sum_{\nu=1}^{\infty}h^\nu P_\nu(z)=(1-2hz+h^2)^{-1/2}-1,\qquad -1<h<1, \tag{2.14}$$

gives

$$k_2(z)=\int_0^1\log h\left(1-\frac{1}{h}\right)\left(\frac{1}{\sqrt{1-2hz+h^2}}-1\right)dh. \tag{2.15}$$

Repeated attempts to integrate this by parts using formulas for indefinite integrals involving expressions of the form $\sqrt{1-2zh+h^2}$, and related integrals to be found in Pierce and Foster [20] and Dwight [7], led us to terms with a closed form expression plus a term involving Dwight [7, formula 731.1] whose right-hand side is an infinite series. This exercise, plus a helpful conversation with R. Askey who suggested that the sum could be reduced to a dilogarithm, convinced us that no readily computable closed form expression was to be found. For this reason we seek to change the problem slightly so that readily computable interpolating and smoothing formulas can be obtained. We do this in the next section.

3. Thin plate pseudo-splines on the sphere. We seek a norm $Q_m^{1/2}(u)$ on $\mathscr{H}_m^0(S)$ which is topologically equivalent to $J_m^{1/2}(u)$ on $\mathscr{H}_m^0(S)$ and for which the reproducing kernel can be obtained in closed form convenient for computation.

Define

$$Q_m(u)=\sum_{\nu=1}^{\infty}{}'\sum_{k=-\nu}^{\nu}\frac{u_{\nu k}^2}{\xi_{\nu k}},\qquad u_{\nu k}=\int_S u(P)Y_\nu^k(P)\,dP,$$

where

$$\xi_{\nu k}=\left[\left(\nu+\frac{1}{2}\right)(\nu+1)(\nu+2)\cdots(\nu+2m-1)\right]^{-1}. \tag{3.1}$$

Since

$$\frac{1}{m^{2m}\xi_{\nu k}}\leqq\frac{1}{\lambda_{\nu k}}\leqq\frac{1}{\xi_{\nu k}},\qquad \nu=1,2,\cdots,\quad k=-\nu,\cdots,\nu,\quad m=2,3,\cdots,$$

we have

$$\frac{1}{m^{2m}}Q_m(u)\leqq J_m(u)\leqq Q_m(u),\qquad u\in\mathscr{H}_m^0(S),$$

and, thus the norms $J_m^{1/2}(\cdot)$ and $Q_m^{1/2}(\cdot)$ are topologically equivalent on $\mathcal{H}_m^0(S)$. The reproducing kernel $R(P, P')$ for $\mathcal{H}_m^0(S)$ with norm $Q_m^{1/2}(\cdot)$ is then

$$
\begin{aligned}
R(P, P') = R_m(P, P') &= \sum_{\nu=1}^{\infty} \sum_{k=-\nu}^{\nu} \xi_{\nu k} Y_\nu^k(P) Y_\nu^k(P') \\
&= \frac{1}{2\pi} \sum_{\nu=1}^{\infty} \frac{1}{(\nu+1)(\nu+2)\cdots(\nu+2m-1)} P_\nu(\cos \gamma(P, P')).
\end{aligned} \tag{3.2}
$$

A closed form expression can be obtained for $R(P, P')$ as follows. Since

$$
\frac{1}{r!}\int_0^1 (1-h)^r h^\nu \, dh \equiv \frac{1}{(\nu+1)\cdots(\nu+r+1)}, \qquad r = 0, 1, 2, \cdots,
$$

then by using the generating function (2.14) for the Legendre polynomials we have

$$
\begin{aligned}
R(P, P') &= \frac{1}{2\pi} \sum_{\nu=1}^{\infty} \frac{1}{(\nu+1)(\nu+2)\cdots(\nu+2m-1)} P_\nu(z) \\
&= \frac{1}{2\pi}\left[\frac{1}{(2m-2)!} q_{2m-2}(z) - \frac{1}{(2m-1)!}\right],
\end{aligned} \tag{3.3}
$$

where

$$
z = \cos \gamma(P, P')
$$

and

$$
q_m(z) = \int_0^1 (1-h)^m (1-2hz+h^2)^{-1/2} \, dh, \qquad m = 0, 1, \cdots. \tag{3.4}
$$

Formulas for $\int h^m (1-2hz+h^2)^{-1/2} \, dh$, $m = 0, 1, 2$ and recursion formulas for general m in terms of the formulas for $m-1$ and $m-2$ can be found in Pierce and Foster [20, pp. 165, 174, 177, 196]. q_m was obtained by hand for $m = 0, 1, 2$ and 3. In the middle of this dull exercise P. Bjornstad observed that the MACSYMA program at MIT, which could be called from the computer science department at Stanford where this exercise was taking place, could be used to evaluate $q_m(z)$ recursively. He kindly wrote such a program and the results appear in Table 1. Thus, for example, $R(P, P')$ for $m = 2$ involves q_2 and, from the table,

$$
q[2] = \frac{A(12W^2 - 4W) - 6CW + 6W + 1}{2},
$$

giving

$$
q_2(z) = \frac{1}{2}\left\{\ln\left(1+\sqrt{\frac{2}{1-z}}\right)\left[12\left(\frac{1-z}{2}\right)^2 - 4\frac{(1-z)}{2}\right] - 12\left(\frac{1-z}{2}\right)^{3/2} + 6\left(\frac{1-z}{2}\right) + 1\right\}.
$$

Note that $q[0]$ which appears in the $m = 1$ case does not lead to a proper *rk* since $q_0(1)$ is not finite. However, a proper *rk* exists for any $m > 1$, and the table can be used to define q_{2m-2} for $m = \frac{3}{2}, 2, \frac{5}{2}, \cdots, 6$.

We collect these results in

THEOREM 2. *The solutions $u_{n,m}$ and $u_{n,m,\lambda}$ to the problems: Find $u \in \mathcal{H}_m(S)$ to*

A′) *minimize $Q_m(u)$ subject to $u(P_i) = z_i$, $i = 1, 2, \cdots, n$,*

B′) *minimize* $\dfrac{1}{n}\sum_{i=1}^{n} (u(P_i) - z_i)^2 + \lambda Q_m(u)$,

are given by

$$\hat{u}_{n,m} = \hat{u}_{n,m0}, \qquad \hat{u}_{n,m,\lambda}(P) = \sum_{i=1}^{n} c_i R(P, P_i) + d,$$

where $R(P, P')$ *is defined by* (3.3) *and* (3.4) *and* $\mathbf{c}$ *and* d *are determined by*

$$(R_n + n\lambda I)\mathbf{c} - dT = \mathbf{z}, \qquad T'\mathbf{c} = 0,$$

where R_n *is the* $n \times n$ *matrix with* j, kth *entry* $R(P_i, P_j)$ *and* $T = (1, \cdots, 1)'$.

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
$q[1]$	$2AW - C + 1$
$q[2]$	$(A(12W^2 - 4W) - 6CW + 6W + 1)/2$
$q[3]$	$(A(60W^3 - 36W^2) + 30W^2 + C(8W - 30W^2) - 3W + 1)/3$
$q[4]$	$(A(840W^4 - 720W^3 + 72W^2) + 420W^3 + C(220W^2 - 420W^3) - 150W^2 - 4W + 3)/12$
$q[5]$	$(A(7560W^5 - 8400W^4 + 1800W^3) + 3780W^4 + C(-3780W^4 + 2940W^3 - 256W^2)$ $-2310W^3 + 60W^2 - 5W + 6)/30$
$q[6]$	$(A(27{,}720W^6 - 37{,}800W^5 + 12{,}600W^4 - 600W^3) + 13{,}860W^5$ $+ C(-13{,}860W^5 + 14{,}280W^4 - 2772W^2) - 11970W^4 + 1470W^3 + 15W^2 - 3W + 5)/30$
$q[7]$	$(A(360{,}360W^7 + 582{,}120W^6 + 264{,}600W^5 + 29{,}400W^4) + 180{,}180W^6$ $+ C(-180{,}180W^6 + 231{,}000W^5 - 71{,}316W^4 + 3072W^3)$ $-200{,}970W^5 + 46{,}830W^4 - 525W^3 + 21W^2 - 7W + 15)/105$
$q[8]$	$(A(10{,}810{,}800W^8 - 20{,}180{,}160W^7 + 11{,}642{,}400W^6 + 2{,}116{,}800W^5 + 58{,}800W^4)$ $+ 5{,}405{,}400W^7 + C(-5{,}405{,}400W^7 + 8{,}288{,}280W^6 - 3{,}538{,}920W^6 + 363{,}816W^4)$ $-7{,}387{,}380W^6 + 2{,}577{,}960W^5 - 159{,}810W^4 - 840W^3 + 84W^2 - 40W + 105)/840$
$q[9]$	$(A(61{,}261{,}200W^9 - 129{,}729{,}600W^8 + 90{,}810{,}720W^7 - 23{,}284{,}800W^6 + 1{,}587{,}600W^5)$ $+ 30{,}630{,}600W^8 + C(-30{,}630{,}600W^8 + 54{,}654{,}600W^7 - 29{,}909{,}880W^6$ $+ 5{,}104{,}440W^5 - 131{,}072W^4) - 49{,}549{,}500W^7 + 23{,}183{,}160W^6$ $-2{,}903{,}670W^5 + 17{,}640W^4 - 420W^3 + 72W^2 - 45W + 140)/1{,}260$
$q[10]$	$(A(232{,}792{,}560W^{10} - 551{,}350{,}800W^9 + 454{,}053{,}600W^8 - 151{,}351{,}200W^7 + 17{,}463{,}600W^6$ $- 317{,}520W^5) + 116{,}396{,}280W^9 + C(-116{,}396{,}280W^9 + 236{,}876{,}640W^8$ $- 158{,}414{,}256W^7 + 38{,}507{,}040W^6 - 2{,}462{,}680W^5) - 217{,}477{,}260W^8$ $+ 127{,}987{,}860W^7 - 24{,}954{,}930W^6 + 930{,}006W^5 + 2{,}940W^4$ $- 180W^3 + 45W^2 - 35W + 126)/1{,}260$

Of course the remarks following Theorem 1 concerning general continuous linear functionals and computing procedures apply here also.

4. Acknowledgments. We would like to thank P. Bjornstad for writing the computer program which generated Table 1, G. Golub for his hospitality at the Stanford University Numerical Analysis Group where some of this work was done and M. Ghil for helpful discussions concerning related stochastic differential equations.

REFERENCES

[1] M. ABRAMOWITZ AND I. STEGUN, *Handbook of Mathematical Functions*, National Bureau of Standards, Applied Mathematics Series 55, 1964.

[2] N. ARONSZAJN, *Theory of reproducing kernels*, Trans. Amer. Math. Soc., 68, (1950), pp. 337–303.

[3] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics, vol.* 1, Interscience, New York, 1953.

[4] P. CRAVEN AND G. WAHBA, *Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation*, Numer. Math., 31 (1979), pp. 377–403.

[5] N. DYN AND G. WAHBA. On the estimation of functions of several variables from aggregated data. University of Wisconsin-Madison, Mathematics Research Center, TSR 1974, July 1979.

[6] J. DUCHON, Interpolation des fonctions de deux variables suivante le principe de la flexion des plaques minces. R.A.I.R.O. Analyse Numerique, 10 (1976), pp. 5–12.

[7] H. B. DWIGHT, *Tables of Integrals and Other Mathematical Data*, Macmillan, New York, 1947.

[8] S. D. FISHER AND J. W. JEROME, *Elliptic variational problems in L^2 and L^∞*, Indiana Univ. Math. J., 23 (1974), pp. 685–698.

[9] R. FRANKE, *A critical comparison of some methods for interpolation of scattered data*, Naval Post-graduate School Report NPS-53-79-003, 1979.

[10] E. GINÉ, *Invariant tests for uniformity on compact Riemannian manifolds based on Sobolev norms*, Ann. Statist., 36 (1975), pp. 1243–1266.

[11] Global Atmospheric Research Programme (GARP), *Modelling for the first GARP global experiment*, GARP Publications, Series No. 14, World Meteorological Organization, Geneva, Switzerland.

[12] M. GOLOMB, *Approximation by periodic spline interpolants on uniform meshes*, J. Approx. Theory, 1 (1968), pp. 26–65.

[13] G. J. HALTINER AND F. L. MARTIN, *Dynamic and Physical Meteorology*, McGraw-Hill, New York, 1979, p. 358.

[14] E. J. HANNAN, *Group representations and applied probability*, J. Appl. Probab., 2 (1965), pp. 1–68.

[15] D. R. JOHNSON AND W. K. DOWNEY, *The absolute angular momentum budget of an extratropical cyclone, Quasi Langrangian diagnostics* 3, Monthly Weather Review, 104 (1976), pp. 3–14.

[16] R. H. JONES, *Stochastic process on a sphere as applied to meterological* 500*mb forecasts*, in Time Series Analysis, Murray Rosenblatt, ed., John Wiley, New York, 1963.

[17] G. KIMELDORF AND G. WAHBA, *Some results on Tchebycheffian spline functions*, J. Math. Anal. Applic., 33, (1971), pp. 82–95.

[18] J. MEINGUET, *Multivariate interpolation at arbitrary points made simple*, ZAMP, to appear.

[19] L. PAIHUA MONTES, *Quelque methodes numeriques pour le calcul de fonctions splines a une et plusiers variables*, Thesis, Université Scientifique et Medicale de Grenoble, 1978.

[20] B. O. PIERCE AND R. M. FOSTER, *A Short Table of Integrals*, Grun and Company, 1956.

[21] Y. SASAKI, *A theoretical interpretation of anisotropically weighted smoothing on the basis of numerical variational analysis*, Monthly Weather Review, 99 (1971), pp. 698–707.

[22] G. SANSONE, *Orthogonal Functions*, revised English edition, Interscience, New York, 1959.

[23] I. J. SCHOENBERG, *Positive definite functions on spheres*, Duke Univ. Math. Journal, 9 (1942), pp. 96–108.

[24] J. L. STANFORD, *Latitudinal-wavenumber power spectra of stratospheric temperature fluctuations*, J. Atmospheric Sci., 36 (1979), pp. 921–931.

[25] F. UTERAS DIAZ, *Quelques resultats d'optimalite pour la methode de validation croisée*, No. 301 mathématiques appliquées, Université Scientifique et Medicale de Grenoble, 1978.

[26] F. UTRERAS DIAZ, *Cross validation techniques for smoothing spline functions in one or two variables*, in Smoothing Techniques for Curve Estimation, T. Gasser and M. Rosenblatt, eds., Lecture Notes in Mathematics 757, Springer-Verlag, New York, 1979, pp. 196–231.

[27] G. WAHBA, *Smoothing noisy data by spline functions*, Numer. Math., 24 (1975), pp. 383–393.

[28] ———, *How to smooth curves and surfaces with splines and cross-validation*, in Proceedings of the 24th Conference on the Design of Experiments in Army Research Development and Testing, ARO Report 79-2, U.S. Army Research Office, Research Triangle Park, North Carolina, 1979, also University of Wisconsin, Madison, Department of Statistics Technical Report 555, 1979.

[29] ———, *Spline interpolation and smoothing on the sphere*, University of Wisconsin, Madison, Department of Statistics Technical Report 584, October, 1979.

[30] G. WAHBA AND J. WENDELBERGER, *Some new mathematical methods for variational objective analysis using splines and cross-validation*, Monthly Weather Review, 108 (1980), pp. 1122–1143.

[31] J. WENDELBERGER, Thesis, to appear.

[32] E. T. WHITTAKER AND G. N. WATSON, *A course of modern analysis*, Cambridge University Press, Cambridge, 1958.

[33] A. M. YAGLOM, *Second order homogeneous random fields*, in Fourth Berkeley Symposium in Probability and Statistics, 2,, J. Neyman, ed., University of California Press, pp. 593–622.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0003 $01.00/0

THE MOLLIFICATION METHOD AND THE NUMERICAL SOLUTION OF AN INVERSE HEAT CONDUCTION PROBLEM*

DIEGO A. MURIO†

Abstract. We show how the inverse problem can be stabilized by reconstructing a slightly "blurred" image of the unknowns. The numerical problem is solved with an absolute minimum of computation and the proposed method is favorably compared against others commonly in use.

Key words. ill-posed problem, mollification method, heat transfer conduction

1. Introduction. In this paper we consider a transient heat conduction problem on a semi-infinite slab with one-dimensional symmetry in which, after measuring the transient temperature history $F(t)$ at some interior point x_1, we would like to recover the boundary heat flux $q(t)$ and temperature $f(t)$. The temperature history $F(t)$ is approximately measurable for all t in $(-\infty, \infty)$.

This inverse problem is an improperly-posed problem in the sense of Hadamard [7]; that is, there are no decent norms for the data and solutions such that the solution depends continuously upon the data.

It is known that certain types of continuous dependence on data can usually be restored by restricting attention to those solutions satisfying certain prescribed global bounds; see Miller [9] and [10], for example. If the unknown function q (or f) is quite smooth, for instance, it is reasonable to assume that some high order derivative of q satisfies a known L^2-bound (Manselli and Miller [8]).

However, for many problems of interest it is to be expected that the unknown function is not very smooth. In this case, the inverse problem can be stabilized if, instead of attempting to find the point values of q, we content ourselves with attempting to reconstruct a slightly "blurred" image of q. One natural functional is $J_\delta q$, the "δ-mollification" of q, that is, the convolution of q with the Gaussian kernel ρ_δ of "blurring radius" δ. Such an approach was also taken in [8].

One of the first papers on inverse heat conduction was written by Stolz [14]. His procedure is an integral equation method which, when discretized, allows a step by step recursive calculation of the solution, but which is unstable if the time steps are made small. Integral equation methods with step by step solutions are also used by Sparrow et al. [13] and by Beck [1], [2], who however adds the distinct improvement of allowing the least-squares use of several future data points to compute the solution at the present time. Burggraf [3] uses a truncation of the classical power series method for attempting to solve the Cauchy initial value problem; however, for this to be convergent the functions involved would have to be analytic and the space interval sufficiently small. Other approaches, using finite difference methods, are studied by Frank [6] and Davies [5].

In the papers mentioned above, the assumptions on the solutions and on the choice of parameters are not usually clearly stated and the consequent continuity with respect to the data is not adequately studied.

In § 2 we present the nondiscrete version of the semi-infinite one-dimensional problem with data specified on a continuum of times t and data error measured in the L^2-norm, and derive stability bounds for the inverse problem.

* Received by the editors May 5, 1980.
† Department of Mathematical Sciences, University of Cincinnati, Cincinnati, Ohio 45221.

The discretized version, involving data at only a discrete sampling of times, occupies our attention in § 3.

In § 4 we introduce certain direct and inverse convolution kernels with which we shall compute our numerical solution and analyze the solution error. We also compare the mollification method (MM) against the one commonly in use due to J. Beck [12]. The graphical comparison of the inverse and direct kernels for the MM method and Beck's shows the clearly superior stability and monotonicity properties of the MM method for a given resolution width. Finally, we present more numerical results of interest.

2. Semi-infinite body. Description of the problem. We consider a semi-infinite slab with one-dimensional symmetry; after measuring the transient temperature history $F(t)$ at some interior point x, we would like to recover the boundary heat flux $q(t)$ and temperature $f(t)$.

We assume linear heat conduction with constant coefficients. After appropriate changes in the space and time scales we may consider without loss of generality the normalized problem with constant conductivity 1 and heat capacity 1. Moreover, we measure the data temperature at the interior point $x_1 = 1$.

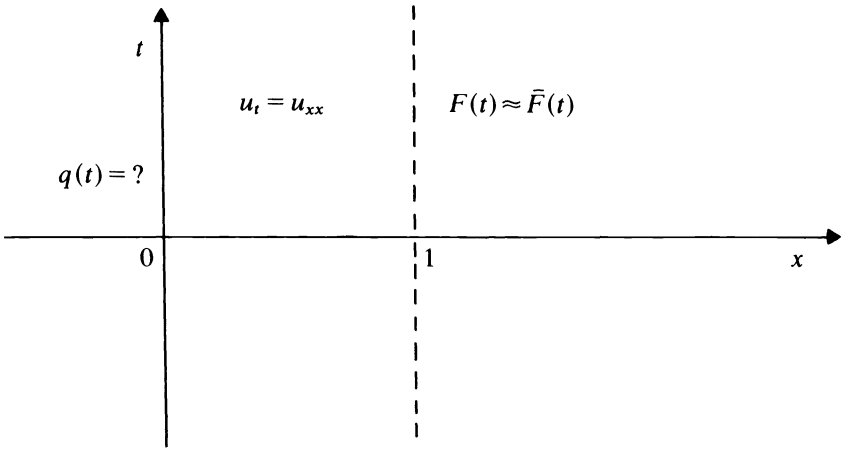
The problem can be described mathematically as follows:

The unknown temperature $u(x, t)$ satisfies

(2.1) $\quad u_t = u_{xx}, \quad 0 < x < \infty, \quad -\infty < t < \infty, \quad u$ bounded as $x \to \infty$,

(2.2) $\quad u(1, t) = F(t)$ with corresponding approximate data function $\bar{F}(t)$,

(2.3) $\quad -u_x(0, t) = q(t)$, the desired but unknown heat flux function.



The temperature function $f(t)$ is of course also desired, but since it depends simply and stably on $q(t)$ we do not need to bother to analyze its computation.

The sinusoidal in time solutions of the heat equation (2.1) are of the form

(2.4) $\quad e^{-(\mu + i\sigma\mu)x} e^{iwt}$, where $\mu = \sqrt{\frac{|w|}{2}}$, $\quad \sigma = \text{sign}\,(w)$.

From (2.2) and the boundary condition (2.3) we get

(2.5) $\quad F(t) \equiv Aq(t) = \dfrac{e^{-(\mu + i\sigma\mu)}}{\mu + i\sigma\mu} q(t).$

Thus, if

$$q(t) = e^{iwt}, \tag{2.6}$$

then

$$Aq(t) = \frac{e^{-(\mu + i\sigma\mu)}}{\mu + i\sigma\mu} e^{iwt}, \tag{2.7}$$

showing that the operator A is strongly smoothing for high frequencies w. In fact, for the direct problem, a high frequency sinusoidal component is damped by the factor $\approx e^{-\mu}(\sqrt{2}\,\mu)^{-1}$ and delayed in its phase by the amount $1/\sqrt{2|w|} + 1/|w|$.

Conversely, however, the inverse problem (attempting to go from $F(t) = Aq(t)$ to $q(t)$) magnifies an error in a high frequency component by the factor $e^{\mu}\sqrt{2}\,\mu$, showing that this inverse problem is greatly ill-posed in the high frequency components.

The stabilized inverse problem. For the moment, in order to use Fourier integral analysis, we are going to assume that all the functions involved are L^2-functions on the whole line $(-\infty, \infty)$ and we will use the corresponding L^2-norm to measure errors. This is rather unnatural, since in many applications one might expect the temperature and the flux never to tend to 0 as $t \to \pm\infty$ but to oscillate about in bounded fashion forever. Nevertheless, this assumption will be later loosened by switching to L^2-norms on bounded intervals of interest.

There are several different ways to stabilize the inverse problem, despite the fact that errors in the high frequency components of $\bar{F}(t)$ are greatly magnified. One can, for example, hypothesize that the unknown q itself is quite smooth, i.e., that some high order derivative of q satisfies a known prescribed L^2-bound. For example, we might know a priori that

$$\|q''\| \leqq E. \tag{2.8}$$

Since $\hat{q}''(w) = -w^2\hat{q}(w)$, this bound in terms of the Fourier transform means

$$\left[2\pi \int_{-\infty}^{+\infty} |w^2\hat{q}|^2 dw\right]^{1/2} \leqq E. \tag{2.9}$$

Such a bound, therefore, just forces the high frequency components of $\hat{q}$ to be fairly small. Since this is the case, we can attempt to determine rather stably the pointwise values $q(\xi)$ at times ξ of interest.

We thus have the problem: Attempt to find the value of $q(\xi)$ at some time ξ of interest, given that

$$\|Aq - \bar{F}\| \leqq E, \tag{2.10}$$

$$\|q''\| < E. \tag{2.11}$$

However, this problem leads to only very poor stability. In fact, the error can be guaranteed to go down only logarithmically as $\varepsilon \to 0$.

On the other hand, for many problems of interest, it is to be expected that the unknown q is not very smooth; thus only a L^2-bound on q itself, rather than on some high order derivative of q, seems appropriate. Because of these considerations, it is natural to assume only a known L^2 global error bound on q,

$$\|q\| \leqq E. \tag{2.12}$$

This bound in terms of the Fourier transform means

$$\left[2\pi \int_{-\infty}^{+\infty} |\hat{q}|^2 \, dw\right]^{1/2} \leqq E, \tag{2.13}$$

and in this case there is nothing that adequately forces down the high frequency part of $\hat{q}(w)$. Therefore, instead of attempting to find the pointwise values of q, we must lower our sights and seek instead some meaningful and useful functional of q which strongly damps the high frequency part of $\hat{q}(w)$.

One such useful functional is $J_\delta q$, the "δ-mollification" of q at time t, defined as

$$J_\delta q(t) \equiv (\rho_\delta * q)(t), \quad \text{where} \tag{2.14}$$

$$\rho_\delta(t) = \frac{1}{\delta\sqrt{\pi}} e^{-t^2/\delta^2} \tag{2.15}$$

is the Gaussian kernel of "blurring radius" δ.

We notice that ρ_δ falls to nearly zero outside a few δ radii from its center, is positive and has total integral 1. $J_\delta q$ merely smooths off the corners of q and damps those short term oscillations on a scale $\lesssim\delta$ while still representing faithfully the longer term features of q. The fact that, among all possible mollification kernels of "spread" δ, the Gaussian kernel has the smallest spread of its Fourier transform makes it a natural choice to work with.

The Fourier transform of $J_\delta q$ is given by

$$\widehat{J_\delta q}(w) = 2\pi\hat{\rho}_\delta \cdot \hat{q} = e^{-w^2\delta^2/4}\hat{q}(w). \tag{2.16}$$

Clearly, this mollification damps those Fourier components of q with wavelength $2\pi/w$ much shorter than $2\pi\delta$; the longer wavelengths are damped hardly at all.

Moreover, $J_\delta q(t)$ at $t = 0$ gives

$$J_\delta q(0) = \int_{-\infty}^{+\infty} \rho_\delta(-s) q(s) \, ds = \int_{-\infty}^{+\infty} e^{-w^2\delta^2/4}\hat{q}(w) \, dw. \tag{2.17}$$

We thus have the following stabilized problem: Attempt to find the linear function $J_\delta q(\xi)$ at some time ξ of interest and for some assigned blurring radius δ, given that q is a particular function satisfying

$$\|Aq - \bar{F}\| \leqq \varepsilon, \tag{2.18}$$

$$\|q\| \leqq E. \tag{2.19}$$

It will turn out in the stability analysis that the bound (2.19) is not necessary to stabilize this problem; the data error bound (2.18) by itself is sufficient to assure Lipschitz continuous dependence on the data as ε tends to zero, provided we keep δ fixed.

However, the prescribed bound (2.19) can actually aid the stability in the case of ε which are not small, or in the case of data at only discrete sampling points in a limited data interval, as will occur in the physically practical numerical methods of § 3.

Stability analysis. The problem is now in a form that can be solved by the method of least-squares; see Miller [9] and [10].

This method is a "nearly-best-possible-method" in the sense that for any seminorm $\langle \cdot \rangle$ which might be used to measure the error, it gives an approximation $\bar{q}$ to q

which satisfies the error bound

(2.20) $\langle \bar{q}-q\rangle \leqq 2M(\varepsilon, E)$, where $M(\varepsilon, E)$ is the "best-possible-stability-bound,"

(2.21) $$M(\varepsilon, E)=\sup\{\langle q\rangle: \|Aq\|\leqq\varepsilon, \|q\|\leqq E\}.$$

As a seminorm to measure the error, we will use the absolute value of the linear function $J_\delta q(\xi)$ at some fixed time ξ of interest, i.e.,

(2.22) $$\langle q(\xi)\rangle = |J_\delta q(\xi)|.$$

If q satisfies (2.18) and (2.19), it also satisfies

(2.23) $$\|\bar{F}-Aq\|^2+\left(\frac{\varepsilon}{E}\right)^2\|q\|^2\leqq 2\varepsilon^2,$$

and we have lost at most a factor of $\sqrt{2}$ going from the two constraints to the one.

Let our approximation $\bar{q}$ be chosen such as to minimize

(2.24) $$\left\{\|\bar{F}-Aq\|^2+\left(\frac{\varepsilon}{E}\right)^2\|q\|^2\right\}.$$

The canonical equation for this minimization is given by

(2.25) $$\left\{A^*A+\left(\frac{\varepsilon}{E}\right)^2 I\right\}\bar{q}=A^*\bar{F}.$$

We can now derive an estimate for $M(\varepsilon, E)$ for the linear functional $J_\delta q(\xi)$. We may of course assume the time of interest to be $\xi=0$.

We want the supremum of

(2.26) $$|J_\delta q(0)| = \left|\int_{-\infty}^{+\infty} e^{-w^2\delta^2/4}\hat{q}(w)\,dw\right|$$

with respect to the two constraints

(2.27) $$\|Aq\|^2 = 2\pi\int_{-\infty}^{+\infty}\left|\frac{e^{-(\mu+i\sigma\mu)}}{\mu+i\sigma\mu}\hat{q}(w)\right|^2 dw\leqq\varepsilon^2$$

and

(2.28) $$\|q\|^2 = 2\pi\int_{-\infty}^{+\infty}|\hat{q}|^2\,dw\leqq E^2.$$

However, it is sufficient to bound (2.26) by the single constraint (2.27) alone. Using the Cauchy inequality, we obtain

(2.29)
$$\begin{aligned}|J_\delta q(0)| &\leqq \varepsilon(2\pi)^{-1/2}\left\{\int_{-\infty}^{+\infty}|e^{-w^2\delta^2/4}\,e^{\mu+i\sigma\mu}(\mu+i\sigma\mu)|^2\,dw\right\}^{1/2}\\ &\leqq \varepsilon\pi^{-1/2}\left\{\int_{-\infty}^{+\infty}e^{-w^2\delta^2/2}\,e^{2\mu}\mu^2\,dw\right\}^{1/2}.\end{aligned}$$

Since $e^{-w^2\delta^2/2}$ is about 0.6 for $w\leqq w_1\equiv 1/\delta$ and falls rapidly to zero for $w>w_1$, while on the other hand $\mu^2e^{2\mu}=(w/2)e^{\sqrt{2}w}$ grows only slowly with w, it

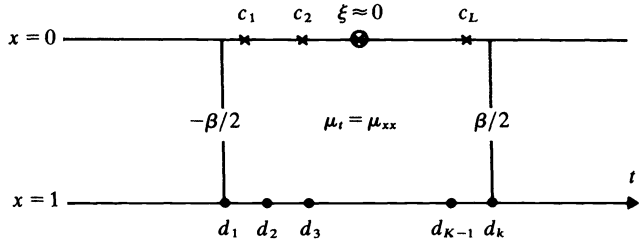
follows that

$$|J_\delta q(0)| \leqq 2\varepsilon\pi^{-1/2}\left\{\int_0^{w_1} \mu^2 e^{2\mu}\, dw\right\}^{1/2} \leqq \varepsilon 2^{1/2}\pi^{-1/2}\delta^{-1}\exp(1/\sqrt{2\delta}), \tag{2.30}$$

which as $\varepsilon \to 0$ becomes the best possible bound but for a factor of two, for fixed δ.

This shows that the error can be guaranteed to go down in Lipschitz fashion as $\varepsilon \to 0$ for fixed δ.

3. Discretized problem. Transition to the discrete case. For many problems of interest, the requirement that our functions are L^2-bounded on the whole infinite interval $(-\infty, \infty)$ is very unnatural. Instead, it seems more appropriate to assume that q is locally L^2-bounded, uniformly on every sufficiently long interval, and that the data for Aq are measured at a discrete set of equally spaced data points in some finite interval of length β; we then seek to reconstruct $J_\delta q(\xi)$ at some point ξ approximately opposite the middle of the data set, as shown in Fig. 1.



●: *discrete data set where* $Aq \approx \bar{F}$.
×: *constraint set where q is bounded.*
⊗: *desired reconstruction point.*

FIG. 1

We pose, then, the following precise problem: Suppose q is an unknown function satisfying

$$\|\bar{F} - Aq\|_{\text{data set}} \leqq \varepsilon, \tag{3.1}$$

$$\|q\|_C \leqq E, \tag{3.2}$$

for every interval C of length greater than or equal to a certain minimal value $\beta_0 \leqq \beta$. We wish to approximately determine the linear functional $J_\delta q(\xi)$ at some time ξ of interest; without loss of generality, we choose our point of reconstruction to be $\xi = 0$.

In this problem, the data set consists of K points $d_1, \cdots, d_k$ in the interval $[-\beta/2, \beta/2]$, with equal spacing $\Delta s = \beta/(K-1)$. The data function $\bar{F}$ is a discrete function measured at these sampling points. The two norms used are the discrete (on the data set) and continuous (on the interval C) root mean square norms.

The interval $[-\beta/2, \beta/2]$ should contain all the data for Aq which might reasonably be expected to enter into the reconstruction of q at time $\xi = 0$, provided we make β sufficiently large. If that is the case, since the operator A makes Aq so smooth, we have reason to believe that a discrete data sampling of Aq in $[-\beta/2, \beta/2]$ contains just about as much information as a continuous sampling, provided that the sampling interval Δs is made sufficiently small.

Aq is given by the integral operator

$$Aq(t) = \int_{-\infty}^{t} P(t-s)q(s)\,ds, \tag{3.3}$$

where the kernel function P has the formula

$$P(t-s) = \begin{cases} (\pi(t-s))^{-1/2} \exp(-1/4(t-s)) & \text{if } t>0, \\ 0 & \text{if } t \leqq 0. \end{cases} \tag{3.4}$$

We note that P is nearly zero for small $t-s$, rises up to a maximum at $t-s=0.5$ and then slowly decreases monotonically to zero. We therefore see that the values of q near the time ξ of interest influence the value of the data $Aq(t)$ almost not at all for small $t-\xi$, and very much for longer $t-\xi$.

In other words, $Aq(t)$ depends strongly on almost the entire "past history of q." This means that $\beta/2$ should be taken so large that the number of points in the discrete data sampling interval $[-\beta/2, \beta/2]$ will become enormous when the sampling interval Δs is made sufficiently small.

One way of avoiding this fact is to subtract from the actual data $Aq(t)$ the influence of the past history of q. Of course, this implies the knowledge of q for all previous times. Because of this difficulty, we will lower our goal for the moment and assume, as an alternative,

$$Aq(t) \equiv 0 \quad \text{for } t \leqq 0. \tag{3.5}$$

The choice of $\beta/2$ sufficiently large certainly allows us to approximate our problem rigorously by a completely discretized one, with q replaced on the constraint set $[-\beta/2+\Delta t/2, \beta/2-\Delta t/2]$ by a piecewise polynomial $\bar{q}$ of the form

$$\bar{q}(t) = \sum_{j=1}^{L} a_j(t) I_j(t) \quad \text{with } (L+1)\,\Delta t = \beta. \tag{3.6}$$

Here, $I_j(t)$ is the characteristic function of the jth subinterval $[(j-\frac{1}{2})\Delta t - \beta/2, (j+\frac{1}{2})\Delta t - \beta/2]$, and

$$a_j(t) = \sum_{i=1}^{M} X_j^i \Psi_i(t), \tag{3.7}$$

where the Ψ_i's are some suitable chosen basic functions or low order polynomials.

Since the operators A and J_δ are very smoothing, it is easy to pick Δt sufficiently small such that $A(\bar{q}-q) < 0.1\varepsilon$ on the data set and $J_\delta(\bar{q}-q)$ is negligibly small at the reconstruction point ξ of interest.

For the numerical application of the method of least-squares to a related problem, where q is assumed to be a β-periodic Nth order trigonometric sum over the interval of interest instead of the approximation mentioned above, see Manselli and Miller [8] and Murio [11].

In order to emphasize just the flexibility of making a complete discretization of the problem, we may replace the L^2-norm in the constraint set $[-\beta/2+\Delta t/2, \beta/2-\Delta t/2]$ by the l^2 or root mean square norm on a discrete set of L equally spaced points $c_1, \cdots, c_L$, with $c_1 = -\beta/2+\Delta t$ and $c_L = \beta/2-\Delta t$.

Thus, any unknown function satisfying (3.1), (3.2) and (3.5) can be well approximated by some finite sum $\bar{q}$ of the form (3.6), in the precise sense that

$$
\begin{aligned}
&\text{(i)} \quad J_\delta \bar{q} - J_\delta q \text{ is negligibly small at the time } \xi \text{ of interest,}\\
&\text{(ii)} \quad \|\bar{F} - A\bar{q}\|_{[-\beta/2,\beta/2]} \leqq 1.1\varepsilon,\\
&\text{(iii)} \quad \|\bar{q}\|_{[-\beta/2+\Delta t/2,\beta/2-\Delta t/2]} \leqq E,\\
&\text{(iv)} \quad \bar{F} = 0 \quad \text{on } [-\beta/2, 0].
\end{aligned} \tag{3.8}
$$

In the following we are going to wipe out all distinctions in the notation between ε and 1.1ε, $\bar{q}$ and q, and we will concentrate on solving the discretized problem.

Numerical method. Given a function q on $[-\beta/2+\Delta t/2, \beta/2-\Delta t/2]$ of the form

$$q(t) = \sum_{j=1}^{L} a_j(t) I_j(t), \tag{3.9}$$

satisfying

$$\|Aq - \bar{F}\|_{[-\beta/2,\beta/2]} \leqq \varepsilon \quad \text{with } \bar{F} = 0 \quad \text{on } [-\beta/2, 0] \tag{3.10}$$

and

$$\|q\|_{[-\beta/2+\Delta t/2,\beta/2-\Delta t/2]} \leqq E, \tag{3.11}$$

we wish to approximately determine the linear functional $J_\delta q(\xi)$ at the point of interest $\xi = 0$.

Combining the two constraints into one, and losing at most a factor of $\sqrt{2}$ in the process, we have

$$\phi(q) = \|Aq - \bar{F}\|^2_{[-\beta/2,\beta/2]} + \left(\frac{\varepsilon}{E}\right)^2 \|q\|^2_{[-\beta/2+\Delta t/2,\beta/2-\Delta t/2]} \leqq 2\varepsilon^2. \tag{3.12}$$

Our least-squares approximation is chosen to be that element q^0 of the form (3.9) (with coefficient vector x^0) which minimizes

$$\phi(q) = \left\{ \|Aq - \bar{F}\|^2_{[-\beta/2,\beta/2]} + \left(\frac{\varepsilon}{E}\right)^2 \|q\|^2_{[-\beta/2+\Delta t/2,\beta/2-\Delta t/2]} \right\}. \tag{3.13}$$

Next, we write the quadratic functional $\phi(q)$ in terms of the coefficient vector x of q.

Thus, the least-squares problem becomes

$$\text{minimize } \phi(x) = \|Hx - h\|^2_K + \left(\frac{\varepsilon}{E}\right)^2 \|Ix\|^2_N, \tag{3.14}$$

where x is the N-dimensional partitioned vector with components

$$x_j = (x_j^1, x_j^2, \cdots, x_j^M)^*, \qquad j = 1, \cdots, L, \quad N = L \times M, \tag{3.15}$$

h is the vector with elements

$$h_k = \bar{F}(d_k)/\sqrt{K}, \qquad k = 1, \cdots, K, \tag{3.16}$$

H is the $K \times N$ partitioned matrix with elements

$$H_{kj} = K^{-1/2}\left(\sum_{i=1}^{M} \theta_i(d_k + \beta/2 - (j - \tfrac{1}{2})\,\Delta t) - \theta_i(d_k + \beta/2 - (j + \tfrac{1}{2})\,\Delta t) \right), \tag{3.17}$$

$$k = 1, 2, \cdots, K, \quad j = 1, 2, \cdots, L.$$

Here $\theta_i(s, t)$ is the temperature response at the point s corresponding to a surface flux $\Psi_i(0, t)$.

The vector x^0 minimizing (3.14) is the solution of the normal equations

$$Zx^0 = \left(H^*H + \left(\frac{\varepsilon}{E}\right)^2 I\right)x^0 = H^*h. \tag{3.18}$$

The desired linear functional can be wrtten as

$$J_\delta q(\xi) = (\rho_\delta * q)(\xi) = \frac{1}{\delta\sqrt{\pi}} \sum_{j=1}^{L} \sum_{i=1}^{M} x_j^i \int_{(j-\frac{1}{2})\Delta t-\beta/2}^{(j+\frac{1}{2})\Delta t+\beta/2} e^{-(\xi-s)^2/\delta^2} \Psi i(s)\, ds. \tag{3.19}$$

If we consider the N-dimensional partitioned vector with components

$$v_j = (v_j^1, v_j^2, \cdots, v_j^M)^*, \qquad j = 1, \cdots, L, \tag{3.20}$$

writing

$$v_j^i = \frac{1}{\delta\sqrt{\pi}} \int_{(j-\frac{1}{2})\Delta t-\beta/2}^{(j+\frac{1}{2})\Delta t+\beta/2} e^{-(\xi-s)^2/\delta^2} \Psi_i(s)\, ds, \qquad i = 1, \cdots, M \tag{3.21}$$

we have, using (3.15),

$$J_\delta q(\xi) = \sum_{j=1}^{L} \sum_{i=1}^{M} x_j^i v_j^j = \sum_{j=1}^{L} (x_j, v_j) \tag{3.22}$$

or

$$J_\delta q(\xi) = (x, v). \tag{3.23}$$

From (3.18), it follows that

$$J_\delta q(\xi) = (Z^{-1}H^*h, v) = (h, HZ^{-1}v) \equiv (h, V). \tag{3.24}$$

The vector $V = HZ^{-1}v$ can be computed and stored once and for all for the time ξ of interest.

Therefore

$$J_\delta q(\xi) = \sum_{k=1}^{K} h_k V_k. \tag{3.25}$$

If our data $\bar{F}$ are measured at a whole long sequence of sample points with equal spacing $\Delta s = \beta/(k-1)$, we can just translate our data set along the t-axis by the multiples $T_j = j\,\Delta s$, j integer, and attempt to reconstruct our linear functional $J_\delta q$ at the new points using the previous weights given by (3.24), if and only if we are able to repeat the conditions for the reconstruction at $\xi = 0$, which requires that $\bar{F}$ be $\equiv 0$ for $t \leqq T_j$.

This can actually be achieved if we subtract the influence upon the data of the last reconstructed point. In doing so, we subtract the influence upon the data of the last $J_\delta q$ instead of q, but this is allowed since A is a smoothing operator and therefore the high frequency components die out very fast in Aq.

Hence, our approximation at any desired point T_j is given by the very cheap and simple discrete convolution against the data sequence, updated as mentioned,

$$J_\delta q(T_j) = \sum_{k=1}^{K} \bar{F}(T_j + d_k)\left(\frac{V_k}{\sqrt{K}}\right). \tag{3.26}$$

Beck's method. In this section we would like to review a numerical method which is commonly in use, due to J. V. Beck [1], [2], [12].

Assuming that the approximate solution has been computed for times $t < T_j$ and that its influence upon the future data has been subtracted to give an adjusted future data $\bar{F}$, as described in the previous section, Beck then considers $q(t)$ to be a low order polynomial over a short portion of the future time record (several time steps in the discretized data). One adjusts the coefficients in the polynomial expansion so that Aq most nearly fits the adjusted future data $\bar{F}$ in the least-squares sense over this short analysis interval. This solution is then taken as the accepted value of $q(t)$ over the T_j single time step only. The analysis interval is then shifted one time step and the entire process is repeated.

If $q(t)$ is a piecewise constant function, and r future temperatures are considered during the entire process, Beck's method reduces to

$$q_M = \frac{\sum_{j=1}^{r+1} \bar{F}_{M+j-1}\phi_j - \sum_{j=1}^{M-1} q_{M-j}(\phi_1 \Delta\phi_j + \cdots + \phi_{r+1}\Delta\phi_{j+r})}{\sum_{j=1}^{r+1} \phi_j^2}, \qquad q_0 \equiv 0. \tag{3.27}$$

Here

$$\bar{F}_M = \bar{F}(M\Delta t), \quad q_M = q(0, (M-\tfrac{1}{2})\,\Delta t), \quad \phi_j = \phi(1, j\Delta t), \quad \Delta\phi_j = \phi_{j+1} - \phi_j.$$

The function $\phi(x, t)$ is the temperature response at point x for a unit step rise for the surface flux, i.e.,

$$\phi(x, t) = 2\pi^{-1/2} t^{1/2}\, e^{-x^2/4t} - x \operatorname{erfc}\left(\frac{x}{2} t^{-1/2}\right) \tag{3.28}$$

with

$$\operatorname{erfc}(x) = 2\pi^{-1/2} \int_x^\infty e^{-s^2}\, ds. \tag{3.29}$$

It has been shown in [1] that in order to have stability and at the same time a relatively good resolution, the product $r\Delta t$ should be taken larger than a certain minimum value and less than approximately 0.3. In this context, $r = 3$ seems to allow the maximum flexibility in the choice of the sample intervals Δt of interest, as determined in [1] and by our own experimentation.

We notice that if $q(t)$ is considered to be a piecewise constant function, the unknown N-dimensional vector x for the mollified method becomes an L-dimensional one, since from (3.15) $N = L \times M = L \times 1$.

Also, (3.16) becomes

$$H_{kj} = K^{-1/2}(\phi(d_k + \beta/2 - (j-\tfrac{1}{2})\Delta t) - \phi(d_k + \beta/2 - (j+\tfrac{1}{2})\,\Delta t)), \tag{3.30}$$

$$k = 1, 2, \cdots, K, \quad j = 1, 2, \cdots, L, \quad \phi \text{ as in (3.28)},$$

and (3.21) is now

$$v_j = \frac{1}{\delta\sqrt{\pi}} \int_{(j-\frac{1}{2})\Delta t - \beta/2}^{(j+\frac{1}{2})\Delta t - \beta/2} e^{-(\xi - s)^2/\delta^2}\, ds, \qquad j = 1, 2, \cdots, L, \tag{3.31}$$

$$v_j = \Phi\left(\frac{(j+\tfrac{1}{2})\Delta t - \beta/2 - \xi}{\sigma}\right) - \Phi\left(\frac{(j-\tfrac{1}{2})\Delta t - \beta/2 - \xi}{\sigma}\right), \tag{3.32}$$

where Φ is the normalized Gaussian distribution function and $\sigma = \delta/\sqrt{2}$.

4. Numerical results.

Direct kernels. Comparison with Beck's method. In order to test the accuracy of our method, we would like, first, to approximately reconstruct a delta function at time $t = 0$ in $u_x(0, t) = q(t)$, by solving the problem

$$\begin{aligned} &u_t = u_{xx}, \qquad 0 < x < \infty, \quad t > 0, \\ &u(1, t) = \bar{F}(t), \quad \text{data}, \\ &u_x(0, t) = \delta_0(t), \quad \text{unknown}, \\ &u(x, 0) = 0, \qquad 0 \leqq x < \infty. \end{aligned} \tag{4.1}$$

We generate the *exact data* as the solution of the well-posed problem

$$\begin{aligned} &u_t = u_{xx}, \qquad 0 < x < \infty, \quad t > 0, \\ &u_x(0, t) = \delta_0(t), \\ &u(x, 0) = 0, \qquad 0 \leqq x < \infty. \end{aligned} \tag{4.2}$$

The solution $u(x, t)$ is the kernel

$$u(x, t) = (\pi t)^{-1/2} e^{-x^2/4t} \tag{4.3}$$

and, in particular,

$$u(1, t) = \bar{F}(t) = (\pi t)^{-1/2} e^{-1/4t}. \tag{4.4}$$

Several numerical solutions of the problem (4.1), the direct kernels, have been computed, using the MM method and Beck's, for different values of the parameters involved. The parameters have the following meaning:

β: Length of data interval.
Δs: Step size sampling for data interval.
Δt: Step size for reconstruction set.
δ: Radius of mollification.

The results are shown in Figs. 2, 3, 4 and 5.

In Table 1 the apparent "support" indicates the interval in which the absolute value of the reconstructed function is greater than 10^{-2}.

In all cases, the maximum value is attained at $t = 0$. Both methods conserve the total integral extremely well since the kernel integrals are practically 1. We also observe that while the direct kernels for the MM method are always positive and symmetric, the direct kernels for Beck's method are slightly nonsymmetric and become partially negative as Δt decreases.

Inverse kernels. Comparison with Beck's method. In order to investigate the stability of our numerical method, we would like to know the amplification factor associated with errors in the data when using the numerical procedures.

We notice that any piecewise constant data function can be expressed by a discrete convolution against the numerical delta function $N(t)$ defined by

$$N(t) = \begin{cases} 1/\Delta s & \text{if } -\tfrac{1}{2}\Delta s \leqq t < \tfrac{1}{2}\Delta s, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Moreover, if we know the solution for the numerical delta function in the data, our inverse kernel, it follows by linearity that the total error in the solution can be obtained as the discrete convolution of the data error against the inverse kernel.

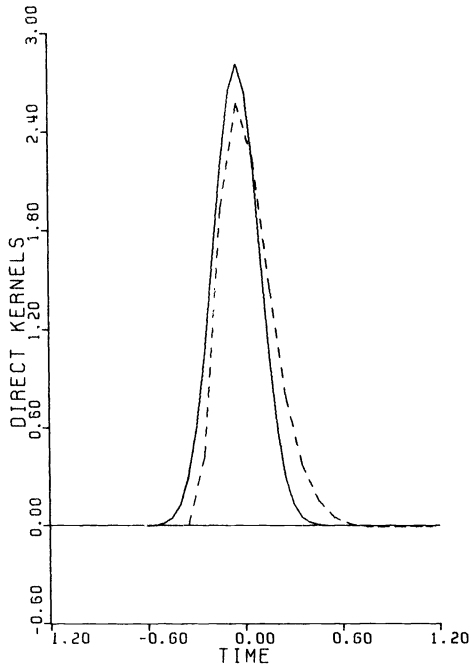


FIG. 2. *Direct kernels. Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = exact data.* ——— MM: $\delta = 4\Delta t = 0.2$; ---- Beck: $2\Delta t = 0.2$.

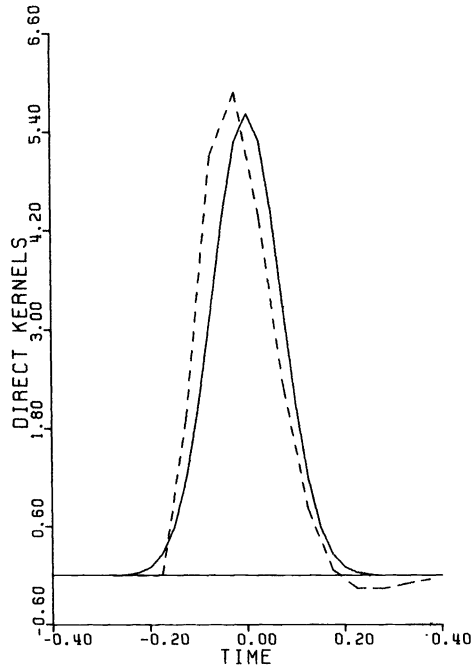


FIG. 3. *Direct kernels. Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = exact data.* ——— MM: $\delta = 4\Delta t = 0.1$; ---- Beck: $2\Delta t = 0.1$.

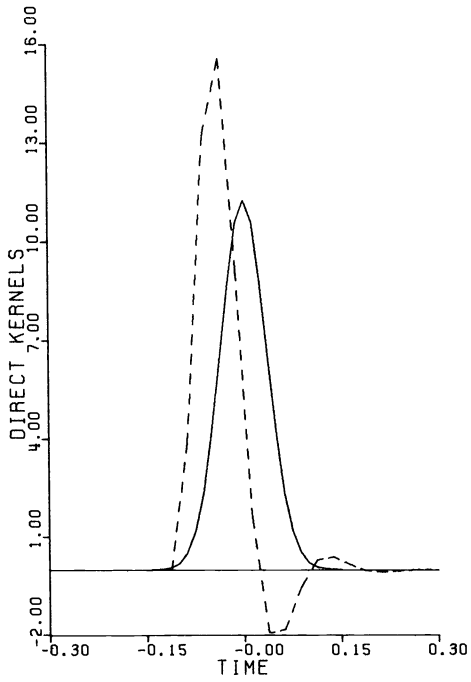


FIG. 4. *Direct kernels. Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = exact data.* ——— MM: $\delta = 4\Delta t = 0.05$; ---- Beck: $2\Delta t = 0.05$.

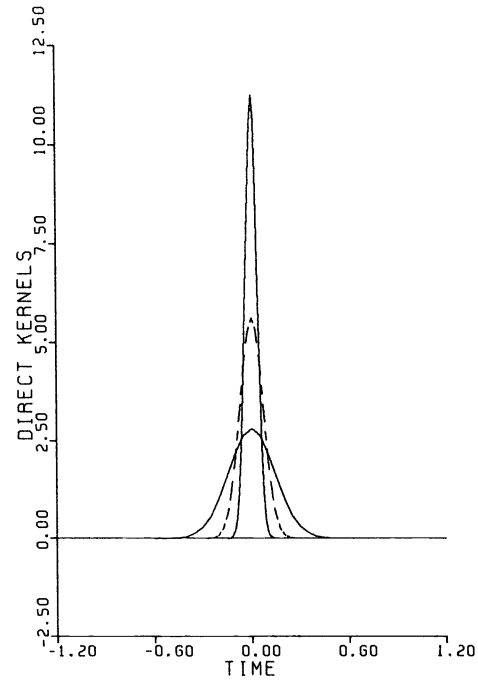


FIG. 5. *Direct kernels. Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = exact data.* MM: $\delta = 4\Delta t = 0.2, 0.1, 0.05$.

TABLE 1
For Figs. 2, 3, 4, 5

	MM	Beck	MM	Beck	MM	Beck
β	2		1		1	
Δt	0.05	0.1	0.025	0.05	0.0125	0.025
Δs	$\Delta t/2$		$\Delta t/2$		$\Delta t/2$	
ε/E	0		0		0	
δ	$4\Delta t$		$4\Delta t$		$4\Delta t$	
Max. Value	2.820	2.552	5.641	5.897	11.280	15.600
"Support"	[−0.6, 0.6]	[−0.3, 1.1]	[−0.3, 0.3]	[−0.2, 0.6]	[−0.15, 0.15]	[−0.12, 0.3]
Integral	0.9995	0.9993	0.9999	0.9999	1.0000	1.0000

If we solve the problem

$$
\begin{aligned}
&u_t = u_{xx}, \qquad 0 < x < \infty, \quad -\infty < t < \infty, \\
&u(1, t) = N(t), \quad \text{data}, \\
&u_x(0, t) = IK, \quad \text{unknown},
\end{aligned} \tag{4.6}
$$

using the MM method and Beck's, we find the inverse kernels shown in Figs. 6, 7 and 8 for different values of the parameters. The analysis of the inverse kernels show that the amplification factor for Beck's method is much larger than ours even though both methods have almost equal resolution according with the direct kernels in Figs. 2, 3 and 4.

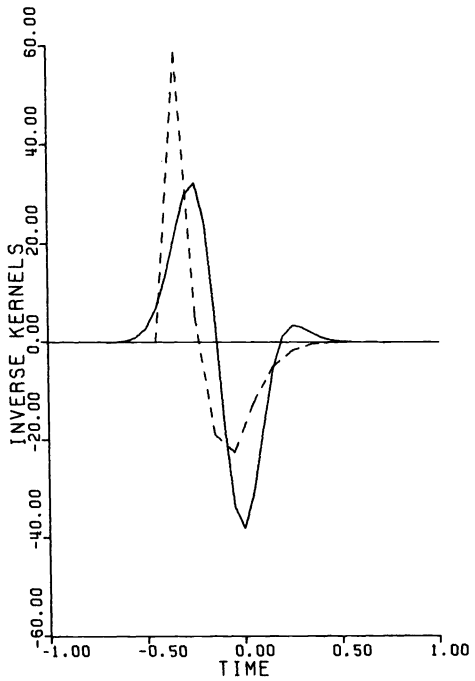


FIG. 6. *Inverse kernels. Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = discrete δ function.*
——— MM: $\delta = 4\Delta t = 0.2$: ---- Beck: $2\Delta t = 0.2$.

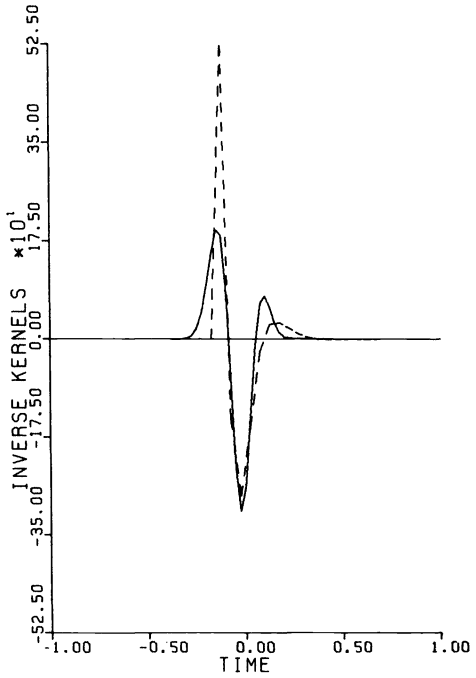


FIG. 7. *Inverse kernels. Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = discrete δ function.* ——— MM: $\delta = 4\Delta t = 0.1$; - - - - Beck: $2\Delta t = 0.1$.

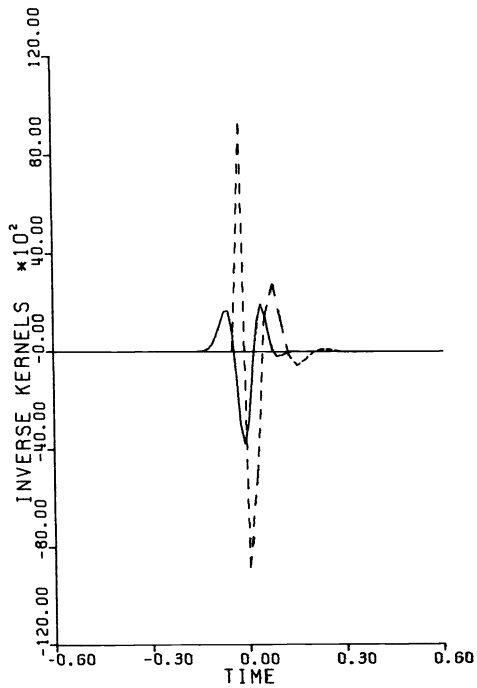


FIG. 8. *Inverse kernels. Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = discrete δ function.* ——— MM: $\delta = 4\Delta t = 0.05$; ---- Beck: $2\Delta t = 0.05$.

TABLE 2
For Figs. 6, 7, 8

	MM	Beck	MM	Beck	MM	Beck
β	2		1		1	
Δt	0.05	0.1	0.025	0.05	0.0125	0.025
Δs	$\Delta t/2$		$\Delta t/2$		$\Delta t/2$	
ε/E	0		0		0	
δ	$4\Delta t$		$4\Delta t$		$4\Delta t$	

The most important feature of the inverse kernels is the fact that they nearly have compact support. This allows us to actually compute the solution by means of the discrete convolution

$$J_\delta q(t) = (IK * \bar{F})(t) \tag{4.7}$$

without needing the history of $\bar{F}$ for very long times either before or after the time t. The inverse kernels are computed once and for all for fixed Δt at the points $s_j = \pm j\,\Delta t$, $j = 0, 1, \cdots, m$, which include the "support" interval of IK. Then if we want to reconstruct the functional $J_\delta q$ at any time $T_i = i\Delta t$, i integer, we read the data in the interval $[T_i - S_m, T_i + S_m]$ and merely use (4.7) in the form

$$J_\delta q(Ti) = \Delta t \sum_{j=-m}^{m} \bar{F}(T_i - j\,\Delta t) IK(j\,\Delta t). \tag{4.8}$$

More numerical results. The solution of the direct problem

$$\begin{aligned} &v_t = v_{xx}, \quad 0 < x < \infty, \quad -\infty < t < \infty, \\ &v_x(0, t) \equiv H(t), \end{aligned} \tag{4.9}$$

the Heaviside function, as data, is given by

$$v(x, t) = \frac{2}{\sqrt{\pi}} \sqrt{t}\, e^{-x^2/4t} - x \operatorname{erfc}\left(\frac{x}{2\sqrt{t}}\right), \qquad t > 0, \tag{4.10}$$

and, in particular,

$$v(1, t) = \frac{2}{\sqrt{\pi}} \sqrt{t}\, e^{-1/4t} - \operatorname{erfc}\left(\frac{1}{2\sqrt{t}}\right), \qquad t > 0. \tag{4.11}$$

We would like to approximately reconstruct a step function in the flux $u_x(0, t) = q(t)$ for $t \in [.1 + \Delta t/2, .3 + \Delta t/2]$, by solving the problem

$$\begin{aligned} &u_t = u_{xx}, \quad 0 < x < \infty, \quad -\infty < t < \infty, \\ &u(1, t) = \bar{F}(t), \quad \text{data}, \\ &u_x(0, t) = H(t - (.1 + \Delta t/2)) - H(t - (.3 + \Delta t/2)), \quad \text{unknown}. \end{aligned} \tag{4.12}$$

Of course, the exact data $F(t)$ are obtained from (4.11); i.e.,

$$F(t) = v(1, t - (.1 + \Delta t/2)) - v(1, t - (.3 + \Delta t/2)). \tag{4.13}$$

The solution of the problem (4.12) has been computed using both the MM with $\Delta t = .0125$, $\Delta s = \Delta t/2$, $\beta = 1$, $\varepsilon/E = 0$, $\delta = 4\Delta t$, and Beck's method with $\Delta t = 0.025$ and $r = 3$. The results are shown in Fig. 9. Our method produces the nearly exact mollified step function as desired. Beck's method produces $\approx 10\%$ overshoot at the leading edge.

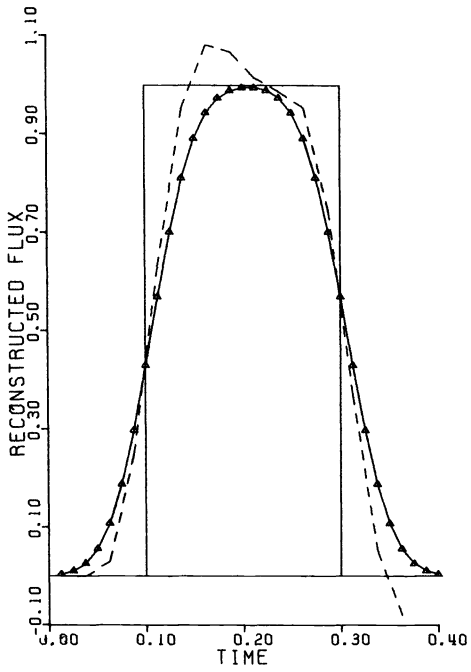


FIG. 9. *Reconstructed flux $q(t)$ corresponding to $\bar{F}(t)$ = exact data.*
——— MM: $\delta = 4\Delta t = 0.05$; ---- Beck: $2\Delta t = 0.05$.

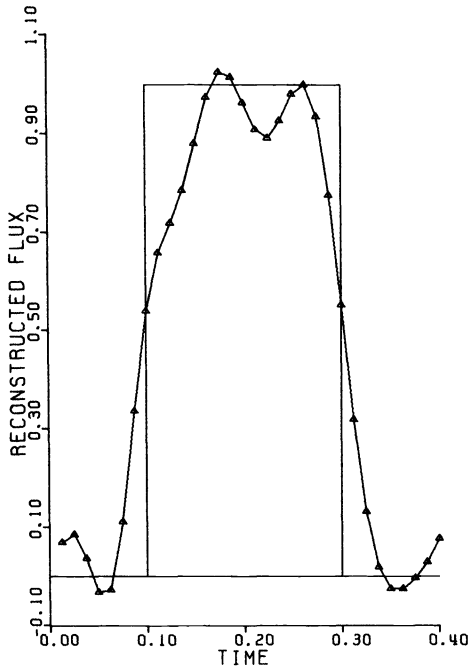


FIG. 10. *Reconstructed flux $q(t)$ corresponding to data with 1% random error.*
MM: $\delta = 4\Delta t = 0.05$.

The solution of problem (4.12) has also been computed using the same parameters but with a 1% random error added to the data; i.e.,

$$\bar{F}(t) = F(t) + .01\theta \max |F(t)|, \tag{4.14}$$

where θ is a random variable with values in $[-1, 1]$, $F(t)$ is given by (4.13) and $\max |F(t)| \approx 0.5$. The MM produces an approximately 10% error as shown in Fig. 10. Beck's method, however, produces oscillations of the order ± 60. This huge amplification of experimental errors is of course predictable because of the large oscillations of the corresponding inverse kernel.

REFERENCES

[1] J. V. BECK, *Calculation of surface heat flux from an internal temperature history*, ASME Paper 62-HT-46, 1962.

[2] ———, *Nonlinear estimation applied to the nonlinear inverse heat conduction problem*, Michigan State University, Department of Mechanical Engineering Report, 1969.

[3] O. R. BURGGRAF, *An exact solution of the inverse problem in heat conduction theory and applications*, J. Heat Transfer, 86C (1964), pp. 373–382.

[4] H. S. CARSLAW AND O. C. JAEGER, *Conduction of Heat in Solids*, 2nd edition, Oxford University Press, London, 1959.

[5] J. M. DAVIES, *Input power determined from temperature in a simulated skin protected against thermal radiation*, J. Heat Transfer, 88C (1966), pp. 154–160.

[6] I. FRANK, *An application of least squares methods . . .*, J. Heat Transfer, 86C (1964), pp. 373–382.

[7] J. HADAMARD, *Le Problème de Cauchy*, Herman et Cie., Paris, 1932.

[8] P. MANSELLI AND K. MILLER, *Calculation of the surface temperature and heat flux on one side of a wall from measurements on the opposite side*, Ann. Mat. Pura Appl., to appear.

[9] K. MILLER, *Least squares method for ill-posed problems with a prescribed bound*, SIAM J. Math. Anal., 1 (1970), pp. 52–74.

[10] K. MILLER AND G. A. VIANO, *On the necessity of nearly-best possible methods for analytic continuation of scattering data*, J. Math. Physics, 14 (1973), pp. 1037–1041.

[11] D. MURIO, *Numerical methods for inverse transient heat conduction problems*, Union Matemática Argentina, to appear.

[12] R. J. MUZZY, J. H. AVILA AND D. E. ROOT, *Determination of transient heat transfer coefficients and the resultant surface heat flux from internal temperature measurements*, General Electric Topical Report GEAP-20731, 1975.

[13] E. M. SPARROW, A. HAJI-SHEIKH AND T. S. LUNDGREN, *The inverse problem in transient heat conduction*, J. Applied Mech., 86E (1964), pp. 309–375.

[14] G. STOLZ JR., *Numerical solutions to an inverse problem of heat conduction*, J. Heat Transfer, 82C (1960), pp. 20–26.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0004$01.00

ON TRACING AN IMPLICITLY DEFINED CURVE BY QUASI-NEWTON STEPS AND CALCULATING BIFURCATION BY LOCAL PERTURBATIONS*

KURT GEORG†

Abstract. An algorithm is presented which traces an implicitly defined curve ($H(x)=0$ for $H:\mathbb{R}^{N+1}\to\mathbb{R}^N$). The Davidenko-IVP is integrated by a self-correcting predictor-corrector method which does not evaluate the Jacobians of H explicitly. Instead, a quasi-Newton method of C. Broyden [Math. Comp., 19 (1965), pp. 577–593] is used in the corrector phase. It is then shown how the algorithm may be used to locate bifurcation points and trace the bifurcating branches by introducing local perturbations of H in the sense of H. Jürgens, H. O. Peitgen, and D. Saupe [in Analysis and Computation of Fixed Points, S. M. Robinson, ed., Academic Press, New York]. Numerical results are reported on a difficult test problem and on the bifurcation of periodic solutions of a differential delay equation.

Key words. curve tracing, homotopy method, continuation method, quasi-Newton update, bifurcation, nonlinear algebraic systems

1. Introduction. Let us assume that $H:\mathbb{R}^{N+1}\to\mathbb{R}^N$ is a (sufficiently) smooth map and zero a regular value of H; i.e., the Jacobian $H'(x)$ has rank N for $H(x)=0$. By a repeated application of the implicit function theorem it may be seen [29] that $H^{-1}(0)$ consists of a disjoint union of simple smooth curves $c(\cdot):\mathbb{R}\to\mathbb{R}^{N+1}$ each being diffeomorphic either to the real line or to a circle.

One such curve, together with a specific parametrization, is singled out by the following properties (C1)–(C4):

(C1) $c(\cdot)$ is a solution curve; i.e., $H(c(s))=0$, $s\in\mathbb{R}$.

(C2) An initial value has to be given, say $c(0)=x$.

(C3) Parametrization is possible according to arc length s; i.e. $|\dot{c}(s)|=1$ for $s\in\mathbb{R}$, where the dot stands for d/ds and $|\cdot|$ indicates Euclidean norm in $\mathbb{R}^{N+1}$.

(C4) An orientation must be given. An easy way to do so is via the determinant condition on the augmented Jacobian,

$$\det\begin{pmatrix}H'(c(s))\\ \dot{c}(s)^T\end{pmatrix}>0 \quad \text{for } s\in\mathbb{R}.$$

Numerically tracing such curves has many important applications, such as the following.

Embedding or continuation methods. See for example the extensive bibliographies in [2], [28], [37], [39], [44]. In these methods, one is merely interested in approximating a point $\bar{x}$ on the curve $c(\cdot)$ which has a certain property, such as $(0,\cdots,0,1)^T\bar{x}=1$, and the curve is only used as an aid for developing an approximation scheme. More generally, one may be interested in finding a point $\bar{x}$ on the curve which is a zero or a local minimal point of a given functional on $c(\cdot)$. This includes turning points, bifurcation points [24], [25], [38] and also the solving of ill-posed problems by the method of regularization [31].

Nonlinear eigenvalue problems. A user is often interested in seeing the dependence of a solution on some given parameter; e.g., one might be interested in approximating a whole branch of solutions.

* Received by the editors February 5, 1980, and in revised form October 30, 1980.

† Colorado State University, Fort Collins, Colorado 80523; University of Bonn, 53 Bonn, W. Germany.
This work was supported in part by Deutsche Forschungsgemeinschaft, SFB 72, Bonn, and the U.S. Air Force Office of Scientific Research under grant AFOSR76-3019.

Usually, the curve $c(\cdot)$ is numerically traced by integrating the so-called Davidenko-IVP [11], which is immediately obtained from (C1)–(C4) by differentiation:

(D) $$\dot{c}(s) = t(s), \quad s \geqq 0, \quad c(0) = x,$$

where $t(s)$ is uniquely defined by

$$H'(c(s))t(s) = 0,$$

$$|t(s)| = 1$$

and

$$\det \begin{pmatrix} H'(c(s)) \\ t(s)^T \end{pmatrix} > 0.$$

Some authors [47]–[51] directly integrate (D) by some IVP-code, e.g., [41]. The drawback to this approach, however, is that these methods are not self-correcting and, once they get away from the curve, they may fail at points of high curvature or at near-bifurcation points. These methods are, however, of interest in the context of homotopy methods, since these methods permit restarting.

Probably C. Haselgrove [18] was the first to see that (D) can be integrated by a predictor-corrector method which takes advantage of the fact that $c(\cdot)$ is defined implicitly:

(PC1) *Predictor step.* Given some points $x_1, \cdots, x_n$ on the curve, a new point y further along the curve is predicted by extrapolation or by using the unit tangents $t(s)$ in (D).

(PC2) *Corrector step.* One chooses an additional functional $\gamma : \mathbb{R}^{N+1} \to \mathbb{R}$ such that the surface $\gamma^{-1}(0)$ contains y and is "sufficiently transversal" to $H^{-1}(0)$. Then, starting from the predicted point y, a Newton method is applied to find a solution x_{n+1} of $H(x) = 0$, $\gamma(x) = 0$. Usually, the surface $\gamma^{-1}(0)$ is taken to be a hyperplane.

For extensive literature on these methods, see [2], [28], [37], [39]. Convergence discussions are fairly classical and may be found in [24], [28], [39].

As is immediately seen, practically all computational expense is paid in the corrector step (PC2). In some cases, e.g., when dealing with discretizations of certain classes of differential equations, the explicit evaluation of the Jacobian H' or a discrete analogue may be comparable to the cost of a few evaluations of H. In most cases, however, one would like to avoid the cost of explicit evaluations of Jacobians. One immediately thinks of updating methods which, especially in higher dimensions, are known to be more efficient than unimproved Newton methods [30]. When moving rather slowly along the curve, the updated "Jacobian" in the last step will have the additional advantage of already being a pretty good starting matrix for the next step, and consequently a few corrector steps, each costing just one evaluation of H, will produce a new point approximately on the curve.

Surprisingly enough, such updating methods do not seem to have been considered very much in the context of curve tracing. In a somewhat different context, F. H. Branin and K. S. Hoo [5] give updating methods some thought but report no numerical experience and seem to have abandoned the idea. C. P. Schmidt [40] uses updating methods for aproximately calculating the unit tangent $t(s)$ in (D). Taking these tangents, an IVP code is applied to integrate the Davidenko equation (D). The numerical example considered in [40] is, however, too simple, and is not convincing that this approach will also succeed in following curves with high curvature, turning points and near-bifurcation points.

In fact, various numerical tests recently performed by the author seem to indicate that updating methods provide no adequate tool for calculating the unit tangent in (D). This may be the reason why updating methods have been given so little consideration in the context of curve tracing. In the absence of an exact Jacobian in (D), it was found to be much safer to approximate the unit tangent by a secant through the last two calculated points.

In § 2 a curve tracing algorithm in the spirit of (PC1)-(PC2) will be presented where the corrector steps are performed by a quasi-Newton method of C. Broyden [6]. An automatic step size control and further items on improving the algorithm will be discussed. In § 3 a rather detailed outline will be given of an algorithm which recently has been used successfully by the author. We want to emphasize, however, that our main objective is to show that the general procedure presented is rather successful, and that much can still be done in order to further improve the efficiency of the algorithm. In § 4 the performance of the algorithm on a rather difficult test problem is reported. In § 5 it is shown how the algorithm can be used to locate bifurcation points and trace new branches. This is done by "switching" local perturbations in the sense of H. Jürgens, H. O. Peitgen and D. Saupe [20]. In § 6 the performance of this technique on bifurcation of periodic solutions of a differential delay equation is reported.

All numerical calculations were performed on a CDC CYBER 172 during a guest visit of the author at Colorado State University.

2. An updating method. Let us begin now to outline a predictor-corrector method in the sense of § 1, (PC1)-(PC2) by incorporating updates for carrying along an approximate Jacobian. In the next section a version of such an algorithm will be described rather precisely, and occasionally we will refer to some statement numbers there in order to make things clearer.

We want to trace a curve $c(\cdot)$ as defined in § 1, (C1)-(C4) and assume that a starting point $x_1 = c(0)$ is somehow given. Without loss of generality, we trace the curve in the positive direction, i.e., $c(s)$ for $s > 0$. Hence, since our aim is to avoid calculations of the Jacobian H', we have to assume that a direction $t_1 \in \mathbb{R}^{N+1}$, $|t_1| = 1$ ($|\cdot|$ indicates the Euclidean norm of $\mathbb{R}^{N+1}$) is given such that $t_1^T \dot{c}(0) > 0$. In fact, t_1 is considered an approximation (our best guess) of the unit tangent $\dot{c}(0)$. Furthermore, a matrix J_1 of N rows and $N+1$ columns has to be given which represents an approximation of the Jacobian $H'(c(0))$, though it turns out in practice that the approximation may be pretty rough. Let us at least assume that the initial augmented "Jacobian,"

$$M_1 = \begin{pmatrix} J_1 \\ t_1^T \end{pmatrix},$$

is nonsingular.

Given a step size $\sigma_1 > 0$, we perform a predictor step

(PS) $$y_1 = x_1 + \sigma_1 t_1.$$

Next we introduce the hyperplane $\gamma_1^{-1}(0)$ through y_1, orthogonal to t_1: $\gamma_1(w) = t_1^T(w - y_1)$, and consider quasi-Newton steps (corrector steps) on the map

$$F_1(w) = \begin{pmatrix} H(w) \\ \gamma_1(w) \end{pmatrix},$$

starting at y_1:

(CS) $$y_{i+1} = y_i - M_i^{-1} F_1(y_i), \qquad i = 1, 2, \cdots.$$

The $(N+1)\times(N+1)$ matrices M_i are subsequently updated by C. Broyden's [6] "good" method:

$$\text{(UD)} \qquad M_{i+1} = M_i + \frac{[F(y_{i+1}) - F(y_i) - M_i(y_{i+1} - y_i)](y_{i+1} - y_i)^T}{|y_{i+1} - y_i|^2}.$$

In actual programming, instead of (UD), the Sherman–Morrison [42] formula is used to directly update $B = M_i^{-1}$ (see § 3, step 12 below).

It is easily seen that all points y_i obtained by the corrector step (CS) lie in the hyperplane $\gamma_1^{-1}(0)$; consequently, the last row t_1^T of M_i is never changed by the update (UD) and the algorithm remains unchanged when the additional functional γ_1 in F_1 is replaced by zero as is done in § 3.

The corrector steps are stopped when a given accuracy is reached, i.e., when $|y_{i+1} - y_i|$ is smaller than a given tolerance. Then the last evaluated point $x_2 = y_{i+1}$ is taken as a new direction $t_2 = (x_2 - x_1)/|x_2 - x_1|$. This new direction replaces the old last row of the matrix M_i, (cf. the update, § 3, step (21)), and we go on with predictor and corrector steps.

Before describing a version of this algorithm in more detail, let us discuss some of its items and related problems.

Step size control. An efficient curve tracing algorithm has to provide an automatic step size control (acceleration or deceleration) according to the curvature and other information such as closeness to other parts of $H^{-1}(0)$. At the moment we are mainly interested in the robustness of the algorithm and therefore offer a rather crude acceleration-deceleration technique, which, however, turns out to work pretty well. If, for various reasons, a predictor step with its subsequent corrector steps is not accepted, we multiply the current step size σ by a deceleration factor β, $0 < \beta < 1$, and try again; if the predictor step with its subsequent corrector steps has been accepted such that a new point approximately on the curve has been found, we multiply the current step length σ by an acceleration factor α, $\alpha > 1$. It turns out that under "normal" conditions a deceleration costs only one evaluation of H, which is a reasonable price to pay.

Nonacceptance of a predictor-corrector cycle. To make the algorithm robust and adaptable also to very difficult problems (see the numerical example in § 4), a list of cases has been made under which a predictor step and its subsequent corrector steps are not accepted and hence deceleration takes place. The cases are:

(a) A corrector step gets too large; i.e., $|y_{i+1} - y_i|$ is greater than a given tolerance (see § 3, step (6)). This is the deceleration which is "normally" observed, and which in most cases costs just one function evaluation H as already mentioned.

(b) Too many corrector steps are needed (cf. § 3, step (9)). This is a rather costly way of decelerating, but will occur only if the conditions along the curve change dramatically, provided the tolerance in (a) is set up in a reasonable way.

(c) The updated matrix becomes almost singular (cf. § 3, steps (11), (20)). This deceleration is very rare, and it simply prevents an unnecessary breaking down of the algorithm.

(d) The updated matrices exhibit a change in sign of determinant (cf. § 3, step (23)). This indicates that the algorithm either jumped over a bifurcation point (see § 5) or is attempting to trace some other parts of $H^{-1}(0)$ in the negative direction. At the moment, we are not willing to accept such steps. In § 5 we will modify this rule.

It should be noted here that the change of determinant of a rank one update is easily calculated by the formula

$$\text{(FD)} \qquad P_+ = P + Puv^T = P(I + uv^T) \Rightarrow \det(P_+) = \det P \cdot (1 + v^T u).$$

This formula is used in § 3, steps (10), (13) and (19), (22).

Improvements. The algorithm described here is only a rather unsophisticated first attempt, though quite successful. There are several items which may result in improving its efficiency, i.e., decrease the number of function evaluations H needed to traverse a certain part of the curve under the condition that the produced points lie within a certain given tolerance on the curve.

A finer acceleration-deceleration technique may be used, which should, however, possess the actual robustness achieved by the present technique. For example, as numerical tests have indicated, it is of rather doubtful use to monitor the step size by some ideal contraction factor or ideal number of iterations, since the updating process may not be able to keep up with the speed of curve tracing (i.e., the step size). This observation also makes it difficult to incorporate step size techniques in the sense of H. J. Wacker [43], [45] or P. Deuflhard [15], especially in those cases where only a few corrector steps are performed and consequently the updated Jacobian is a rather coarse approximation to the "true" Jacobian.

A higher-order extrapolation may be performed in the predictor step. However, one has to observe that a predicted point further away from the curve is no problem for the quasi-Newton method (corrector step) as long as the intersection of the curve with the current hyperplane singles out exactly one point near the predicted one. Hence, higher-order extrapolation may offer less improvement than one expects at a first glance (or indeed perhaps no improvement at all).

Instead of Broyden updates (UD) one may use a blending of Broyden updates and secant methods as discussed by D. M. Gay and R. B. Schnabel [16]. Preliminary tests indicate that some 10–20% function evaluations of H may be saved that way. More generally, instead of Broyden updates, one may use some other least change secant update in the sense of J. E. Dennis and R. B. Schnabel [14]. Thus, it would be possible to save much computational expense in case of symmetry (e.g., if N coordinates of H form a gradient) or sparseness of H' (e.g., if H corresponds to a discretization of a boundary value problem).

One may be tempted to reduce the accuracy demanded for stopping the corrector steps (cf. § 3, step (7)) in order to save function evaluations. However, one should wait at least until the quasi-Newton method shows good contractions in order to allow a sufficiently updated "Jacobian," and from then on additional steps increase the accuracy considerably. Hence one can afford to be rather generous here in terms of approximating points on the curve quite precisely.

Convergence. As has already been done in the case of Newton steps [24], [28], [39], it should be possible to prove the convergence of the present algorithm under reasonable assumptions, i.e., to show that a finite part of the curve is approximated within a given accuracy provided the maximal step size is small enough and the initial data t_1, J_1 are sufficiently close to the unit tangent $\dot{c}(0)$ and the Jacobian $H^1(c(0))$, respectively, at the start. One may use here the existing proofs for the convergence of quasi-Newton methods (see, e.g., [7], [8], [12], [13], [37]). Details are currently being worked out and will be reported elsewhere.

3. Description of an algorithm for numerically tracing an implicitly defined curve. Suppose that $H: \mathbb{R}^{N+1} \to \mathbb{R}^N$ is a sufficiently smooth map and zero a regular value of H. Given an initial point $x \in \mathbb{R}^{N+1}$ such that $H(x) = 0$, there is exactly one smooth curve $c(\cdot): \mathbb{R} \to \mathbb{R}^{N+1}$ with the properties in § 1, C1–C4. Our algorithm, sketched in the sequel, traces this curve by sequentially generating points approximately on the curve. The aim here is not to offer a perfect computer program but to point out the main steps of the algorithm. The statements, numbered from (1) to (25), are written in the mentality of programming languages. In particular, the equality sign does not indicate a

mathematical equation. Hopefully, this makes the logic of the algorithm easier to understand and also provides some first basis for a computer program.

Start. Before starting the algorithm one has to define some values for monitoring it:

σ_0 = minimal step length along the curve $(\sigma_0 > 0)$

σ_∞ = maximal step length $(\sigma_0 < \sigma_\infty)$

κ_0 = minimal corrector step length $(\kappa_0 > 0)$

κ_∞ = maximal corrector step length $(\kappa_0 < \kappa_\infty)$

α = acceleration factor $(\alpha > 1)$

β = deceleration factor $(0 < \beta < 1)$

n_∞ = maximal number of corrector steps

δ_0 = minimal factor for updating the determinant $(\delta_0 > 0)$

Finally, a map $H : \mathbb{R}^{N+1} \to \mathbb{R}^N$ has to be given in the form of a subroutine. For technical reasons we introduce the short notation

$$H_0(\cdot) = \begin{pmatrix} H(\cdot) \\ 0 \end{pmatrix} \in \mathbb{R}^{N+1} \quad \text{and} \quad e = (0, \cdots, 0, 1)^T \in \mathbb{R}^{N+1}.$$

Initial data. Some initial values have to be loaded.

x = an initial point on the curve, i.e., $x \in \mathbb{R}^{N+1}$, $H(x) = 0$, $c(0) = x$.

t_A = predictor direction, i.e., $t_A \in \mathbb{R}^{N+1}$, $|t_A| = 1$, and t_A not orthogonal to $\dot{c}(0)$. However, t_A need not be a "good" approximation of $\pm\dot{c}(0)$.

A = a nonsingular $(N+1) \times (N+1)$ matrix such that the last row of A^{-1} coincides with t_A^T. In case of difficulties, one may use an update in the sense of (21) below to obtain this property. The inverse A^{-1} need not be a "good" approximation of the augmented Jacobian

$$\begin{pmatrix} H'(x) \\ t_A^T \end{pmatrix}.$$

$\delta_A = \det(A^{-1})$.

σ = initial step length $(\sigma_0 < \sigma < \sigma_\infty)$.

We now list the steps of the algorithm.

Steps. The following PRINT statement is only included in order to make it clearer when a step is completed and a new point on the curve approximated.

(1) Print x.

Predictor step.

(2) Let $y = x + \sigma t_A$.

The integer n counts the number of corrector steps.

(3) Let $n = 1$.

The next statement is introduced in order to indicate that A and δ_A should be saved in case the predictor-corrector cycle is not accepted.

(4) Let $B = A$ and $\delta_B = \delta_A$.

Next, the corrector step is introduced. Since the last row of B^{-1} coincides with t_A^T, it is easily seen that the corrected point z lies on the same hyperplane defined by t_A as the predicted point y; i.e., $t_A^T(z - y) = 0$.

(5) Let $z = y - BH_0(y)$.

If the corrector step is too large, the predictor-corrector cycle is not accepted.

(6) If $|z-y|>\kappa_\infty$ go to (15).

If the corrector step is too small, we accept the result as a new point on the curve.

(7) If $|z-y|<\kappa_0$ go to (18).

Otherwise, i.e., if $\kappa_0 \leqq |z-y| \leqq \kappa_\infty$, we continue with the corrector steps.

(8) Let $n=n+1$.

In case too many corrector steps are required, the predictor-corrector cycle is not accepted.

(9) If $n>n_\infty$ go to (15).

We calculate the factor by which the determinant of B^{-1} is multiplied in an updating step (see (13)).

$$\text{(10)} \quad \text{Let } \delta_F = \frac{(z-y)^T B(H_0 z - H_0 y)}{|z-y|^2}.$$

If this factor is very small, the predictor-corrector cycle is not accepted.

(11) If $|\delta_F|<\delta_0$ go to (15).

Otherwise, we update B by the values of H on y and z. Note that the old and the new B^{-1} have the same last row t_A^{-1}.

$$\text{(12)} \quad \text{Let } B = B - \frac{[B(H_0 z - H_0 y)-(z-y)](z-y)^T B}{(z-y)^T B(H_0 z - H_0 y)}.$$

If δ_B is the old determinant of B^{-1}, the new one is obtained by

(13) Let $\delta_B = \delta_B \delta_F$.

A new corrector step is performed by

(14) Let $y=z$ and go to (5).

In case the predictor-corrector cycle is not accepted, we decelerate.

(15) Let $\sigma = \beta\sigma$.

If the step size gets too small, the algorithm failed.

(16) If $\sigma<\sigma_0$ then STOP "Step size too small".

Otherwise, we try the smaller predictor step.

(17) Go to (2).

Now we have the case that the last corrector step was so small that we would like to accept z as a new point on the curve. However, some last tests are necessary. The new predictor direction would be

$$\text{(18)} \quad \text{Let } t_B = \frac{z-x}{|z-x|}.$$

We calculate the factor by which the determinant of B^{-1} is multiplied in a last updating (see (21) below).

(19) Let $\delta_F = t_B^T B e$.

If this factor is very small, the predictor-corrector cycle is not accepted.

(20) If $|\delta_F|<\delta_0$ go to (15).

Otherwise, we substitute the last row of B^{-1} by t_B^T. This can be done by the following update:

$$\text{(21)} \quad \text{Let } B = B - \frac{Be(B^T t_B - e)^T}{t_B^T B e}.$$

If δ_B is the old determinant of B^{-1}, the new one is obtained by

(22) Let $\delta_B = \delta_B \delta_F$.

One last test has to be performed. We started with a matrix A (det $A^{-1} = \delta_A$) and now have a matrix B (det $B^{-1} = \delta_B$). If the signs of δ_A and δ_B are different, this indicates that the algorithm attempts to trace a part of $H^{-1}(0)$ in the negative direction (cf. § 2, (d)).

(23) If $\delta_A \delta_B < 0$ go to (15).

Otherwise we accept z as a new point on the curve.

(24) Let $A = B$, $x = z$, $t_A = t_B$, $\delta_A = \delta_B$.

Since the predictor-corrector cycle was successful, we try a bigger predictor step.

(25) Let $\sigma = \min(\sigma_\infty, \alpha \cdot \sigma)$ and go to (1).

Stopping criteria are provided according to the user's preferences.

Remarks. As has already been pointed out, at the moment we are mainly interested in the robustness of the algorithm. Hence, the step size control is essentially given by tests to detect "dangerous" situations. Consequently, much can be done to ensure a "faster" traversing of those parts of the curve which are "nice". Different strategies and step size controls are presently being tested. Let us point out two questions specifically.

(a) The choice of the acceleration-deceleration factors α, β is quite arbitrary and has to be studied in more detail. Generally, it will depend also on the maximal number of corrector steps. In the example below we took $\alpha = 2^{1/4}$ and $\beta = \frac{1}{2}$, which just means that 4 accelerations compensate one deceleration (halving). To increase the efficiency of the algorithm, an update procedure for α and β in the sense of P. Deuflhard [15] is presently being tested, but satisfactory results are not yet available.

(b) Step (4) implies that two matrices are to be stored at all times, which seems to be necessary for performing the special acceleration-deceleration tests given here. If this turns out to induce severe storage requirements, these tests should be changed in such a way that only one current approximation of the Jacobian is needed. A version of such algorithm is presently being tested and seems to give satisfactory results.

4. A numerical example. Let us describe the performance of the above algorithm on an extremely difficult numerical example which has been studied by L. T. Watson [2], [49]. We identify $\mathbb{R}^{N+1}$ with $\mathbb{R}^N \times \mathbb{R}$ ($x = (y, \lambda)$) and consider the operator $F: \mathbb{R}^N \to \mathbb{R}^N$ defined by

$$\phi_k(y) = \exp\left(\cos\left(k \cdot \sum_{i=1}^{N} \eta_i\right)\right),$$

where $y = (\eta_1, \cdots, \eta_N)^T$ and $F(\cdot) = (\phi_1(\cdot), \cdots, \phi_N(\cdot))^T$. Since F is bounded, by Brouwer's theorem it is easily seen that F has a fixed point, but its approximation is complicated due to the highly oscillatory nature of F. We refer in the following to the case $N = 10$.

The idea is to set up a homotopy $H: \mathbb{R}^{N+1} \to \mathbb{R}^N$ by $H(x) = H(y, \lambda) = y - \lambda F(y)$, and follow the curve in $H^{-1}(0)$, beginning at $x = (y, \lambda) = 0$, in the positive λ-direction until the level $\lambda = 1$ is hit.

The monitoring parameters (see § 3, Start) were chosen in the following way: $\sigma_0 = .0001$, $\sigma_\infty = 1.$, $\kappa_0 = .00001$, $\kappa_\infty = .1$, $\alpha = 2^{-4}$, $\beta = .5$, $n_\infty = 12$, $\delta_0 = .001$. Initial data: $x = 0$, $t_A = (0, \cdots, 0, 1)^T$, $A = \mathrm{Id}$, $\sigma = .1$. The algorithm was stopped in case $e^T x > 1$, i.e., $\lambda > 1$.

All points on the curve were approximated within an accuracy of about .00001 (i.e., $\kappa_0 = .00001$), and the norm of H on these points never exceeded .00001.

To follow the whole curve, 5,936 function evaluations H were necessary. Table 1 shows some interesting points on the curve. The current arc length is used as a counter to mark a point on the curve. The most frequent step sizes lay between .1 and .2. If we

TABLE 1

Arc length	Step size	Level λ
2.497466	.141421	.148741
3.277641	.025000	.170660
6.776182	.200000	.221610
6.942125	.001563	.203722
11.946218	.237841	.345137
12.180036	.014865	.337824
25.913004	.237841	.238765
25.982551	.000195	.231201
29.744268	.400000	.575842
31.557443	.059460	.536515
53.618232	.475683	.752125
56.175174	.084090	.841209
58.216670	.237841	.662295
58.590599	.000465	.610784
61.600321	.475683	.878493
62.621013	.070711	.834151
72.077008	.565685	.874211
72.540371	.070711	.848342
80.250663	.475683	.963553
80.398535	.141421	.959610
86.541966	.282843	.944249
86.584322	.042045	.947475
87.209329	.141421	.992502
87.377694	.168179	1.003455

take into consideration that a corrector step up to length .1 ($\kappa_\infty = .1$) was allowed and that the total arc length was about 72, this gives some idea of the high curvature the solution curve has practically everywhere. Most frequently, deceleration by corrector step length occurred (§ 3, step (6)). Only at very small step sizes ("danger") other deceleration tests (e.g., § 3, steps (9), (11), (20), (23)) became active. The smallest observed step size was .000195 (at arc length 25.982551). Here, without the sign-of-determinant test (§ 3, step (23)), the algorithm would have jumped onto some other part of $H^{-1}(0)$ and would have proceeded in the reverse direction. The largest step length, .565685, was observed at arc length 72.077008.

We emphasize here that our aim was not to approximate a fixed point of F by a minimal number of steps but to show that our algorithm was able to safely follow the curve. L. T. Watson, using an IVP code as already mentioned above, does not follow the whole curve but stops if the step size gets too small and restarts in $x = (y^0, 0)$ with a new homotopy $H(y, \lambda) = y - \lambda F(y) - (1-\lambda)y^0$, where (y^0, λ^0) is the last approximated point on the old curve. He thus was able to calculate a fixed point of F with 4,644 Jacobian evaluations (see [2], [49]). A different approach is used by P. Brandenburg [4], applying a numerically stable simplicial deformation algorithm, proposed by the author [17], directly on the zero problem $y - F(y) = 0$. He uses 4,547 evaluations of H to approximate a fixed point of F very precisely. However, it should be mentioned that no curve $c(\cdot)$ in the sense of § 1: C1–C4 is followed.

Generally, it is not possible to confront the amount of computer work induced by an evaluation of a Jacobian with the corresponding function evaluation (of H, in this case), since this varies considerably. Nevertheless, it has become customary for test problems to count one Jacobian evaluation as N (= dimension) function evaluations, if no special structure (like sparseness) is taken into account.

5. Calculating bifurcation by local perturbations. Let us now show how the above derivative-free algorithm may be used to calculate bifurcation points and additional branches. The approach given here has been inspired by the use of Sard's theorem in [9], [27] and is a variant of one of the topological perturbation devices studied in [20] (see also [33]).

Again, we consider a sufficiently smooth map $H: \mathbb{R}^{N+1} \to \mathbb{R}^N$ and a smooth solution curve $c: \mathbb{R} \to \mathbb{R}^{N+1}$ parametrized, say, according to arc length; i.e., $H(c(s)) = 0$ and $|\dot{c}(s)| = 1$ for $s \in \mathbb{R}$. Suppose that every point $x \in H^{-1}(0)$ is regular; i.e., the Jacobian $H'(x)$ has rank N, with the exception of one point: $c_0 = c(0)$. We consider again the augmented Jacobian

$$M(s) = \begin{pmatrix} H'(c(s)) \\ \dot{c}(s)^T \end{pmatrix}$$

which consequently is singular only for $s = 0$.

LEMMA 1. *If* $\det(M(s))$ *changes sign at* $s = 0$ *then* c_0 *is a bifurcation point.*

This is a well-known result and may be proved by a degree argument; cf. P. H. Rabinowitz [36]. It also could be deduced from our discussion below.

The case considered here is more general than the one considered by M. G. Crandall and P. H. Rabinowitz [10], even if zero is a simple eigenvalue of $M(0)$. In fact, they impose an additional transversality condition in order to obtain a local parametrization of the bifurcating curve. A constructive analogue of their discussion has been given by H. B. Keller [24], [25] (see also [22], [23], [26]), who indeed numerically approximates the bifurcation point and the new branch. H. B. Keller uses, however, not only the Jacobian H' but even a higher derivative H''. W. C. Rheinboldt [38], in a related approach, avoids the computation of H'' by using a singular chord method and thus is able to switch to the new branch by just one evaluation of the Jacobian H' and several evaluations of H.

Our approach will be totally different. We will apply the above algorithm together with a local perturbation to switch [20] to the new branch by using only a few evaluations of H. Furthermore, we do not need a good approximation of the bifurcation point. In fact, since our method is derivative-free, the only way to precisely approximate the bifurcation point (if it is wished to do so) is by means of curve tracing. The basis of our considerations is the following observation.

LEMMA 2. *For a given open bounded neighborhood* V *of* c_0 *(e.g., an open ball), let* $\zeta: \mathbb{R}^{N+1} \to \mathbb{R}$ *be a sufficiently smooth function such that* (i) $\zeta(x) = 0$ *for* $x \notin V$ *and* (ii) $\zeta(x) > 0$ *for* $x \in V$. *Then for almost all* $d \in \mathbb{R}^{N+1}$ *the map* $H_d: \mathbb{R}^{N+1} \to \mathbb{R}^N$, *defined by* $H_d(x) = H(x) - \zeta(x)d$, *has zero as a regular value.*

The proof is given by using a parametrized version of Sard's theorem; cf. [1], [9]. For $x \notin V$, the Jacobian of $H_d(x)$ with respect to x coincides with the Jacobian $H'(x)$, and hence has rank N provided that $x \in H_d^{-1}(0)$. For $x \in V$, the Jacobian of $H_d(x)$ with respect to d is a positive multiple of the identity map on $\mathbb{R}^N$, and hence again has rank N. Thus the total derivative of the map $(x, d) \to H_d(x)$ has rank N at all points (x, d) such that $H_d(x) = 0$, and the conclusion follows from the above mentioned Sard's theorem.

Now, note the fact that outside V the maps H_d and H coincide. If $p: \mathbb{R} \to \mathbb{R}^{N+1}$ is a zero-curve of H_d, due to the fact that zero is a regular value of H_d, the augmented Jacobian

$$\begin{pmatrix} H_d'(p(s)) \\ \dot{p}(s)^T \end{pmatrix}$$

does not change sign of determinant. Hence, if a solution curve $p(\cdot)$ of H_d, which in the beginning coincides with $c(\cdot)$, is traced into V, then after leaving V the curve $p(\cdot)$

cannot coincide with $c(\cdot)$ anymore but must represent a new branch coming from the bifurcation point.

It is now easy to sketch how our above algorithm should be applied to calculate a new branch. Suppose one numerically traces a solution curve $c(\cdot)$ by generating points $x_1, x_2, \cdots, x_k, x_{k+1}, \cdots$ approximately on $c(\cdot)$, and suppose that during the step $x_k \to x_{k+1}$ one encounters a change of sign in the determinants (see the test (23) in § 3, which in this case obviously should not be used as a deceleration test but rather as a bifurcation indicator). One then may stop the algorithm, define the open ball V around the center $\frac{1}{2}(x_k + x_{k+1})$ with radius $\frac{1}{2}|x_{k+1} - x_k|$ and perform a local perturbation H_d in the sense of Lemma 2. Using this new map H_d and the negative last direction, i.e., $-(x_{k+1} - x_k)/|x_{k+1} - x_k|$, one now continues the algorithm with a smaller step size, say $\frac{1}{4}|x_{k+1} - x_k|$, into V. After leaving V, the algorithm traces a new solution which has branched off at a bifurcation point somewhere in V. We emphasize that this numerical procedure is extremely simple and, as the example in § 6 shows, uses only a few evaluations of H or H_d.

It should be pointed out here that bifurcation points will indeed be detected numerically even in our case, where instead of the exact Jacobian we are merely dealing with approximate Jacobians given by means of Broyden updates. This is due to the fact that we consider only bifurcation points characterized in a "stable" way by a change of sign of the augmented Jacobian as described above. Clearly, the step size should be "small" with respect to the curvature of $c(s)$, and it has to be studied more precisely what this means numerically.

Though the method sketched here permits that the change of sign is reflected in the approximate augmented Jacobian only some steps later, numerical experience indicates that the change-of-sign test (23) is very sensitive and usually gets activated immediately after a bifurcation point of the above-mentioned type has been encountered.

If one wishes to approximate the bifurcation point more precisely (e.g., the center $\frac{1}{2}(x_k + x_{k+1})$ is a very rough guess) one may stop again after leaving V, and turn back, using now the unperturbed map H. One thus generates a sequence $\tilde{x}_1, \tilde{x}_2, \cdots, \tilde{x}_l, \tilde{x}_{l+1}, \cdots$ for which, somewhere in V, one again encounters a change of sign of the determinants, say, during the step $\tilde{x}_l \to \tilde{x}_{l+1}$. One then may interpolate the points x_{k-1}, x_k, x_{k+1} (respectively, $\tilde{x}_{l-1}, \tilde{x}_l, \tilde{x}_{l+1}$) by two polynomials $p(t)$ (respectively, $\tilde{p}(\tilde{t})$) of degree 2 with coefficients in $\mathbb{R}^{n+1}$ and calculate a local minimum point of $(t, \tilde{t}) \to |p(t) - \tilde{p}(\tilde{t})|$. The interpolation and minimization costs just some vector additions and multiplications and a few Newton iterations in $\mathbb{R}^2$. Certainly, if one wishes, it is possible to use a more sophisticated curve-fitting procedure instead of the above rather simple-minded interpolation of degree 2, which, however, in most cases will be sufficiently accurate. Details will appear elsewhere.

Thus, the method of local perturbations together with our algorithm may be used to trace solution curves, switch branches at bifurcation points and accurately approximate a bifurcation point. The present method will be particularly efficient when implemented as an interactive computer program, allowing a user to trace solution branches according to the information he gets. Before we report in § 6 in some detail on a numerical example, let us illustrate the effect of local perturbation by two very simple examples.

Example 1. Consider the map $H : \mathbb{R}^2 \to \mathbb{R}$ defined by $H(x, y) = xy$. $H^{-1}(0)$ consists of the x- and y-axis, and 0 is a bifurcation point. If we define orientation by the augmented Jacobian as indicated in § 1, C4, we get a picture as illustrated in Fig. 1. Let us now introduce a local perturbation H_d. In order to simplify the presentation, we do not take a smooth perturbation; however, H_d may be regarded as a limit of such smooth

perturbations.

$$H_d(x, y) = \begin{cases} xy & \text{if } x^2+y^2 \geqq 1, \\ xy+\frac{1}{2}(1-x^2-y^2) & \text{if } x^2+y^2<1, \end{cases}$$

where $d \in \{+1, -1\}$ is chosen. It is easily seen that the solution curves of H_d now have a form as illustrated in Fig. 1.

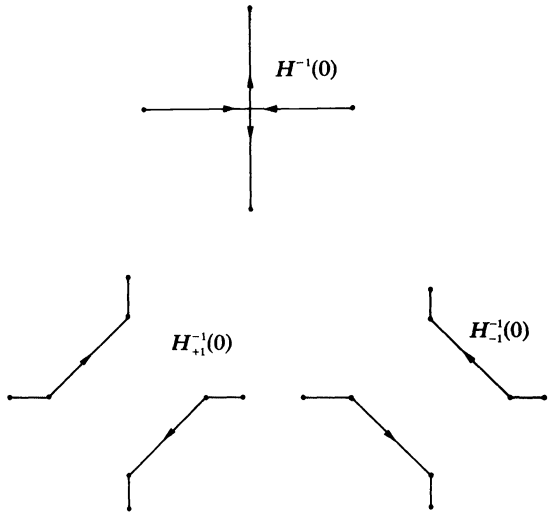


FIG. 1

Example 2. Consider the map $H: \mathbb{R}^2 \to \mathbb{R}$ defined by $H(x, y) = xy^3 - yx^3$. $H^{-1}(0)$ consists of the four lines $x = 0$, $y = 0$, $x = y$, $x = -y$, and the orientation is indicated in Fig. 2. To avoid unnecessary calculations, since we are only interested in a qualitative picture, we consider the following point-to-set map:

$$H_d(x, y) = \begin{cases} \{xy^3 - yx^3\} & \text{for } |x|+|y|>1, \\ \text{convex hull of } \{xy^3 - yx^3, d\} & \text{for } |x|+|y|=1, \\ \{d\} & \text{for } |x|+|y|<1, \end{cases}$$

which may be regarded as an (appropriate) limit of smooth perturbations. By $H_d^{-1}(0)$ we mean $\{(x, y): 0 \in H_d(x, y)\}$. Fig. 2 illustrates these "perturbed" zero-sets.

6. A numerical example. As a numerical example, we consider a differential delay equation. Our aim here is not to give a thorough numerical study of such equations but merely to illustrate that the ideas of § 5 can be successfully applied. Differential delay equations, and in particular bifurcation of such equations, have been numerically studied with great success by applying simplicial curve-tracing methods (see [19], [20], [33]).

Let us consider the following delay equation:

(E) $x'(t) = -\lambda f(x(t-1))$, where x is defined for $t \geqq 0$ and the equation is supposed to hold for $t > 1$.

Given a continuous function $x: [0, 1] \to \mathbb{R}$, equation (E) uniquely extends x to a function $x: [0, \infty) \to \mathbb{R}$. We are interested here in periodic solutions, i.e., $x(t) = x(t+t_0)$ for all $t \geqq 0$ and some period $t_0 > 0$. To simplify the discussion, let us make the following quite

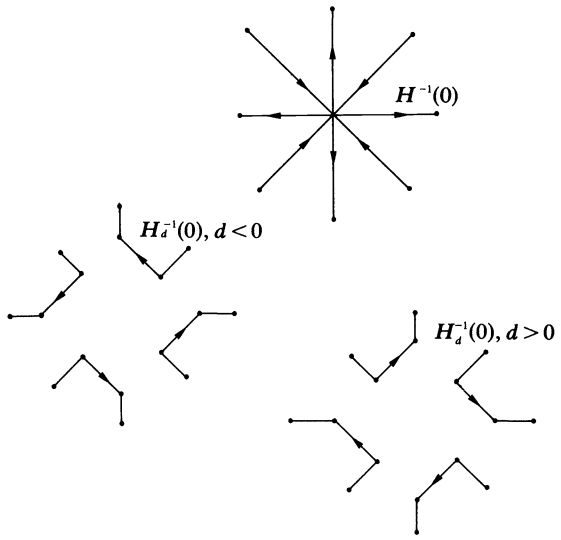


FIG. 2

strong assumptions on f:

(A) Suppose that $f:\mathbb{R}\to\mathbb{R}$ is continuous, odd (i.e., $f(x)=-f(-x)$, $x\in\mathbb{R}$), differentiable at 0 with $f'(0)=1$, and $f(x)x>0$ for $x\neq 0$.

Obviously, $x(\cdot)=0$ is a periodic solution of (E) for all λ, the so-called line of trivial solutions. It can be shown ([21], [32]) that at $\lambda=\pi/2$ a branch of nontrivial periodic solutions bifurcates off the trivial line. Actually, given the oddness of f, one may restrict one's attention to higher symmetry solutions of period 4, which can be obtained by the following simpler integral equation on $C[0, 1]$:

$$\text{(I)}\qquad x(t)+\lambda\int_{1-t}^{1} f(x(s))\,ds=0.$$

(See [3], [20] for more details.) In our numerical example, we took the nonlinearity

$$f(x)=\begin{cases}\tan(x) & \text{for } |x|<1,\\ \tan(\operatorname{sign}(x)) & \text{for } |x|\geqq 1,\end{cases}$$

which causes bifurcation to the left (see [46]). We discretized x on 20 points and integrated by Simpson's rule, thus obtaining an approximation of (I) of the following form:

$$\text{(Z)}\qquad H(x)=0,\quad \text{where } H:\mathbb{R}^{21}\to\mathbb{R}^{20}.$$

The eigenvalue parameter λ is identified with the last coordinate of x.

We chose an open ball V with radius 1 around the bifurcation point $x_0=(0,\cdots,0,\pi/2)$ and considered the local perturbation

$$\text{(P)}\qquad H_d(x)=H(x)+5[(1-|x-x_0|)^+]^2,\quad \text{where } \alpha^+=\begin{cases}\alpha & \text{for } \alpha\geqq 0,\\ 0 & \text{for } \alpha<0,\end{cases}$$

which is not very smooth, but was sufficient for our purposes.

Phase I. We started the algorithm outside V on the trivial line to the left of $\pi/2$ with local perturbation switched "on", ran through V and left on a nontrivial branch of solutions, which we traced until $\lambda\approx\pi$.

Phase II. We then turned backwards with local perturbation switched "off," ran through V, jumped over the bifurcation point (change in sign of determinant!) and kept on until again $\lambda \approx \pi$.

Phase III. We turned back again with local perturbation switched "on", ran through V and ended on the trivial line, however, now to the right of $\lambda = \pi/2$.

TABLE 2

λ	Norm	det	nH	$\in V$?
Phase I		Perturbation Switched "on"		
.564542	.000000	1.00	—	no
.669299	.207494	.63	6	yes
.730894	.342399	4.56	5	,,
.844784	.520616	4.73	4	,,
1.059703	.734074	4.85	5	,,
1.176255	.812746	4.68	8	,,
1.370601	.900836	3.54	5	,,
1.513161	1.185213	.98	9	no
1.497593	1.377060	1.52	8	,,
1.477227	1.532187	1.27	5	,,
1.443739	1.793451	1.58	5	,,
1.386802	2.161622	1.87	4	,,
1.286451	2.678963	2.54	5	,,
1.112223	3.403504	3.65	6	,,
1.126755	3.752345	3.08	7	,,
1.209609	4.218078	4.34	6	,,
1.348957	4.866837	4.45	5	,,
1.550629	5.782416	4.64	5	,,
1.765242	6.753571	4.50	4	,,
1.987587	7.724993	4.38	5	,,
2.209123	8.696390	4.42	4	,,
etc.				
Phase II		Perturbation switched "off"		
1.443111	1.797965	−1.42	–	,,
1.542055	.799152	−.35	6	yes
1.564386	.203581	.20	7	,,
1.555164	.537202	.44	5	,,
1.527851	1.008033	.88	4	no
Phase III		Perturbation switched "on"		
1.415781	1.983624	−1.89	–	,,
1.534186	.987081	−.91	6	yes
1.591989	.956899	−1.55	8	,,
1.640919	.940542	−3.54	5	,,
1.709551	.918142	−3.82	4	,,
1.805648	.882667	−5.77	4	,,
1.937740	.820506	−5.58	5	,,
2.112331	.704692	−5.95	5	,,
2.321663	.484422	−5.05	5	,,
2.516619	.087802	−1.72	10	,,
2.786151	.000002	−2.15	12	no

In Table 2 we show the more interesting steps of the algorithm in the three phases. The numerical effort to generate a new point is represented by the number nH of evaluations of H or H_d, each nH corresponding approximately to .035 CPU-seconds. The points on the curve were approximated within an accuracy of .00001 (Euclidean norm), and the norm of H on these points never exceeded .00001. The norm in Table 2 is taken over the first 20 coordinates of x (Euclidean norm), i.e., without λ.

Acknowledgment. I am very grateful to Eugene L. Allgower for numerous discussions.

REFERENCES

[1] R. ABRAHAM AND J. ROBBIN, *Transversal Mappings and Flows*, W. A. Benjamin, New York, 1967.

[2] E. L. ALLGOWER AND K. GEORG, *Simplicial and continuation methods for approximating fixed points and solutions to systems of equations*, SIAM Rev., 22 (1980), pp. 1–58.

[3] N. ANGELSTORF, *Global branching and multiplicity results for periodic solutions of functional differential equations*, in Functional Differential Equations and Approximation of Fixed Points, H.-O. Peitgen and H. O. Walther, eds., Lecture Notes in Mathematics 730, Springer, New York, 1979, pp. 32–45.

[4] P. BRANDENBURG, *Ein simplizialer modifizierter Algorithmus mit kontinuierlich kleiner werdender Gitterweite und Anwendung*, Diplomarbeit, Bonn, 1979.

[5] F. H. BRANIN, JR. AND K. S. HOO, *A method for finding multiple extrema of a function of N variables*, in Numerical Methods for Nonlinear Optimization, F. Lootsma, ed., Academic Press, New York, 1972, pp. 231–237.

[6] C. BROYDEN, *A class of methods for solving nonlinear simultaneous equations*, Math. Comp., 19 (1965), pp. 577–593.

[7] ———, *The convergence of single-rank quasi-Newton methods*, Math. Comp., 24 (1970), pp. 365–382.

[8] C. BROYDEN, J. E. DENNIS, JR. AND J. MORE, *On the local and superlinear convergence of quasi-Newton methods*, J. Inst. Math. Appl., 12 (1973), pp. 223–246.

[9] S. N. CHOW, J. MALLET-PARET AND J. A. YORKE, *Finding zeros of maps: homotopy methods that are constructive with probability one*, Math. Comp., 32 (1978), pp. 887–899.

[10] M. G. CRANDALL AND P. H. RABINOWITZ, *Bifurcation from simple eigenvalues*, J. Funct. Anal., 8 (1971), pp. 321–340.

[11] D. DAVIDENKO, *On a new method of numerically integrating a system of nonlinear equations*, Dokl. Akad. Nauk. SSSR, 88 (1953), pp. 601–604 (In Russian).

[12] J. E. DENNIS, JR., *A brief survey of convergence results for quasi-Newton methods*, in Nonlinear Programming, SIAM-AMS Proceedings of Symposia in Applied Mathematics, American Mathematical Society, Providence, RI, pp. 185–199.

[13] J. E. DENNIS, JR. AND J. MORE, *A characterization of superlinear convergence and its applications to quasi-Newton methods*, Math. Comp., 28 (1974), pp. 549–560.

[14] J. E. DENNIS, JR. AND R. B. SCHNABEL, *Least change secant updates for quasi-Newton methods*, SIAM Rev., 21 (1979), pp. 443–459.

[15] P. DEUFLHARD, *A stepsize control for continuation methods and its special application to multiple shooting techniques*. Numer. Math., 33 (1979), pp. 115–146.

[16] D. M. GAY AND R. B. SCHNABEL, *Solving systems of nonlinear equations by Broyden's method with projected updates*, in Proc. of Nonlinear Programming III, Madison, Wisconsin, to appear.

[17] K. GEORG, *A simplicial deformation algorithm with applications to optimization, variational inequalities and boundary value problems*, in: Proc. of the Symposium on Fixed Point Algorithms and Complementarity, Academic Press, New York, to appear.

[18] C. HASELGROVE, *Solution of nonlinear equations and of differential equations with two-point boundary conditions*, Comput. J., 4 (1961), pp. 255–259.

[19] H. JÜRGENS AND D. SAUPE, *Methoden der simplizialen Topolgie zur numerischen Behandlung von nichtlinearen Eigenwertund Verzweigungsproblemen*, Diplomarbeit, Bremen, 1979.

[20] H. JÜRGENS, H.-O. PEITGEN AND D. SAUPE, *Topological perturbations in the numerical study of nonlinear eigenvalue and bifurcation problems*, in Analysis and Computation of Fixed Points, S. M. Robinson, ed., Academic Press, New York, to appear.

[21] J. L. KAPLAN AND J. A. YORKE, *Ordinary differential equations which yield periodic solutions of differential delay equations*, J. Math. Anal. Appl., 48 (1974), pp. 317–324.

[22] J. P. KEENER AND H. B. KELLER, *Perturbed bifurcation theory*, Arch. Rat. Mech. Anal., 50 (1973), pp. 159–175.

[23] H. B. KELLER, *Nonlinear bifurcation*, J. Differential Equations, 7 (1970), pp. 417–434.
[24] ———, *Numerical solution of bifurcation and nonlinear eigenvalue problems*, in Applications of Bifurcation Theory, P. H. Rabinowitz, ed., Academic Press, New York, 1977, pp. 359–384.
[25] ———, *Constructive methods for bifurcation and nonlinear eigenvalue problems*, in Computing Methods in Applied Sciences and Engineering, Lecture Notes in Mathematics 704, Springer, New York, 1979, pp. 241–251.
[26] H. B. KELLER AND W. F. LANGFORD, *Iterations, perturbations, and multiplicities for nonlinear bifurcation problems*, Arch. Rat. Mech. Anal. 48 (1972), pp. 83–108.
[27] R. B. KELLOGG, T. Y. LI AND J. A. YORKE, *A constructive proof of the Brouwer fixed point theorem and computational results*, SIAM J. Numer. Anal., 4 (1976), pp. 473–483.
[28] R. MENZEL AND H. SCHWETLICK, *Zur Lösung parameterabhängiger nichtlinearer Gleichungen mit singulären Jacobi-Matrizen*, Numer. Math., 30 (1978), pp. 65–79.
[29] J. W. MILNOR, *Topology from the Differentiable Viewpoint*, University Press of Virginia, 1969.
[30] J. J. MORE AND M. Y. COSNARD, *Numerical comparison of three nonlinear equation solvers*, TM-286, Argonne National Lab. Argonne, IL, 1976.
[31] M. Z. NASHED AND L. B. RALL, *Annotated bibliography on generalized inverses and applications*, in Generalized Inverses and Applications, M. Z. Nashed, ed., Academic Press, New York, 1976, pp. 771–1041.
[32] R. D. NUSSBAUM, *A global bifurcation theorem with applications to functional differential equations*, J. Funct. Anal., 19 (1975), pp. 319–338.
[33] H.-O. PEITGEN AND M. PRÜFER, *The Leray-Schauder continuation method is a constructive element in the numerical study of nonlinear eigenvalue and bifurcation problems*, in Functional Differential Equations and Approximation of Fixed Points, H.-O. Peitgen and H. O. Walther, eds., Lecture Notes in Mathematics 730, Springer, New York, 1979, pp. 326–409.
[34] M. PRÜFER, *Calculating global bifurcation*, in Continuation Methods, H. J. Wacker, ed., Academic Press, New York, 1978, pp. 187–213.
[35] ———, *Simpliziale Topologie und globale Verzweigung*, Dissertation, Bonn, 1978.
[36] P. H. RABINOWITZ, *Some global results for nonlinear eigenvalue problems*, J. Funct. Anal., 7 (1971), pp. 487–513.
[37] 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, 1974.
[38] ———, *Numerical methods for a class of finite dimensional bifurcation problems*, SIAM J. Numer. Anal., 15 (1978), pp. 1–11.
[39] ———, *Solution field of nonlinear equations and continuation methods*, preprint, 1979.
[40] C. P. SCHMIDT, *Approximating differential equations that describe homotopy paths*, preprint, 1979.
[41] L. F. SHAMPINE AND M. K. GORDON, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.
[42] J. SHERMAN AND W. J. MORRISON, *Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix*, Ann. Math. Statist., 20 (1949), pp. 621ff.
[43] H. J. WACKER, *Minimierung des Rechenaufwandes fur spezielle Iterationsverfahren vom Typ Minimales Residuum*, Computing, 18 (1977), pp. 209–224.
[44] ———, *A summary of the developments on imbedding methods*, in Continuation Methods, H. J. Wacker, ed., Academic Press, New York, 1978, pp. 1–36.
[45] H. J. WACKER, E. ZARZER AND W. ZULEHNER, *Optimal step-size control for the globalized Newton method*, in Continuation Methods, H. J. Wacker, ed., Academic Press, New york, 1978.
[46] H. O. WALTHER, *A theorem on the amplitudes of periodic solutions of delay equations with applications to bifurcation*, J. Differential Equations, 29 (1978), pp. 396–404.
[47] L. T. WATSON, *Solving the nonlinear complementary problem by a homotopy method*, SIAM J. Control Optim., 17 (1979), pp. 36–46.
[48] ———, *An algorithm that is globally convergent with probability one for a class of nonlinear two-point boundary value problems*, SIAM J. Numer. Anal., 16 (1979), pp. 394–401.
[49] ———, *A globally convergent algorithm for computing fixed points of C^2-maps*, Appl. Math. Comput., 5 (1979), pp. 297–311.
[50] L. T. WATSON, T. Y. LI AND C. Y. WANG, *The elliptic porous slider—a homotopy method*, J. Appl. Mech., 45 (1978), pp. 435–436.
[51] L. T. WATSON AND D. FENNER, *Chow-Yorke algorithm for fixed points or zeros of C^2-maps*, ACM Trans. Math. Software, to appear.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0005 $01.00/0

ON THE USE OF SPARSE MATRIX APPROXIMATION TO THE JACOBIAN IN INTEGRATING LARGE SETS OF ORDINARY DIFFERENTIAL EQUATIONS*

M. B. CARVER† AND S. R. MACEWEN†

Abstract. It is well known that large sets of coupled ordinary differential equations are difficult to integrate efficiently. The "stiffness" of the equations, arising from the wide range of embedded time constants, necessitates that implicit integration techniques be used. These normally require some form of algebraic equation solution involving the Jacobian matrix, a necessary operation which becomes increasingly expensive with the size of the equation set.

Efforts to partition large equation sets into linear and nonlinear or stiff and nonstiff subsets have proved fruitful, but are difficult to implement in a general manner, as such a division cannot always be found.

In the integration algorithm due to Gear, the implicit requirement is handled in an approximate manner by using a predictor-corrector iteration, again involving the Jacobian. The expediency of this operation is greatly enhanced by using sparse matrix techniques, but the operation remains relatively expensive for large sets. This paper illustrates that the approximation to the Jacobian matrix may be automatically adjusted to reduce the degree of coupling, and partition the predictor-corrector iteration into implicit and explicit segments. This further reduces the magnitude of the matrix operation without detracting from the accuracy of the integration.

Key Words. sparse matrix, Jacobian, differential equations, integration

1. Introduction. Numerical algorithms designed to efficiently integrate stiff ordinary differential equation sets use implicit techniques which normally require some form of linear equation solution involving the Jacobian matrix. The fact that this operation can be expensive has long been recognized. Gear [1] handled the implicit requirement by using recent approximations to the Jacobian in the predictor-corrector, and Hindmarsh further developed this theme by providing a series of codes [2] containing alternative means of expressing the Jacobian. These were particularly effective for the banded Jacobians which frequently arise in the solution of one-dimensional partial differential equations by the method of lines. Large equation systems arising from practical applications frequently do not yield banded Jacobians, but invariably the Jacobian is sparse. Sparse matrix techniques have been used with routines based on the Gear algorithm by Curtis [3], Carver [4] and Hindmarsh [5], [6]. Apart from the obvious storage and speed advantages inherent in sparse matrix techniques, quite a startling increase in efficiency is obtained in automatically optimizing the numerical evaluation of the Jacobian. In solving large problems in mass action chemical kinetics, Carver [7] has shown sparse matrix techniques more efficient for any set larger than 25 ordinary differential equations. Jacobian matrices associated with the ordinary differential equations arising from these kinetics problems are sparse and exhibit no fixed structure. By contrast, those arising from partial differential equations, particularly single equations in one dimension, are banded in structure and are best handled using a banded matrix algorithm. However, coupled partial differential equations, or sets of partial differential equations coupled with ordinary differential equations, such as those arising in neutron kinetics, generate a more complex Jacobian, which is striped rather than banded and may include randomly distributed elements which must be included implicitly in the predictor-corrector iteration [4]. This can be done efficiently only by using sparse matrix techniques for the Jacobian.

* Received by the editors April 4, 1980.

† Atomic Energy of Canada Limited, Chalk River Nuclear Laboratories, Chalk River, Ontario KOJ 1JO, Canada.

On integrating coupled two-dimensional partial differential equations, one finds the matrix problem even more severe, as the number of nonzero entries per variable is considerably greater than in one-dimensional cases and the structure is not simply banded. This, together with the fact that two-dimensional cases invariably require a large number of grid points, makes the matrix problem so severe that standard fully implicit techniques are impossible, and become at best impractical even when sparse matrices are introduced.

In previous work, success in such large problems has been gleaned by using partially implicit techniques which partition the equations in such a manner that all are not included in implicit form, thus reducing the magnitude of the matrix operation this requires. A number of methods of so dividing the equation set have been investigated.

Partitioning the equations into slow response and fast response sets has been considered by Blum [8], who proposes integrating the slow response sets by a simple explicit method while integrating the fast response or stiff sets by implicit methods. Porsching et al. [9] have applied a similar rationale to analytically partition the conservation equations. Palusinski and Wait [10] note further that if the stiff set also happens to be linear, as is often the case, impressive increases in efficiency are possible. Hofer [11] presents a formal evaluation of partitioning, and also shows it to be very effective for cases in which the fast decaying components originate from only a few easily identified equations. In all such cases, communication between slow and fast response sets must be established correctly by means of suitable interpolants; the method proposed below eliminates the need for interpolants.

All of the above techniques deal strictly with ordinary differential equations, and assume that the user will be able to identify the relevant partition a priori and write the equations separately. If the ordinary differential equations are generated from coupled partial differential equations, no suitable partition is evident, and in fact normally none can be made. It is, however, fruitful to envisage a partition, not with reference to particular equations, but with reference to particular terms in the equations and the relative influence of such terms on the Jacobian matrix. Thus by removing from the implicit formulation any terms which do not significantly influence convergence, one reduces the degree of implicit coupling and eventually partitions the predictor-corrector into implicit and explicit segments. In a recent paper, Enright and Kamel [12] also investigate this approach. They propose a method which automatically introduces such a partition during a similarity transform and also permits a fast update of the iteration matrix. Their tests of the method show the partitioning to be most effective at stringent tolerances. The method is restricted to full matrices.

In the following sections, it is shown that the iteration matrix can be automatically partitioned by using only slight modifications to standard sparse matrix techniques. The criteria used to accomplish this are discussed, and selected examples are given.

2. The philosophy of sparse matrix Jacobian approximations.

Background. The algorithm first published by Gear [1], and later expanded by Hindmarsh [2], [5], [6] and others [3], [4], has become one of the most popular integration techniques used in numerical analysis of ODE's and PDE's. Its success is due primarily to the efficiency it derives from minimizing the amount of matrix manipulation necessary to effect an implicit solution, and also to efficient selection of optimal integration step size.

The formulae used to integrate the ODE initial value problem,

$$\dot{y} = f(y), \qquad y(0) = Y_0, \tag{1}$$

are linear methods of the form

$$Y_n = \sum_{j=1}^{k_1} \alpha_j y_{n-j} + h \sum_{j=0}^{k_2} \beta_j \dot{y}_{n-j}, \tag{2}$$

where the vector y_i approximates $y(t_i)$, $\dot{y}_i = f(y_i, t_i)$ approximates $\dot{y}(t_i)$, $h = t_{i+1} - t_i$, and α and β are coefficients. For Adams methods of order q, $k_1 = 1$ and $k_2 = q - 1$; for backward differentiation formulae $k_1 = q$, $k_2 = 0$; but in either case $\beta_0 \neq 0$, so the formula is implicit.

Equation (2) may be written

$$g(y_n) = y_n - h\beta_0 f(y_n) - \sum_{j=1}^{k_1} \alpha_j y_{n-j} - \sum_{j=1}^{k_2} \beta_j \dot{Y}_{n-j} = 0. \tag{3}$$

This nonlinear equation may be solved by Newton iteration. For iteration $m+1$,

$$y_{n(m+1)} = y_{n(m)} - P_{n(m)}^{-1} g(y_{n(m)}), \tag{4}$$

where

$$P_{n(m)} = \left[\frac{\partial g}{\partial y}\right]_{y_{n(m)}} = \left[I - h\beta_0 \frac{\partial f}{\partial y}\right]_{y_{n(m)}} = I - h\beta_0 J_n, \qquad J_n = \left[\frac{\partial f}{\partial y}\right]_{y_n}. \tag{5}$$

As this is an iteration rather than an exact solution, replacing $P_{n(m)}$ by $P_{n(0)}$ avoids the necessity of repeating matrix calculations during the iteration at a very slight loss in the rate of convergence. In practice, further savings, again at a slight loss in convergence rate, are obtained by using $P_{n'}$ instead of P_n, where $P_{n'}$ is the matrix computed at some previous $n' < n$. Thus P_n would have to be computed only when $P_{n'}$ fails to produce rapid convergence. The loss of convergence due to the approximation is of course problem dependent. Although in theory the quadratically convergent iteration is reduced by such approximation to a linear convergent iteration, in practice the associated penalty is frequently acceptably small.

Although these approximations reduce the number of matrix manipulations, the evaluation of the matrix P, and subsequent LU decomposition required for repeated application, must still be completed a fairly large number of times during a typical integration. The efficiency of the algorithm therefore depends heavily on the expediency with which these operations are carried out. Recognizing this, Hindmarsh [2] provided a number of alternative versions of the GEAR code. Collectively, they contain several options for treating the matrix P as a full matrix, a banded matrix or a diagonally dominant matrix, and also provide a functional iteration option in which the Jacobian is not used at all and P reduces to I. For the particular matrices addressed, these options are of course exact. However, for the general case, in which systems are not truly banded, the above options may still be used successfully, but each may be regarded as a successively weaker approximation, and thus requires successively smaller step sizes to attain convergence. On the other hand, if the equation set is large, storing and handling the P_n matrix as a sparse matrix considerably reduces matrix manipulation cost while retaining the same convergence rate as the full matrix solution [4].

Integration using sparse techniques. The principles of sparse approximation to be discussed apply in general to any suitable combination of integration and sparse matrix modules. The GEARZ integrator was used for this particular study. This package uses the sparse matrix routines of Curtis and Reid [13] within a framework based on a combination of the Hindmarsh packages GEAR and GEARB [2]. GEARZ

is documented elsewhere [4], and its performance has been compared extensively to other state of the art packages [14].

The sparse matrix handling of the linear equation problem of (5) consists of six distinct stages. In the first, the structure of the Jacobian is assessed to determine the position of nonzero entries. In the second, the sequence of perturbing the y_j is optimized as discussed below to minimize the number of function evaluation calls required to obtain all the numeric values of the elements J_{ij}. The evaluation calls are made, and then the analyze stage determines a suitable pivot strategy for the Gaussian elimination. The operate stage performs the LU decomposition, and finally the solve stage does the back substitution to complete the equation solution. In the full matrix mode only three stages are required: evaluation, decomposition and solution. The advantage of using $P_{n'}$ instead of P_n lies in the fact that only the solution stage is required while $P_{n'}$ is in use. Sparse matrix analysis retains this advantage, and for a given structure the structure assessment, sequence optimization and analyze stages are also required only once. This introduces an additional degree of freedom in the protocol; that is at what point the structure of the matrix should be reassessed. As it is expensive, this operation is used as a last line of defence in the decision hierarchy. In the event that certain Jacobian elements are initially zero but become significant during the evolution of the integration, structure must be assessed once initially, again after the first few steps and then periodically during the integration. As equation systems may in fact have fixed Jacobian structure, the frequency of evaluation is placed under user control. Thus in sparse matrix mode $P_{n'}$ in (5) is replaced by $P_{n''}$, where n'' indicates that $P_{n''}$ is evaluated at $t_{n'} \leqq t_n$ using the sparsity structure found at $t_{n''} \leqq t_{n'}$.

A further advantage of using sparse or banded matrix techniques is that the numerical evaluation of the Jacobian may be optimized. The normal method of evaluating a Jacobian numerically is to perturb one of the dependent variables y_j and to evaluate $J_{ij} = \Delta f_i / \Delta y_j$ for all i. The procedure is carried out for each of the m variables j, and requires m calls to the function evaluation routine. If the system is tridiagonal, every third y_j may be perturbed simultaneously, as each y_j affects only $f_{j\pm1}$. Thus the Jacobian for a tridiagonal system may be evaluated with only three function evaluation calls. Similar savings may be realized in any system in which the structure is known, and in particular by examining the structure of the sparse matrix [13]. In the neutron kinetics example discussed below, the Jacobian of a 242-equation system is evaluated automatically by only seven function evaluation calls. The economics introduced by this approach to Jacobian evaluation is particularly important in view of the fact that it is rarely possible to express the Jacobian of a large system in an analytical manner.

Sparse approximation. The above section outlines the advantages of using sparse matrix techniques to evaluate an arbitrarily structured Jacobian and perform the linear equation solution (4) with the most accuracy possible, thereby allowing the integration to proceed with maximum step size. We now address the problem of large equation sets. In particular, the ordinary differential equations arising from the method of lines solution of coupled two-dimensional partial differential equations give rise to matrices which are sparse but have a fairly large number of nonzero elements. The matrix problem again becomes severe even when the sparse matrix approach is used. Typically a significant amount of time is required to perform each matrix operation, and the storage required to hold the decomposed matrix is high, possibly even exceeding available core.

This fact means that the criterion for integrating efficiently may become inverted. That is, although normally to integrate a small system of equations it is advisable to use a

new matrix evaluation to accelerate step size, for a large system it may be more suitable to reduce the frequency and the accuracy of the matrix operation at the expense of a reasonable decrease in step size.

This situation gives rise to a number of possibilities.

(1) *Automatic partitioning.* In order to reduce the size of the matrix problem, the approximation to P_n in (5) is carried one further step, again at the possible loss of convergence rate. $P_{n''}$ is replaced by $\tilde{P}_{n''}$, where $\tilde{P}_{n''}$ is an approximation obtained by selecting only terms which exert strong influence on the Jacobian coupling. This is readily accomplished by including those terms $J_{ij} = \Delta f_i / \Delta y_j$ which satisfy

$$|J_{ij}| > \frac{\varepsilon^*(|f_i| + |f_i + \Delta f_i|)}{2}. \tag{6}$$

In normal sparse matrix analyses of the Jacobian, the parameter ε is set to a number which merely distinguishes a nonzero element from effective numeric zero, but if larger values of ε are used, terms representing weak coupling will also be omitted from the structure. This has the effect of rendering the predictor-corrector iteration partially explicit instead of fully implicit, as the weakly coupled terms will be handled by functional iteration instead of Newton iteration. Although the level of partition is selected heuristically by increasing ε, the resulting partition is not arbitrary but closely reflects the relative strength of particular coupling terms and is readily related to the physics of the problem.

(2) *Sparsity control.* During the decomposition of a sparse matrix, a certain amount of "fill-in" or sparsity loss is inevitable, and the decomposed matrix requires more space than the original Jacobian. To a certain extent, the sparsity loss can be controlled by pivot strategy, and by introducing a "drop tolerance" which omits all fill-in elements below the specified magnitude. Again, the effects of this omission are problem dependent. Often the penalty in accuracy is acceptable, and in the present application it is advisable to minimize sparsity loss, provided this only marginally degrades step size, as storage and manipulation speed are considerably affected.

(3) *Step size and order changes.* For small equation sets, the strategy is to choose order to maximize step size without considering whether this will require matrix evaluation. With a large system, it is advisable to permit the computation to continue with current status unless a considerably larger possible step is permitted. Furthermore, when a step size increase is made, it is not inevitably necessary to immediately reevaluate the Jacobian, unless convergence fails. It is much cheaper to let convergence failure dictate the necessity of reevaluating the Jacobian than to risk unnecessary evaluation. Fixed leading coefficient formulae, in which P remains unchanged through a change in the order [15], could be advantageous here.

(4) *Sparsity structure evaluation.* When an approximation $\tilde{P}_{n''}$ to the sparse matrix is used it becomes more likely that the structure of significant elements may change, so this possibility must be addressed. In the event that the density increases during computation, the possibility also arises that in a large system the storage required for manipulation of the new matrix will exceed that available. In such cases it is necessary to freeze the matrix structure and forbid further structure assessment. This may of course degrade convergence and step size, but permits the computation to continue. While heuristic criteria can be introduced to govern the logic of the above decisions, it is apparent that they will not be universally applicable, as the relative expense of function evaluations and matrix operations will differ from case to case. Thus the governing parameters should be made accessible for possible user control.

The effects of these parameters are illustrated in the examples which follow. The numerical results shown are of course specific to GEARZ, but similar trends could be expected in integrators of like generic origin. A number of improved sparse matrix routines have evolved since the introduction of MA18. We would expect the advantages offered by sparse approximation using these routines would be at least commensurate with those discussed. More recent routines, for example, allow better control of sparsity loss. In particular, the successor to MA18, MA28, offers faster, more accurate operation at the expense of slightly more indexing overhead [16]. The priorities of the current study, however, dictate the sacrifice of some speed to save storage.

Neutron kinetics benchmark. The first example of automatic sparse partitioning involves the equations describing the evolution in time and space of neutron populations at various energies in a reactor submitted to perturbation. If $\phi_g(x, t)$ is the neutron flux for energy group g, C_l is the delayed neutron precursor density for delayed group l, S_g is a neutron source, $T_f(x, t)$ and $T_c(x, t)$ are fuel and coolant temperatures, and the remaining functions describe space-dependent reactor properties, then these equations may be written for a total of G and L groups as

$$\frac{\partial \phi_g}{\partial t} = a_g \nabla \cdot (d_g \nabla \phi_g) - b_g \phi_g + \sum_{g'=1}^{G} C_{gg'} \phi_{g'} + \sum_{l=1}^{L} e_{lg} C_l + S_g, \qquad g = 1, G, \tag{7}$$

$$\frac{dC_l}{dt} = \sum_{g=1}^{G} P_{gl} \phi_g - q_l C_l, \qquad l = 1, L, \tag{8}$$

$$\frac{\partial T_f}{\partial t} = -u_f T_f + v_f T_c + \sum_{g=1}^{G} w_{fg} \phi_g, \tag{9}$$

$$\frac{\partial T_c}{\partial t} = -u_c T_c + v_c T_f + \sum_{g=1}^{G} w_{cg} \phi_g - S_c \frac{\partial T_c}{\partial x}. \tag{10}$$

If N spatial points are chosen, these generate $(2+G+L)N$ coupled linear ordinary differential equations. Here we illustrate the solution of ANS benchmark problem ID6-A2. In this benchmark, boundary values of the flux are zero, $\phi_g(x, 0) = \phi_{g0}(x)$ and the values of all necessary coefficients are given in full detail in reference [17], which also gives a number of independent solutions. The benchmark uses two energy groups, six delayed groups and 121 points, but omits the temperature equations, thus generating 968 ordinary differential equations. These may be solved by traditional partitioning; the 242 equations for the two fast flux groups are integrated by the GEARZ algorithm, and the 726 equations describing the slower delayed neutron groups are solved by synchronized Euler integration. The resulting Jacobian matrix (Fig. 1) has a form suggesting block iterative solution, but 242 equations are readily handled as a sparse matrix and all Jacobian elements may be evaluated using only seven calls to the function defining the equations. From the computing statistics shown for this problem in Table 1, it is obvious that the standard sparse matrix techniques are the only practical means of completing the required integration; they achieve the same step size as the full matrix method, but use orders of magnitude fewer function evaluations to complete the required 4 seconds problem time. In this case, note that the use of the banded and diagonal options can be regarded as successively weaker approximations to the true Jacobian, and perform accordingly.

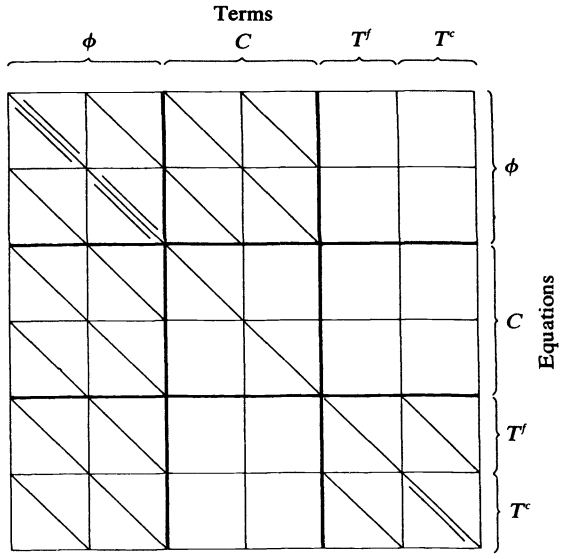


FIG. 1. *Structure of Jacobian matrix, neutron kinetics example.*

TABLE 1
Performance of GEARZ *algorithm with neutron kinetics equations.*

Problem time	Method	Functional iteration	Diagonal approx.	Banded (triagonal)	Full matrix	Sparse matrix
10^{-5}	NF	390	664	2239	2239	357
	NS	164	149	41	41	41
	CP	5	10	17	35	9
	H	10^{-7}	10^{-7}	10^{-6}	10^{-6}	10^{-6}
10^{-3}	NF	~3000	12884	6248	4478	472
	NS	~3000	1725	112	91	91
	CP	~400	152	43	73	12
	H	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-4}
10^{-1}	NF	—	—	42118	6295	587
	NS	—	—	369	153	153
	CP	—	—	271	143	16
	H	—	—	10^{-5}	.004	.004
1	NF	—	—	—	17302	994
	NS	—	—	—	211	211
	CP	—	—	—	375	33
	H	—	—	—	.05	.05
4	NF	—	—	—	—	1744
	NS	—	—	—	—	260
	CP	—	—	—	—	50
	H	—	—	—	—	.2

Required: to complete the given problem times with 0.01% relative imposed tolerance.
NF Number of function calls. CP Computing time on CDC CYBER 175.
NS Number of line steps. H Time step at given problem time.
— Incomplete after 300 CP seconds computing time.

The results of secondary partitioning are given in Table 2. Here, in addition to the above primary partition by equations, the PDE block is further partitioned into partially implicit form by use of the significance or sparse partitioning parameter ε in (6). Note that in this case the equation set has already been partitioned once, and only a

TABLE 2
Effect of sparse-partitioning parameters on integration performance, neutron kinetics example.

Problem time	Method	No. of entries in Jacobian	Decomposed matrix size	Jacobian evaluations	Function calls	Integration steps	CP time
10^{-5}	$\varepsilon \leqq 10^{-14}$	953	1188	9	357	41	8
	10^{-13}	844	1025	95	1354	72	36
	10^{-12}	710	943	131	2554	15	41
	10^{-10}	670	737	138	2230	38	33
	10^{-8}	242	242	142	2086	64	32
	Functional Iteration	—	—	142	390	164	5
	Diagonal	242	242	140	664	149	3
	Full	242^2	242^2	9	2239	41	26
10^{-3}	$\varepsilon < 10^{-14}$	953	1188	18	472	92	12
	10^{-13}	796	969	151	863	223	100
	10^{-12}	—	—	—	—	—	—
	Functional Iteration	—	—	—	—	—	—
	Diagonal	—	—	1040	12884	1725	65
	Full	—	—	18	4478	92	91
4	$\varepsilon \leqq 10^{-14}$	953	1188	91	1744	260	50
	Diagonal	—	—	—	—	—	—
	Full	—	—	—	—	—	—

— Indicates this option was not completed as it consumed excessive time.

few further terms may be rendered explicit before the efficiency of integration begins to deterioriate markedly. The increase of ε from 10^{-15} to 10^{-14} reduces the number of Jacobian entries from 953 to 844. The integration then requires about twice as many steps but uses approximately the same computing time as a full matrix solution. When the ε parameter is made large enough, the Jacobian matrix is neglected, the P matrix reduces to the identity matrix and the predictor-corrector is reduced entirely to functional iteration. There is of course a considerable amount of effort wasted in generating the identity matrix in this manner. The simple functional iteration option proceeds much faster and is more efficient for all values above some large value of ε. However, the results are the same in each case, confirming that the sparse approximation does reduce to functional iteration in the limit. This fact, that sparse approximation returns the correct results even in this severe limiting case, permits the method to be used with confidence for truly large systems of equations which cannot be partitioned a priori on physical grounds. In some cases it is not only impossible to provide enough storage for the full matrix inversion, but there is also insufficient storage for an accurate sparse matrix inversion. In such cases it may be possible to obtain a reasonably efficient solution by increasing the sparse partition parameter ε until the storage required for the

sparse matrix operations will fit into available core. This procedure can of course be automated. The second example involves such a case.

Point defect kinetics example. The irradiation of a crystalline solid by energetic particles, such as electrons or neutrons, results in the production of vacancy and interstitial point defects. The coupled partial differential equations which define the production, mutual recombination and loss to internal surfaces of each type of point defect are given by

$$\frac{\partial C_i}{\partial t} = P_i - RC_vC_i + D_i\nabla\cdot\left\{\nabla C_i + \frac{C_i}{kT}\nabla E_i\right\}, \tag{11}$$

$$\frac{\partial C_v}{\partial t} = P_v - RC_vC_i + D_v\nabla\cdot\left\{\nabla C_v + \frac{C_v}{kT}\nabla E_v\right\}. \tag{12}$$

In (11) and (12), P_i, C_i and D_i, P_v, C_v and D_v are the production rate, concentration and diffusion coefficient for interstitials and vacancies respectively; R is the recombination rate constant; k is Boltzman's constant and T is the absolute temperature. Typically, $P_v = P_i$, and D_v is about six orders of magnitude less than D_i. E_i and E_v define the interaction between interstitials and vacancies respectively and the sinks to which they migrate. The nature of the sinks is defined by the choice of region in which (11) and (12) are to be solved, and by the boundary conditions imposed on the surfaces of that region. Specific forms of E appropriate for polar coordinates have been given by Wolfer and Ashkin [18]. For the simplest case of misfit interaction only, E is of the form

$$E_v = \frac{B_v \sin\theta}{r}, \tag{13}$$

$$E_i = \frac{B_i \sin\theta}{r}, \tag{14}$$

with $B_i \gg B_v$.

The study currently underway involves solving (11) and (12) in polar coordinates for a variety of asymmetric boundary formulations. Here, however, we consider only the simple case of concentric circles with C_v and C_i specified on each. The values of the constants used for the example are given in Table 3.

The precise solution of the point defect kinetics equations (11) and (12) depends on the choice of grid resolution, and on the method used to approximate the spatial

TABLE 3
Input constants (point defect kinetics).

Parameter	Vacancy	Interstitial
P	1×10^{-7} at.fr./s	1×10^{-7} at.fr./s
D	4.69×10^{-11} cm^2/s	1.08×10^{-4} cm^2/s
R	2.64×10^{3} s^{-1}	2.64×10^{3} s^{-1}
B/kT	0.0	5.48×10^{-7} dyne-cm^2
r_I (inner boundary)	1.4×10^{-7} cm	1.4×10^{-7} cm
r_0 (outer boundary)	4.2×10^{-5} cm	4.2×10^{-5} cm
C_0	3.78×10^{-11}	8.79×10^{-27}

The boundary conditions are defined at $r = r_I$ and $r = r_0$, for each point defect, by the expression $C = C_0 \exp(-B \sin\theta/r)$.

derivatives. To obtain an accurate solution a refined treatment of differentiation along the radial direction is required. This solution is the subject of a further publication [19]. However, as the differentiation method does not detract from the efficiency of sparse matrix partitioning techniques, we here use a simple exploratory example to illustrate the technique, employing a simple equally divided 10×10 grid with standard fourth-order differentiation formulae in the radial direction and second-order in the azimuthal. With E_i and E_v defined by (13), (14) and Table 3, the equations may be written as

$$\frac{\partial C_i}{\partial t} = P_i - RC_vC_i + D_i \frac{\partial^2 C_i}{\partial r^2} + D_i \frac{\partial C_i}{\partial r}\left(\frac{1}{r} + \frac{1}{kT}\frac{\partial E_i}{\partial r}\right) + \frac{D_i}{kTr^2}\frac{\partial E_i}{\partial \theta}\frac{\partial C_i}{\partial \theta} + \frac{D_i}{r^2}\frac{\partial^2 C_i}{\partial \theta^2}, \tag{15}$$

$$\frac{\partial C_v}{\partial t} = P_v - RC_vC_i + D_v \frac{\partial^2 C_v}{\partial r^2} + \frac{D_v}{r^2}\frac{\partial^2 C_v}{\partial \theta^2}. \tag{16}$$

Since E_i defined by (14) is harmonic, boundary conditions are applied only in the r-direction, reducing the number of active differential equations to 160. The Jacobian has the structure shown in Fig. 2 with eight nonzero entries per row; thus the total number of nonzero entries is 1280. Successively increasing the ε parameter in (6)

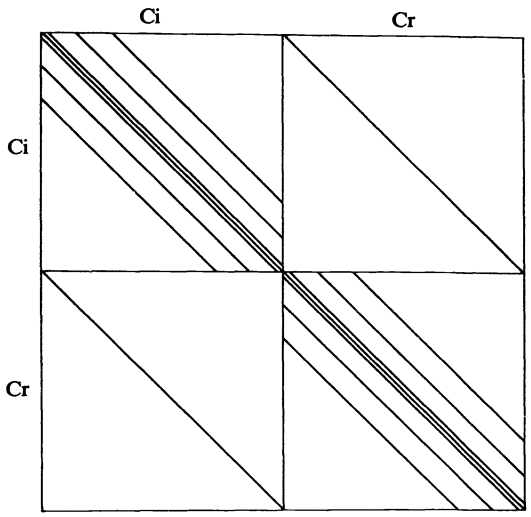


FIG. 2. *Structure of Jacobian matrix, point defect kinetics equations.*

reduces the number of entries by eliminating the weaker coupling elements from the Jacobian as shown in Table 4. The effect of such elimination is summarized in Table 5. The integration is required to complete in 10 seconds problem time. Note that a solution incorporating all the Jacobian elements consumes a considerable amount of time in matrix manipulation. This time is reduced considerably by eliminating weak

TABLE 4
Effect of sparse matrix partitioning parameter on Jacobian structure example, point defect kinetics

ε	No. of entries in Jacobian	No. of entries in decomposed matrix	C_v equation coupling included				C_i equation coupling included			
			r	θ	Diag.	C_i	r	θ	Diag.	C_v
$<10^{-6}$	1280	4920	4	2	1	1	4	2	1	1
10^{-5}	1120	4222	2	2	1	1	4	2	1	1
10^{-4}	880	2120	2	0	1	1	4	2	1	1
10^{-3} to 1	720	1572	0	0	1	0	4	2	1	1
10	540	1270	0	0	1	0	2	2	1	1
10^2	400	458	0	0	1	0	2	0	1	1
10^3	200	200	0	0	1	0	0	0	1	0

coupling terms, and the Jacobian involving C_v can be reduced entirely to functional iteration form without degrading convergence and attainable step size. Thus the equations have been partitioned, in that the C_v equation has been rendered explicit in the predictor-corrector. The C_i equation remains very stiff, and any attempt to eliminate further terms degrades convergence considerably. As even the cross coupling term RC_vC_i is eliminated from the Jacobian, it would appear that in the absence of this term, the C_i solved alone would be stiff and the C_v equation solved alone would not be stiff. This indeed turns out to be so, but particularly in the presence of the coupling term, this partition is not self-evident.

Note that the range $10^{-3} \leqq \varepsilon \leqq 1$ reduces the C_v equation to functional iteration, and any $\varepsilon < 1$ removes terms from the C_i Jacobian. A value of $\varepsilon = 10^3$ initially reduces the C_i equation to functional iteration. Note that this value requires the same number of function evaluations as the standard functional iteration option for problem time $t \leqq 10^{-5}$, but as the computation progresses more Jacobian terms become significant. $\varepsilon = 10^3$ admits these terms into the calculation and finishes the entire computation in reasonable time, whereas the functional iteration option does not. Finally, the table shows that reducing the number of entries in the Jacobian also reduces the size of the decomposed matrix by more than a proportionate amount. The latter matrix is the one which limits the calculation as it is always the larger, so the reduction of fill-in is a considerable saving.

A sufficiently accurate representation in space requires 480 grid points and special differentiating techniques [19]. Similar savings are made in this case, in fact sparse partitioning is essential, as without some elimination the matrix manipulation requirements exceed the 98K core available in the CDC 175 computer.

Some further advantages of sparse partitioning. Apart from the obvious advantage that sparse partitioning reduces matrix storage and manipulation time in such a way that hitherto unsolved problems become more tractable, several further advantages are apparent.

Firstly, it removes from the user the necessity of personally determining a partition on physical or mathematical grounds. This is extremely useful in general simulation applications, in which the user may not have specialized knowledge.

Secondly, during the solution phase the integration is implicit, and the step size and order of the integration formula is the same for all components. Naturally, this avoids the difficulties of matching order and formulae which are inherent in classic partitioning.

TABLE 5

Effect of sparse matrix partitioning parameter on integration performance, point defect kinetics

					$t = 10^{-5}$			$t = 10^{-3}$			$t = 10^{-1}$			$t = 10$			
ε	NJI	NJD	NCV	NCI	NF	NS	CP	NF	NS	CP	NF	NS	CP	NF	NS	CP	NJF
$<10^{-6}$	1280	4420	8	8	382	6	45	587	61	63	—	—	—	—	—	—	1280
10^{-5}	1120	4222	6	8	362	6	26	541	61	41	—	—	—	—	—	—	1280
10^{-4}	880	2120	3	8	356	6	13	528	61	26	672	91	37	986	141	56	880
10^{-3} to 1	720	1572	1	8	350	6	12	515	61	20	655	91	30	908	141	43	720
10	540	1270	1	6	344	6	11	502	61	20	650	91	28	1158	163	54	720
10^{2}	400	548	1	4	320	6	8	1211	127	30	1369	158	41	1895	230	72	664
10^{3}	200	200	1	1	727	94	16	2210	268	49	2641	295	58	3035	419	92	646
Functional iteration					231	94	6	—	—	—	—	—	—	—	—	—	—
Full matrix					1272	6	35	—	—	—	—	—	—	—	—	—	—
Diagonal approximation					218	63	5	2314	510	36	—	—	—	—	—	—	—

NJI Number of elements initially included in Jacobian.
NJF Number of elements finally included in Jacobian.
NJF Number of elements in decomposed Jacobian NJI.
NCV Number of Jacobian elements per C_v equation.
NCI Number of Jacobian elements per C_i equation.
CPT Computation time elapsed (CDC CYBER 175).
t Problem time (completed to 10^{-4} relative accuracy).
— Indicates this option was not completed as it consumed excessive time.

The sparse partitioning method is also fully dynamic, and automatically caters for changes in the required partitioning which may evolve during integration in cases of variable stiffness.

Further study of the possibilities may improve the method of invoking sparse partitioning. The current use of the ε drop tolerance in the sparse structure assessment is simple to use and automatically assigns a suitable partition, but unfortunately the relationships among the value of ε, the number of terms it will eliminate and the effect of eliminating these terms are not directly predictable. Experience has shown, however, that short trial runs using different values of ε will often quickly indicate a suitable partition for full computation, and an option which prints Jacobian structure and values has proved extremely useful in identifying the relationships between ε and the physical meaning of the associated partition.

Finally, it should be pointed out that the use of sparse matrix partitioning is not restricted to the case on which Jacobian elements are evaluated numerically. For cases such as the MACKSIM chemical kinetics simulation package [7], in which the Jacobian is computed from analytical expressions but is stored and handled as a sparse matrix, a similar drop tolerance decision may be made on the magnitude of any element before the element is included in the matrix structure pattern.

REFERENCES

[1] C. W. GEAR, *The automatic integration of ordinary differential equations*, Comm. ACM, 14 (1971), pp. 176–190.

[2] A. C. HINDMARSH, *The* LLL *Family of Ordinary Differential Equation Solvers*, UCRL-78129, April, 1976.

[3] A. R. CURTIS, *The* FACSIMILE *Numerical Integrator for Stiff Initial Value Problems*, AERE-*R*-9352, April, 1979.

[4] M. B. CARVER AND A. P. BAUDOUIN, *Solution of reactor kinetics problems using sparse matrix techniques in an* ODE *integrator for stiff equations*, in Advances in Computer Methods for Partial Differential Equations, R. Vichnevetsky, ed., AICA Press, 1975; Atomic Energy of Canada Limited report AECL-5177, 1975.

[5] J. W. SPELLMAN AND A. C. HINDMARSH, GEARS—*Solution of Ordinary Differential Equations Having a Sparse Jacobian Matrix*, UCID-30116, August, 1975.

[6] A. C. HINDMARSH AND A. H. SHERMAN, GEARS—*A Package for the Solution of Sparse Stiff Ordinary Differential Equations*, UCRL-84102, March, 1980.

[7] M. B. CARVER AND A. W. BOYD, *A program package using stiff sparse techniques for the automatic solution of mass action chemical kinetics*, Internat. J. Chem. Kinet., 11 (1979), pp. 1097–1108.

[8] H. BLUM, *Numerical integration of large scale systems using separate step sizes*, in Numerical Methods for Differential Equations and Simulation, A. W. Bennett and R. Vichnevetsky, eds., IMACS, North-Holland, Amsterdam, 1978.

[9] T. A. PORSCHING, J. H. MURPHY AND J. A. REDFIELD, *Stable numerical integration of conservation equations for hydraulic networks*, Nucl. Sci. & Eng., 43 (1971), pp. 218–225.

[10] O. PALUSINSKI AND J. WAIT, *Simulation methods for combined linear and nonlinear systems*, Simulation, 30 (1978), pp. 85–97.

[11] E. HOFER, *A partially implicit method for large stiff systems of* ODEs *with only few equations introducing small time constants*, SIAM J. Numer. Anal., 13 (1976), pp. 645–663.

[12] W. H. ENRIGHT AND M. S. KAMEL, *Automatic partitioning of stiff systems and exploiting the resulting structure* ACM TOMS, 4 (1979), pp. 373–385.

[13] A. R. CURTIS AND J. K. REID, FORTRAN *Routines for the Solution of Sparse Sets of Linear Equations*, AERE-R-6844, 1971.

[14] M. B. CARVER, *In search of a robust integration algorithm for simulation use: some tests, results and recommendations*, Simulation Council Proceedings on Numerical Integration, R. E. Crosbie, ed., in press.

[15] K. R. JACKSON AND R. SACKS-DAVIES, *An alternative implementation of variable step formulas for stiff* ODEs, Tech. Rep. 121, Dept. of Computer Science, University of Toronto, 1978.

[16] I. S. DUFF AND J. K. REID, *Some design features of a sparse matrix code*, ACM TOMS, 4 (1979), pp. 18–25.
[17] *Argonne Code Center Benchmark Problem Book*, ANL-7416-1, Argonne National Laboratory, 1972.
[18] W. G. WOLFER AND N. ASHKIN, *Diffusion of vacancies and interstitials to edge dislocations*, J. Appl. Phys., 47 (1976), pp. 791–800.
[19] S. R. MACEWEN, M. B. CARVER AND J. M. BLAIR, to be published.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0006 $01.00/0

ON THE POINT VORTEX METHOD*

D. W. MOORE†

Abstract. The discretization of the integrodifferential equation governing the evolution of a vortex sheet leads to a representation of the sheet by point vortices. It is shown, by examination of the special case of a uniform circular vortex sheet, that the chaotic motion which often arises when the point vortex representation is used is due to the amplification of numerically introduced disturbances. The mechanism is a discrete form of Helmholtz instability. The linear smoothing method of Longuet-Higgins and Cokelet (1976) and the repositioning method of Fink and Soh (1978) are shown to reduce the instability.

Key Words. point vortex, Helmholtz instability, vortex sheet

1. Introduction. The problem considered in this paper is that of integrating the equation governing the motion of a vortex sheet in two dimensions.

A convenient formulation, and one which is implicit in numerical work going back to Rosenhead (1931), is in terms of Birkhoff's and Rott's circulation coordinate. Suppose P is a typical fluid particle of the sheet and Q is a reference fluid particle. If the vorticity in the sheet is of one sign,[1] the net vorticity or circulation Γ in the arc QP is a monotone function of the arc distance between Q and P, and Γ is a Lagrangian intrinsic coordinate. It is Lagrangian because the circulation between the fluid particles Q and P is invariant as the vortex sheet evolves.

Then, if $z(\Gamma, t)$ is the complex coordinate of P with respect to axes in which the flow at infinity is zero, the evolution of the sheet from an initial shape $z(\Gamma, 0)$ $(0 \leqq \Gamma \leqq \Gamma_e)$ is governed by the singular integrodifferential equation

$$\frac{\partial z^*}{\partial t}(\Gamma, t) = \frac{-i}{2\pi} \int\!\!\!\!\!\!- _0^{\Gamma_e} \frac{d\Gamma'}{z(\Gamma, t) - z(\Gamma', t)}, \qquad 0 < \Gamma < \Gamma_e; \tag{1.1}$$

the slash in the integral sign denotes the Cauchy principal value.

Difficulties of four different types arise when this problem is tackled numerically.

1. The evaluation of the principal value integral cannot accurately be accomplished merely by applying a standard integration formula.

2. The integral can diverge at the end points of the vortex sheet.

3. Portions of the sheet with $\Gamma' \neq \Gamma$ can have small values of $|z(\Gamma, t) - z(\Gamma', t)|$. This happens in a tightly wound spiral (Maskew (1977)) or in the modeling of separation from smooth surfaces (Fiddes (1980)) where the sheet is close to its image in the surface. This makes the accurate evaluation of the integral difficult.

4. A straight uniform vortex sheet of strength γ is unstable to small sinusoidal disturbances of any wavelength λ, the growth rate being $\pi\gamma/\lambda$. This phenomenon of Helmholtz instability persists in curved nonuniform vortex sheets, at least for short waves, unless the sheet is rapidly stretching (Moore and Griffith-Jones (1974), Moore (1976)). In the author's view, this will imply that roundoff and truncation errors are rapidly amplified to cause the chaotic motion which often ruins practical calculation. Moreover, since the shorter the wave the faster it grows, nonlinear excitation of short waves can lead to a singularity developing in $z(\Gamma, t)$ at a finite time even if $z(\Gamma, 0)$ is analytic (Saffman and Baker (1978), Moore (1979)). Thus the initial value problem for (1.1) is ill-posed.

* Received by the editors August 20, 1980, and in revised form October 31, 1980.

† Department of Mathematics, Imperial College, Exhibition Road, London SW7, England.

[1] If not, the sheet can be split up into a finite number of arcs or (Baker (1980)) other Lagrangian coordinates introduced.

The second and third difficulties will not be treated here and—for simplicity—the vortex sheet will be a simple closed curve, so that

$$z(\Gamma_e, t) = z(0, t); \tag{1.2}$$

the shape of the sheet is analytic everywhere.

The first difficulty was overcome by Van der Vooren[2] (1965) who subtracted from the integrand in (1.1) a function with the same singularity, which thus rendered the integrand regular. The canceling function could be integrated analytically. This is recommended practice in numerical analysis (Isaacson and Keller (1966, p. 346)) but its application to the vortex sheet was novel.

If trapezoidal integration is used, Van der Vooren's procedure leads to

$$\frac{-i}{2\pi}\int_0^{\Gamma_e} \frac{d\Gamma'}{z_s - z(\Gamma', t)} = \frac{-i}{2\pi} \sum_{\substack{p=0 \\ p\neq s}}^{p=N} \frac{\Delta\Gamma}{z_s - z_p} - \frac{i\Delta\Gamma}{4\pi} \frac{\left(\dfrac{\partial^2 z}{\partial\Gamma^2}\right)_s}{\left(\dfrac{\partial z}{\partial\Gamma}\right)_s^2}, \tag{1.3}$$

where

$$z_p = z(p\Delta\Gamma, t), \qquad (p = 0, 1, 2, \cdots, N), \tag{1.4}$$

and where the range of integration $(0, \Gamma_e)$ has been divided into N equal portions of length $\Delta\Gamma$. Van der Vooren's interest was in the growth of periodic waves on an infinite uniform vortex sheet, but his result holds also for a closed vortex sheet; a proof is given in the Appendix.

If Van der Vooren's formula is substituted into the governing integrodifferential equation it becomes

$$\frac{\partial z_s^*}{\partial t} = -\frac{i}{2\pi} \sum_{\substack{p=0 \\ p\neq s}}^{N} \frac{\Delta\Gamma}{z_s - z_p} - \frac{i\Delta\Gamma}{4\pi} \frac{\left(\dfrac{\partial^2 z}{\partial\Gamma^2}\right)_s}{\left(\dfrac{\partial z}{\partial\Gamma}\right)_s^2}, \tag{1.5}$$

and if the differential coefficients in the last term are replaced by finite differences, so that, for example,

$$\left(\frac{\partial z}{\partial\Gamma}\right)_s = \frac{z_{s+1} - z_{s-1}}{2\Delta\Gamma} \tag{1.6}$$

and

$$\left(\frac{\partial^2 z}{\partial\Gamma^2}\right)_s = \frac{z_{s+1} - 2z_s + z_{s-1}}{\Delta\Gamma^2}, \tag{1.7}$$

a system of ordinary differential equations results. The first term on the right is the familiar point vortex approximation introduced by Rosenhead (1931), and thus the error in the point vortex approximation is less than e_N, where

$$e_N = \frac{\Gamma_e}{4\pi N} \max \frac{\left|\dfrac{\partial^2 z}{\partial\Gamma^2}\right|_s}{\left|\dfrac{\partial z}{\partial\Gamma}\right|_s^2}. \tag{1.8}$$

[2] This work, which appeared in 1965 in report form, will shortly be published.

Thus the point vortex approximation is a consistent approximation because the truncation error e_N goes to zero as the number of integration points goes to infinity, although it is a crude approximation for any practical value of N. The criticism of the method made recently by Fink and Soh (1978) is thus unjustified and Baker (1980) has, by a different method, established the asymptotic validity of the point vortex formula. In fact, as pointed out by Van der Vooren and by Baker, the error in (1.3) is exponentially small, because the right-hand side of (1.3) is the result of applying trapezoidal integration to a smooth periodic function (Isaacson and Keller (1966, p. 340)).

The fourth difficulty remains to be resolved. In discussing it, it must be kept in mind that the vortex sheet is itself an approximation to a thin shear layer. Now a straight shear layer with uniform vorticity and constant thickness is not violently unstable to very short waves, as was shown by Rayleigh (1945, p. 392). Waves with wavelength less than a critical wavelength equal to 4.8 shear-layer thicknesses are not amplified.

The contention that Helmholtz instability is the fundamental difficulty in vortex sheet calculations has been contested by Fink and Soh (1978), who claim that integration error causes the chaotic motion. In support of this, Fink and Soh introduced a repositioning technique aimed at improving the accuracy of the evaluation of the integral in (1.1) and found that chaotic motion disappeared. Moreover, a deliberately introduced disturbance did not amplify or cause chaotic motion.

In § 2 an attempt is made to resolve this dispute. A uniform circular vortex sheet is studied, because the exact solution of (1.1) and the exact solution of the point vortex representation equation (1.5) are both known. It is shown that errors grow rapidly, whether the corrected or uncorrected point vortex formula is used, and that the growth rate of the error is that of the most unstable mode of Helmholtz instability of the representing point vortices. The most unstable wave has a period of two spacings, whereas Fink and Soh introduced a disturbance of length 10 spacings the growth rate of which on linear theory is 0.36 of that of the most unstable wave. A more stringent test still of the hypothesis that this discrete form of Helmholtz instability is present is to Fourier analyze the error; this reveals the presence of less rapidly growing modes, each growing at its predicted growth rate.

If it is granted that it is the growth of short waves which can ruin calculations with vortex sheets, it is sensible to consider ways of removing the instability. This is because, as has been pointed out, the instability is introduced by the step of replacing a shear layer of small, but finite, thickness by a vortex sheet.

One could give up the vortex sheet approximation and return to the computation of the evolution of a thin layer. This is without doubt the most satisfactory procedure, but it involves much more computation.

An alternative approach is to modify the integrodifferential equation (1.1) to allow for finite thickness (Moore (1978), Dhanak (1980)) but the resulting equation, while only a little more complicated than (1.1), has not proved amenable to computation. The equation is valid only for disturbances that are long compared to the shear-layer thickness and it has a spurious short wave instability rather like that of the KdV equation, which is also a long wave approximation.

Another possibility is to apply a linear smoothing formula, such as that introduced by Longuet-Higgins and Cokelet (1976) in their work on nonlinear water waves. In § 3 their formula is tried on the point vortex representation of a straight uniform vortex sheet. It is shown analytically, by an extension of Von Karman's analysis (Lamb (1932, p. 225)), to remove completely the most unstable mode of instability, while the less unstable modes are suppressed or have reduced growth rates. A test on the uniform

circle shows that one complete revolution can be followed when 60 vortices are used but that chaotic motion set in at about one and one quarter revolutions.

There remains the question of the absence of chaotic motion in Fink and Soh's (1978) calculations and in the subsequent use of Fink and Soh's method by Sarpkaya and Shoaff (1979). In § 4 a simplified version of this repositioning method is examined analytically, again in the context of the straight uniform vortex sheet. It is shown to remove the most unstable mode and to reduce the growth of the higher modes of Helmholtz instability in this case and thus, while its effect on the accuracy of the point vortex method can be questioned (Baker (1980)), it is an effective stabilizing technique.

Repositioning was applied to the circle and was shown to increase the time of onset of visible error by a factor of two. The error appeared, not as chaotic motion, but as a spurious wave.

Linear smoothing and repositioning were compared for an initially circular nonuniform vortex sheet. The gross features of the evolution agree but features involving few vortices do not. Repositioning was markedly more successful in preventing chaotic motion, which—in these tests—it suppressed completely.

2. The circular vortex sheet. A simple exact solution of (1.1) is that for a uniform circular vortex sheet of unit strength and unit radius. The flow field for $|z|>1$ is a potential vortex of circulation 2π, whereas for $|z|<1$ the fluid is at rest. The circular vortex sheet $|z|=1$ represents a discontinuity across which the fluid velocity jumps from 0 to 1 so that the fluid particles comprising the sheet move with a velocity $\frac{1}{2}$, which is the average of the velocities on the two sides of the sheet. In circulation coordinate terms the solution is

$$z(\Gamma, t) = \exp\{i(\Gamma + \tfrac{1}{2}t)\}, \qquad 0 \leqq \Gamma \leqq 2\pi, \tag{2.1}$$

as can be verified by substituting in (1.1) and evaluating the integral by residues.

If the sheet is replaced by N equal point vortices each of circulation $2\pi/N$ placed along the sheet at angular separations $2\pi/N$, the resulting regular polygon of point vortices rotates with angular velocity $\Omega_N^{(0)}$, where

$$\Omega_N^{(0)} = \frac{1}{2}\left(1 - \frac{1}{N}\right). \tag{2.2}$$

If, however, Van der Vooren's correction is applied with fourth order differences for the derivatives, so that in (1.5)

$$\left(\frac{\partial z}{\partial \Gamma}\right)_s = \frac{z_{s-2} - 8z_{s-1} + 8z_{s+1} - z_{s+2}}{12\Delta\Gamma} \tag{2.3}$$

and

$$\left(\frac{\partial^2 z}{\partial \Gamma^2}\right)_s = \frac{-z_{s-2} + 16z_{s-1} - 30z_s + 16z_{s+1} - z_{s+2}}{12\Delta\Gamma^2}, \tag{2.4}$$

the regular polygon rotates with velocity $\Omega_N^{(4)}$, where

$$\Omega_N^{(4)} = \frac{1}{2}\left(1 - \frac{1}{N}\right) + \frac{3}{4N}\sec^2\frac{1}{2}\phi(7 - \cos\phi)(4 - \cos\phi)^{-2}. \tag{2.5}$$

In (2.5)

$$\phi = \frac{2\pi}{N} \tag{2.6}$$

is the angular separation of the point vortices. It is clear that

$$\Omega_N^{(0)} = \tfrac{1}{2} + O(N^{-1}), \tag{2.7}$$

and it can be shown that

$$\Omega_N^{(4)} = \tfrac{1}{2} + O(N^{-5}), \tag{2.8}$$

which demonstrates the improvement in accuracy obtained when Van der Vooren's correction is added to the point vortex formula; even higher accuracy can be obtained by going to higher order differences. It may be noted that Fink and Soh's (1978) idea of replacing the vortex sheet by a polygon with a point vortex at the center of each side of the polygon gives (2.2) if the polygon is escribed and $\Omega_N^{(0)} \sec^2 \frac{1}{2}\phi$ if the polygon is inscribed. In either case, the error is $O(N^{-1})$.

To examine the role of the discrete form of Helmholtz instability, (1.5) was integrated numerically both with and without the inclusion of Van der Vooren's correction. Fourth order Runge–Kutta integration was used with a fixed time step δt. This time step must be shorter than the shortest time scale associated with the evolution of small disturbances to the polygonal array. If N is large, this shortest time scale is approximated by the e-folding time T_N of the most unstable mode of a uniform *linear* array of point vortices each of strength $2\pi/N$ at spacing $2\pi/N$ for which (Lamb (1932, p. 226))

$$T_N = \frac{8}{N}. \tag{2.9}$$

The analysis can be repeated with Van der Vooren's correction included, to find that, when fourth order differences are employed, the most unstable mode grows more rapidly with the e-folding time T'_N, where

$$T'_N = \frac{8}{N}\left(1 + \frac{16}{3\pi^2}\right)^{-1}. \tag{2.10}$$

The mode shape is one of "pairing" and has a period of two spacings, so that its wavelength is $4\pi/N$. One may note that a wave of length $4\pi/N$ on a uniform vortex *sheet* of unit strength has an e-folding time $4/N$, which is half the corresponding value for an array. This might explain why making the spatial integration more accurate is reducing the e-folding time.

In the integration δt was chosen to be $T_N/5$ or $T_N/10$, the latter as a check.

The N point vortices were at positions

$$z_s(0) = \exp\{i(s-1)\phi\}, \qquad s = 1, 2, \cdots, N, \tag{2.11}$$

at time $t = 0$, so that according to the exact solution they are at positions

$$z_s(t) = z_s(0) \exp(i\Omega_N^{(r)} t), \qquad s = 1, 2, \cdots, N \tag{2.12}$$

at time t. If the calculated positions are $z'_s(t)$ then a set of complex errors $e_s(t)$ can be defined by

$$e_s(t) = z'_s(t) - z_s(t), \qquad s = 1, 2, \cdots, N. \tag{2.13}$$

In Fig. 1 is plotted the logarithm of the mean error $\bar{e}$ defined by

$$\bar{e}(t) = \frac{1}{N} \sum_{s=1}^{N} |e_s(t)|, \tag{2.14}$$

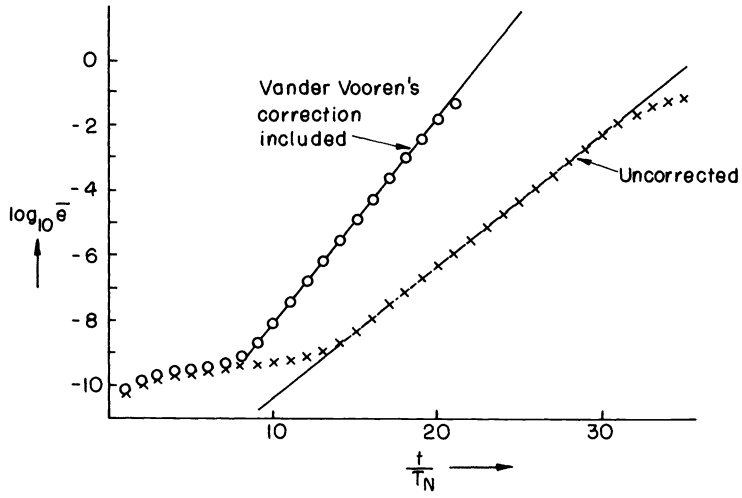


FIG. 1. *The error $\bar{e}(t)$ in the case $N = 60$, $\delta t = T_N/5$, with and without Van der Vooren's correction.*

both with and without Van der Vooren's correction, in the case $N = 60$. Fourth order differences were used to compute Van der Vooren's correction. In both cases, the mean error grows initially rather slowly. Next there is an extensive region of exponential growth, to which straight lines have been fitted. Finally the growth slackens off at values of $\bar{e}$ of about 0.1, due almost certainly to nonlinear coupling between modes.

If the principal claim of this paper is correct and the error in vortex calculations is due to the discrete form of Helmholtz instability, the growth rate σ of the error must be that of the most unstable mode of this instability. This maximum growth rate for a regular polygon of identical point vortices was calculated by Havelock (1931), and for large N his results give

$$\sigma = \frac{N}{8} - \frac{1}{2} + O\left(\frac{1}{N}\right),$$

which gives $\sigma = 7.0$[3] when $N = 60$, while the value derived from Fig. 1 is 6.99.

Havelock's analysis can be repeated with Van der Vooren's correction included, to predict an increased growth rate σ' given by

$$\sigma' = \frac{N}{8}\left(1 + \frac{16}{3\pi^2}\right) - \frac{1}{2} + O\left(\frac{1}{N}\right),$$

which gives $\sigma' = 11.05$ when $N = 60$ compared to a value 10.95 derived from Fig. 1.

The initial stage of slow growth can be interpreted as the stage in which the most unstable mode has not grown enough to leave the other modes behind. However, the length of this initial stage is about ten e-folding times of the most unstable mode, so that the question calls for further examination. This is attempted later.

Havelock in fact considered the evolution of more general disturbances to a regular polygon of N equal point vortices and showed that there are N normal modes in which the vortices are displaced to complex positions $\tilde{z}_s(t)$, where

$$\tilde{z}_s(t) - z_s(t) = z_s(t) w_s^{(p)}(t), \qquad s = 1, 2, \cdots, N, \tag{2.15}$$

[3] The exact value when $N = 60$ is $6.991066 \cdots$; see (2.17).

where p is a mode label taking on the values $0, 1, \cdots, N-1$. The function $w_s^{(p)}(t)$ gives the mode shape, and

$$w_s^{(p)}(t) = \varepsilon e^{\sigma_p t} \exp\{isp\phi\}, \tag{2.16}$$

where ε is a disturbance amplitude, assumed small, and σ_p is the growth rate given by

$$\sigma_p^2 = \tfrac{1}{4}p(N-p)N^{-2}(p(N-p)-2N+2). \tag{2.17}$$

The most unstable mode is $p = N/2$ (provided N is even), and this mode is one of pairing of neighboring vortices, just as in the case of the linear array.

If this instability is responsible for the failure of calculations in which a vortex sheet is replaced by point vortices, it ought to be possible to detect these modes by Fourier analysis of the computed complex errors $e_s(t)$. Thus complex Fourier coefficients

$$E_p(t) = \frac{1}{N}\sum_{s=1}^{N} e^{-is(p+1)\phi} e_s(t), \qquad p = 0, 1, \cdots, N-1 \tag{2.18}$$

were calculated. If $N = 60$, the modes $p = 30, 20, 15, 12$ and 10 have a periodicity of 2, 3, 4, 5 and 6 spacings and thus are analogues of modes which can occur in a linear array of vortices, the first being pairing. The growth of these modes is shown in Fig. 2 and their growth rates are compared with the values computed from Havelock's result (2.17) in Table 1. The agreement is good and there can be little doubt that Helmholtz instability—in its discrete form—is present. Note, however, that nonlinear effects occur at an early stage of the calculation.

TABLE 1
Comparison of theoretical and numerical growth rates

Periodicity	σ from (2.17)	σ from Fig. 2
2 spacings	6.991	6.91
3 spacings	6.155	6.17
4 spacings	5.110	5.15
5 spacings	4.280	4.32
6 spacings	3.642	3.62

These modes are not specially strongly excited and the complete spectrum is shown in Fig. 4. The instability is not obvious graphically until a late stage, as is illustrated in Figs. 3a, 3b and 3c. Moreover, the flatness of the spectrum explains why the effect of instability, when it does appear at a magnitude (of, say, 5%) which is graphically apparent, does not display any obvious modal features. The flatness of the spectrum is due in part to nonlinear effects.

Examination of the spectrum for small times explains the delay in the appearance of growth at the growth rate of the most unstable mode. The Fourier coefficient E_0 is $O(10^{-10})$ while the remaining coefficients are $O(10^{-14})$ at $t = 0.2667$ in the case $N = 60$, so that the most unstable mode is making a negligible contribution to the total error. It may be noted that an increase δr in the distance of each vortex from the center of the regular polygon gives an error

$$e_s = \delta r\, e^{is\phi}, \qquad s = 1, 2, \cdots, N, \tag{2.19}$$

which has $E_0 = \delta r$ and $E_p = 0$ $(1 \leqq p \leqq N-1)$. Thus E_0 represents a small *systematic* error, and it is natural to suppose that it is caused by the truncation error arising from

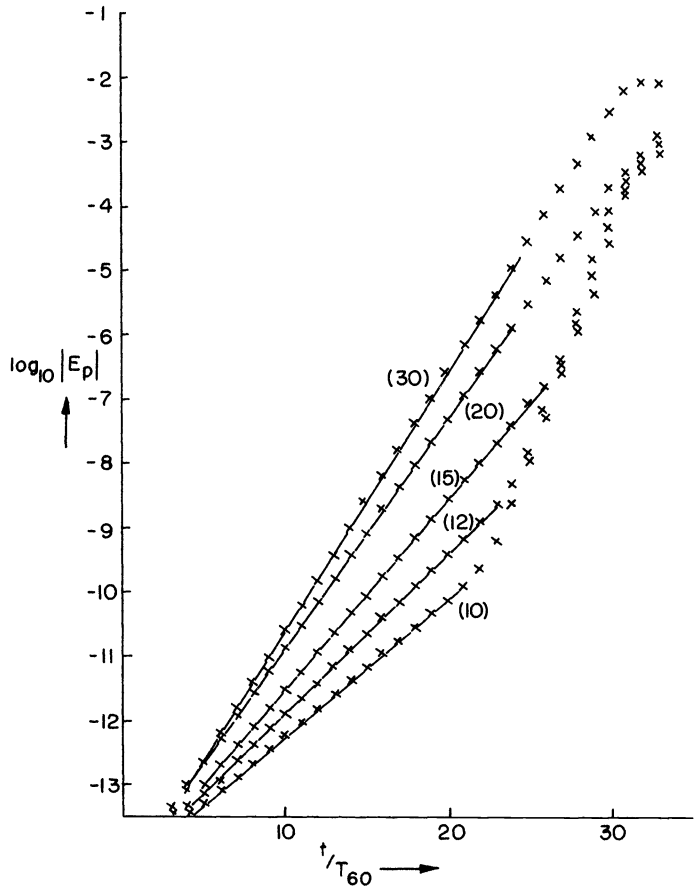


FIG. 2. *The growth of the modes* $p = 30, 20, 15, 12$ *and* 10 *in the case* $N = 60$ *and* $\delta t = T_N/5$. $\mathrm{Log}_{10}|E_p|$ *is plotted against* t/T_{60}. *Van der Vooren's correction was not applied.*

use of the fourth order Runge–Kutta formula. This was checked by noting that this error diminished by a factor $\frac{1}{16}$ when the time step was halved. However, there seemed no reason to reduce this truncation error below the value $O(10^{-10})$ by the use of very short time steps.

3. The suppression of discrete Helmholtz instability by linear smoothing. In this section the effect on the growth of short waves of the linear smoothing formulae used by Longuet-Higgins and Cokelet (1976) will be studied.

Linear formulae smooth by replacing the complex coordinate of each point vortex z_s by a linear combination $\hat{z}_s$ of the coordinates of itself and its neighbors, and "five-point"

$$\hat{z}_s = \frac{-z_{s-2} + 4z_{s-1} + 10z_s + 4z_{s+1} - z_{s+2}}{16} \tag{3.1}$$

and "seven-point"

$$\hat{z}_s = \frac{-z_{s-3} + 9z_{s-1} + 16z_s + 9z_{s+1} - z_{s+3}}{32} \tag{3.2}$$

formulae were proposed.

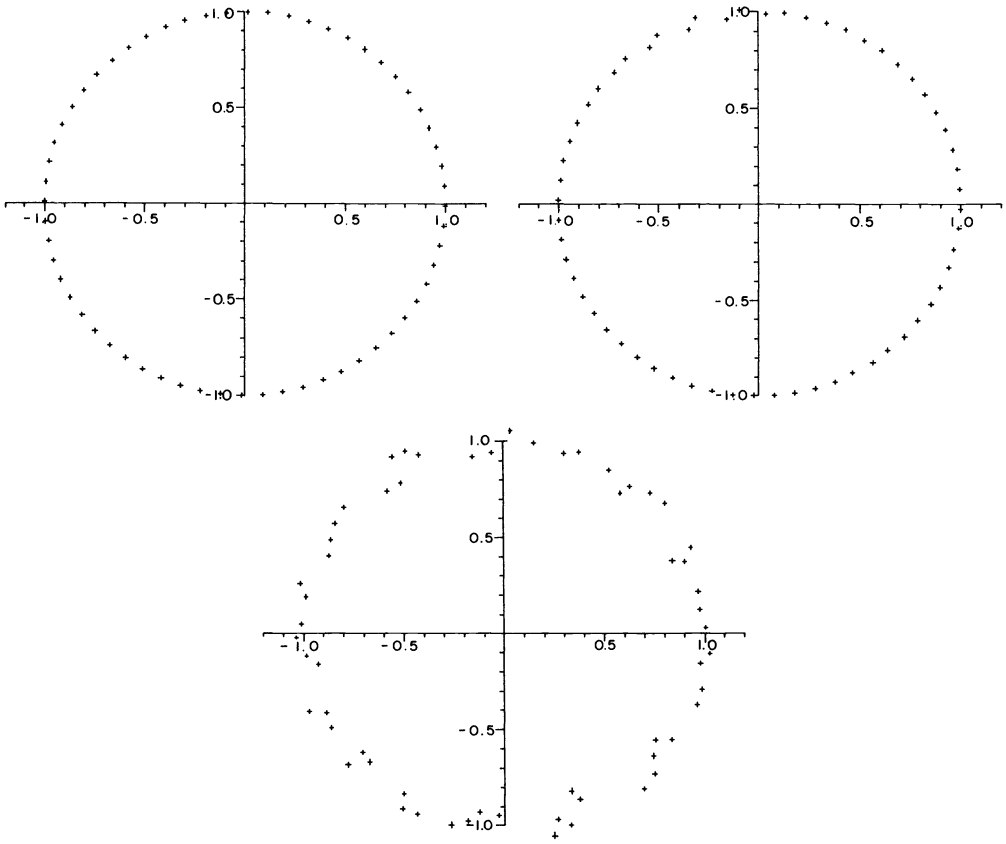


FIG. 3. *The appearance of the point vortices at* (a) $t = 3.6$, (b) $t = 4.0$, (c) $t = 5.6$. *Note that the modal structure is not apparent at* $t = 4.0$.

Both of these formulae can be written in the form

$$\hat{z}_s = A_0 + \sum_{r=1}^{M} A_r(z_{s+r} + z_{s-r}), \tag{3.3}$$

where, in order to ensure that the positions of equally spaced point vortices on a straight line are unchanged,

$$A_0 + \sum_{r=1}^{M} 2A_r = 1. \tag{3.4}$$

The effect of such a smoothing formula on the growth of waves on an array of point vortices representing an evolving vortex sheet can be studied by utilizing the analysis given by Von Karman of this representation of a straight, infinite uniform vortex sheet.

Suppose that an infinite linear array of point vortices have complex coordinates $nh + \tilde{z}_n(t)$ and equal strengths $\Delta\Gamma$, where $|\tilde{z}_n| \ll h$, where h is the spacing in the undisturbed state. Then, following Lamb (1932), the complex velocity $\rho_s(t)$ induced at the vortex at $sh + \tilde{z}_s$ is given approximately by

$$\rho_s(t) = \frac{i\Delta\Gamma}{2\pi h^2} \sum_{p \neq s} \frac{\tilde{z}_s(t) - \tilde{z}_p(t)}{(s-p)^2}, \tag{3.5}$$

where nonlinear terms have been dropped.

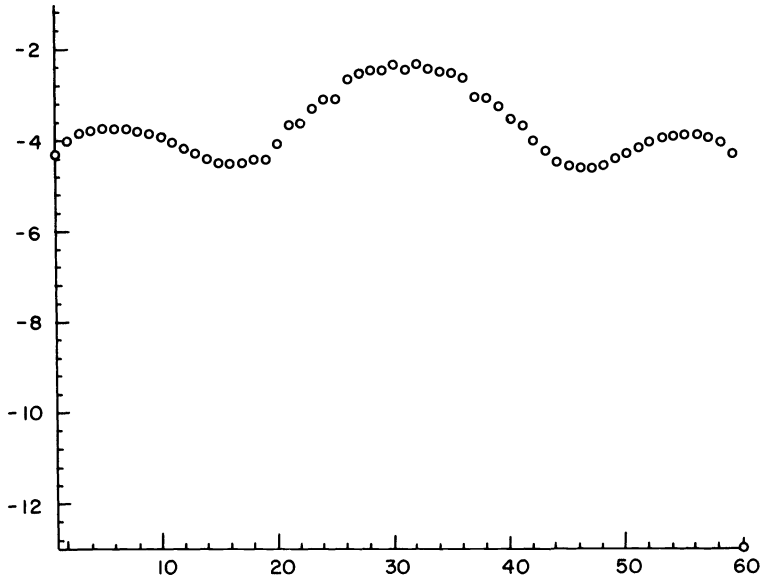


FIG. 4. *The spectrum at* $t = 4.0$ *in the case* $N = 60$ $\delta t = T_N/5$. $\mathrm{Log}_{10}|E_{p+1}|$ *is plotted against* p *to reveal the comparative flatness of the spectrum near the peak at* $p = 30$. (*Note that Fig.* 1 *to Fig.* 4 *were derived from distinct computer runs although the parameters* N *and* δt *were held fixed.*)

For simplicity, it will be assumed that Euler integration rather than Runge–Kutta integration is being used, and then it follows from the Euler integration formula that the vortices at time $t + \delta t$ will be $sh + z'_s$, where

$$z'_s = \tilde{z}_s(t) + \rho^*_s(t)\delta t + O(\delta t^2). \tag{3.6}$$

If the smoothing formula is applied to z'_s to obtain the new positions, then

$$\tilde{z}_s(t+\delta t) = A_0 z'_s + \sum_{r=1}^{M} A_r(z'_{s+r} + z'_{s-r}), \tag{3.7}$$

so that, from (3.6),

$$\tilde{z}_s(t+\delta t) = A_0\tilde{z}_s + \sum_{r=1}^{M} A_r(\tilde{z}_{s+r} + \tilde{z}_{s-r}) + \delta t\{A_0\rho^*_s + \sum_{r=1}^{M} A_r(\rho^*_{s+r} + \rho^*_{s-r})\}. \tag{3.8}$$

The evolution of a spatially periodic disturbance to the array will be examined, so that

$$\tilde{z}_s(t) = a(t)\exp\{is\psi\}, \tag{3.9}$$

where, if the mode has a period of R spacings,

$$\psi = \frac{2\pi}{R}, \qquad R = 2, 3, 4, \cdots. \tag{3.10}$$

Substitution of (3.9) into (3.5) and (3.8) leads, after some algebra, to an equation determining the evolution of the amplitude $a(t)$; this is

$$a(t+\delta t) = F(\psi)\left\{a(t) - \frac{i\Delta\Gamma\delta t}{\pi h^2}G(\psi)a^*(t)\right\}, \tag{3.11}$$

where

$$F(\psi) = 1 - 4 \sum_{r=1}^{M} A_r \sin^2\left(\frac{r\psi}{2}\right) \tag{3.12}$$

and

$$G(\psi) = \tfrac{1}{4}\psi(2\pi - \psi). \tag{3.13}$$

It can be verified that, for either of the smoothing formulae of Longuet-Higgins and Cokelet, the function F satisfies

$$0 < F \leqq 1, \tag{3.14}$$

and this condition will be assumed in what follows.

Equation (3.11) is a difference equation for $a(n\delta t)$ which has both a (possibly) growing and a decaying solution. The growing solution is

$$a(n\delta t) = \alpha_0(1-i)\mu^n, \qquad n = 0, 1, 2, \cdots, \tag{3.15}$$

where α_0 is a real constant and

$$\mu_R = F\left(\frac{2\pi}{R}\right)\left(1 + \frac{\Delta\Gamma\delta t}{\pi h^2} G\left(\frac{2\pi}{R}\right)\right). \tag{3.16}$$

If there is no smoothing, $F = 1$ and

$$\mu = 1 + \frac{\Delta\Gamma\delta t G}{\pi h^2}. \tag{3.17}$$

Thus in the limit $\delta t \to 0$ and $n \to \infty$, with $n\delta t = t$, the solution for $a(t)$ reduces to that given by Lamb.

In this case, the mode which grows most rapidly corresponds to $R = 2$ or $\psi = \pi$, because this makes G as large as possible. This disturbance is the shortest wave the discrete array can support and represents a pairing of adjacent vortices. The e-folding time of this mode is T, where

$$T = \frac{4h^2}{\pi\Delta\Gamma}; \tag{3.18}$$

T forms a useful time scale for the subsequent discussion. Since $\Delta\Gamma/h$ is the circulation per unit length in the vortex sheet, $T \to 0$ as $h \to 0$ so that the e-folding time of the most unstable mode becomes smaller as the number of point vortices is increased. Note that when $h = 2\pi/N$ and $\Delta\Gamma = 2\pi/N$, as in the regular polygon of identical vortices of § 2, (3.18) gives $T = T_N$.

If T is introduced into (3.16), it becomes

$$\mu_R = F\left(\frac{2\pi}{R}\right)\left(1 + \frac{4\delta t(R-1)}{TR^2}\right). \tag{3.19}$$

Since $F < 1$ for $\psi \neq 0$, the application of smoothing reduces the growth rate of all modes. Moreover, since $F(\pi) = 0$ for either (3.1) or (3.2), the most unstable mode $R = 2$ is suppressed. Higher modes are suppressed if $\mu_R < 1$.

Accurate Euler integration anyway requires $\delta t \ll T$, and in this case the disturbance evolves nearly continuously, with a growth rate $\bar{\sigma}_R$ given by

$$\bar{\sigma}_R = \frac{1}{T}\left\{\frac{4(R-1)}{R^2} + \frac{T}{\delta}\log\left(1 - \sin^4\left(\frac{\pi}{R}\right)\right)\right\}, \tag{3.20}$$

where, for definiteness, the smoothing formula (3.1) has been used to evaluate F. For example, if $\delta t = T/5$, as in most of the calculations presented here, $\bar{\sigma}_R$ is negative for $R = 2, 3, 4$, and is positive for $R > 5$ with a maximum when $R = 8$; however, σ_5 is only 0.004, so that mode is effectively suppressed.

The smoothing formula (3.1) was applied to the case of the circular vortex sheet studied in the previous section. Fig. 5 shows the spectrum of the error in case $N = 60$, $t = T_N/5$. Only the wings of the spectrum grow (so long as nonlinear effects are negligible) and, as anticipated, the most unstable modes are suppressed. The analysis of this section cannot strictly be applied to a polygonal rather than a linear array. However, the Havelock mode of mode label p has a period of approximately $R = [N/p]$ spacings, and if the above discussion of (3.20) is applied it suggests that $0 \leqq p \leqq 10$ (and $59 > p > 49$) are unstable, in fair agreement with Fig. 5. The predicted maximum growth rate is $0.329T^{-1}$, achieved when $p \sim 7$. Chaotic motion had not appeared at $t = 12$, but had appeared at $t = 15$. Thus a postponement by a factor of roughly 3 has been attained.

It is not easy to estimate a priori the errors introduced by this type of smoothing. The condition (3.4) imposed on the smoothing coefficients A_r ensures that

$$\Delta\Gamma \sum_{s=1}^{N} \hat{z}_s = \Delta\Gamma \sum_{s=1}^{N} z_s, \tag{3.21}$$

so that the linear momentum is unchanged by the smoothing. However, it has not been possible to estimate a priori the effect of the smoothing on the angular momentum or kinetic energy.

Linear smoothing has one adverse effect which is easily demonstrated by applying it to the test case of the regular polygon. If z_s is given by (2.11) then

$$\hat{z}_s = z_s F(\phi), \tag{3.22}$$

where, as in § 2, $\phi = 2\pi/N$ is the angular separation of the point vortices. Thus, since $F < 1$, the radius of the polygon is reduced each time smoothing is applied. This error will arise in any situation, such as in rolling up, where nearly circular vortex sheets occur. If $N = 20$ the reduction in radius is by a factor 0.999401 for formula (3.1) and 0.998233 for formula (3.2).[4]

If N is large, then it can be shown that, in the case of smoothing formula (3.1),

$$F = 1 - \frac{\pi^4}{N^4} + O\left(\frac{1}{N^6}\right). \tag{3.23}$$

Suppose a time step $\delta t = T_N/I$ is used and integration is carried forward to the time 4π corresponding to a complete revolution. Then $4\pi I/T_N$ time steps are needed, so that in view of (3.23) and (2.9) the radius of the polygon when $t = 2\pi$ is

$$\exp\left\{-\frac{I\pi^5}{2N^3}\right\}.$$

Thus the error in radius tends to zero with increasing N and is merely 0.35% in the case $N = 60$ and $I = 5$.

It was shown in § 3 that a change δr in the radius of each vortex produces a Fourier coefficient E_0 of $O(\delta r)$ and zero higher coefficients. The effect of the bias in the smoothing formula (3.1) is apparent in Fig. 5.

[4] This may in part explain why Longuet-Higgins and Cokelet found the first formula gave greater accuracy in practice.

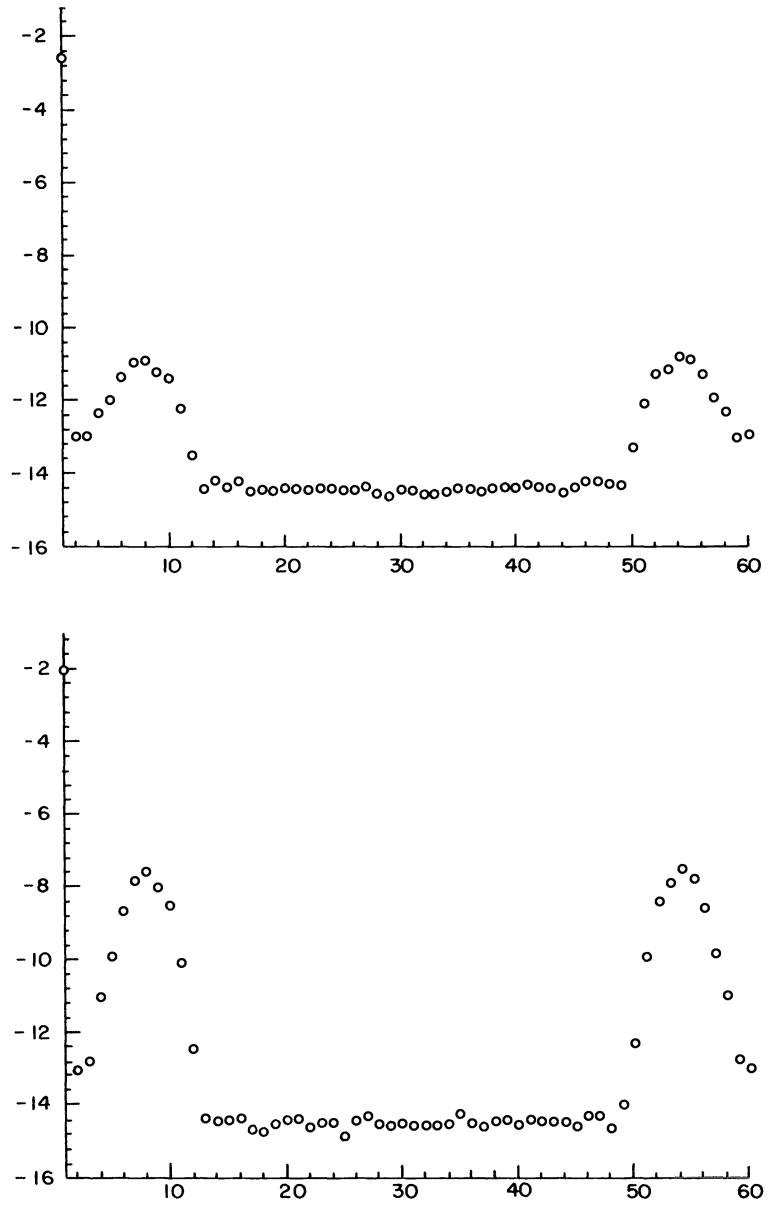


FIG. 5. $\mathrm{Log}_{10}|E_{p+1}|$ *as a function of* p *in the case* (a) $t = 4.0$ *and* (b) $t = 8.0$, *with smoothing given by* (3.1) *and with* $N = 60$ *and* $\delta t = T_N/5$. *The effect of the bias is noticeable in the large value of* $|E_0|$.

The error can be further reduced by applying smoothing only every S time steps—this was Longuet-Higgins' and Cokelet's procedure. But then it can be shown that (3.20) is replaced by

$$\bar{\sigma}_R = \frac{1}{T}\left\{\frac{4(R-1)}{R^2} + \frac{T}{S\delta t}\log\left(1-\sin^4\frac{\pi}{R}\right)\right\}. \tag{3.24}$$

Thus $R = 2$ remains suppressed, but the stabilizing effect of the smoothing is reduced for the higher modes. An assessment based on the value of N and the time for which

stabilization is needed must be made to see if this option is useful, given the likely initial errors.

4. The repositioning technique. The repositioning technique was introduced by Fink and Soh (1978) in an attempt to improve the accuracy of the point vortex method. In this section, its effect on the discrete Helmholtz instability of a uniform linear array of point vortices is examined.

Fink and Soh's repositioning technique was designed to deal with a vortex sheet with ends, and this leads to complications which can be avoided in the present case. Thus a simplified version of the technique is described here.

Suppose that at some instant t the closed vortex sheet is described by N point vortices of strengths $\Delta\Gamma_p$ at positions z_p, where $p = 1, 2, \cdots, N$. The first step is to calculate the arc distance s_p of vortex p from vortex 1 using the equation

$$s_p - s_{p-1} = |z_p - z_{p-1}|, \tag{4.1}$$

where $s_1 = 0$ and where, if p falls outside the range $1 \leqq p \leqq N$, the periodicity of z

$$z_{p+N} = z_p \tag{4.2}$$

is used. With this definition, the length of the vortex sheet is $L = s_{N+1}$.

The next step is to reposition the vortices $p = 2, 3, \cdots, N$ at $\hat{z}_p$, where the new coordinates are chosen so that the vortices are equally spaced in arc distance from the vortex 1. Thus the repositioned vortices have arc coordinates $\hat{s}_p$, where

$$\hat{s}_p = \frac{(p-1)L}{N}. \tag{4.3}$$

If the repositioning is carried out frequently, $\hat{z}_p \approx z_p$ and the repositioning can be achieved by interpolation. The function $z(s)$ is known at unequally spaced points $s = s_p$, so that using three-point Lagrange interpolation, for example, gives

$$\begin{aligned}\hat{z}_p = {} & z_{p-1}\frac{(\hat{s}_p - s_p)(\hat{s}_p - s_{p+1})}{(s_{p-1} - s_p)(s_{p-1} - s_{p+1})} + z_p\frac{(\hat{s}_p - s_{p-1})(\hat{s}_p - s_{p+1})}{(s_p - s_{p-1})(s_p - s_{p+1})} \\ & + z_{p+1}\frac{(\hat{s}_p - s_{p-1})(\hat{s}_p - s_p)}{(s_{p+1} - s_{p-1})(s_{p+1} - s_p)}, \qquad p = 2, 3, \cdots, N.\end{aligned} \tag{4.4}$$

Finally, the new vortex strengths are calculated. To do this the sheet strength $\gamma(s)$, defined by

$$\gamma(s) = \frac{\partial\Gamma}{\partial s}, \tag{4.5}$$

is calculated at $s = s_p$ from

$$\gamma(s_p)\frac{(s_{p+1} - s_{p-1})}{2} = \Delta\Gamma_p, \tag{4.6}$$

where an estimate of the arc length of sheet represented by the vortex p has been used. Then $\gamma(\hat{s}_p)(p = 1, 2, \cdots, N)$ is calculated from the set of values $\gamma(s_p)$ using the interpolation formula (4.4) and invoking the periodicity of $\gamma(s_p)$ if necessary. With $\gamma(\hat{s}_p)$ in hand, the new point vortex strengths follow from

$$\Delta\hat{\Gamma}_p = \gamma(\hat{s}_p)\left(\frac{L}{N}\right). \tag{4.7}$$

Clearly the repositioning produces no change in coordinates or strengths when applied to a regular polygonal array, so that the bias introduced by linear smoothing is absent.

For the case of the slightly perturbed uniform linear array, the effect of repositioning can be discussed analytically. If, as before, the vortices are at positions

$$z_p = ph + \tilde{z}_p,$$

then

$$|z_{p+1} - z_p| = h + \tilde{x}_{p+1} - \tilde{x}_p + O(|\tilde{z}_p|^2), \tag{4.8}$$

so that, in view of (4.1),

$$s_p = (p-1)h + \tilde{x}_p - \tilde{x}_1 + O(|\tilde{z}_p|^2). \tag{4.9}$$

This approximation can be substituted into (4.4) and the corresponding interpolation formula for γ to show that, denoting repositioned quantities by $\hat{}$,

$$\hat{\tilde{z}}_p = i\tilde{y}_p + x_1, \qquad p = 1, 2, \cdots, N, \tag{4.10}$$

and

$$\gamma(\hat{s}_p) = \gamma(s_p), \qquad p = 1, 2, \cdots, N; \tag{4.11}$$

however, variations of the point vortex strengths are induced by the change of length, and, if the vortex with label p has strength $\Delta\Gamma + \tilde{\delta}_p$,

$$\hat{\tilde{\delta}}_p = \tilde{\delta}_p - \frac{\Delta\Gamma}{2h}(x_{p+1} - x_{p-1}). \tag{4.12}$$

Von Karman's analysis can now be repeated, allowing for the variations of point vortex strength induced by the repositioning. Thus the complex velocity ρ_s is given by

$$\rho_S = \frac{i\Delta\Gamma}{2\pi h^2} \sum_{p \neq s} \frac{\tilde{z}_s - \tilde{z}_p}{(s-p)^2} - \frac{i}{2\pi h} \sum_{p \neq s} \frac{\tilde{\delta}_p}{(s-p)}, \tag{4.13}$$

instead of (3.4).

The analysis of § 3 can now be repeated; for the mode defined by (3.9)[5] and (3.10) the growth factor μ_R is given by

$$\mu_R = 1 + \frac{\delta t}{T}\left[\frac{8}{\pi R}\left(1 - \frac{1}{R}\right)\left(1 + \frac{2}{R}\right)\sin\left(\frac{2\pi}{R}\right)\right]^{1/2}, \tag{4.14}$$

where δt is the time step of Euler integration and where T, defined in (3.18), is the natural time scale of the uniform linear array. Clearly μ_2 is unity, so that the most unstable mode of discrete Helmholtz instability is suppressed. Evidently $\mu_R > 1$ for $R \geqq 3$, so that the higher modes still amplify when repositioning is adopted. However, numerical examination of the result shows that the growth factor of the modes is reduced by the application of repositioning. The mode $R = 4$ (representing a periodicity of four spacings) is the most unstable, and its growth rate is roughly one half of the growth rate of the most unstable mode in the undamped case. Thus the time to the appearance of a given level of error should be increased by a factor of roughly two when

[5] Actually it is essential to include both $e^{+is\psi}$ and $e^{-is\psi}$ dependences in the calculation and also a term independent of s; the lengthly algebraic details will not be given.

repositioning is applied. A detailed comparison of the growth factors is given in Table 2, where values of ν, defined by

$$\mu = 1 + \frac{\delta t}{T}\nu, \tag{4.15}$$

are compared.

TABLE 2
Growth factors $1+\nu(\delta t/T)$ *without and with repositioning*

Periodicity	ν (discrete Helmholtz)	ν (eqn. (4.14))
2 spacings	1.0	0.0
3 spacings	.889	.404
4 spacings	.750	.489
5 spacings	.640	.482
6 spacings	.556	.452

The version of the repositioning technique described above was applied to the uniform circular array. As in § 3, 60 vortices were used and repositioning was applied at each time step, the time step being, as before, $T_N/5$. The errors became graphically apparent at about $t = 8.0$, or at twice the time in the unsmoothed case of § 2, as anticipated. However, the motion was *not chaotic* (Fig. 6) although it became so eventually.

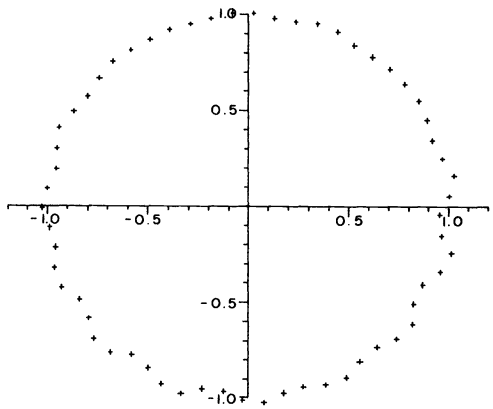


FIG. 6. *The point vortices at* $t = 8.4$ *in the case* $N = 60$, $\delta t = T_N/5$ *when repositioning was used. A spurious wave has emerged.*

The spectrum of the error at $t = 4.0$ is shown in Fig. 7. Clearly disturbances with wavelength roughly $[60/15] = 4$ spacings are being most strongly amplified. This is in agreement with the conclusions of the analysis for the uniform straight array.

5. Further tests and discussion. The work presented in the previous sections has shown that, in the case of the uniform circular vortex sheet, chaotic motion of the representing point vortices is brought about, not by inaccurate numerical integration, but by the growth of discrete Helmholtz instability. This chaotic motion could be delayed by either linear smoothing or repositioning, the latter acting as a nonlinear

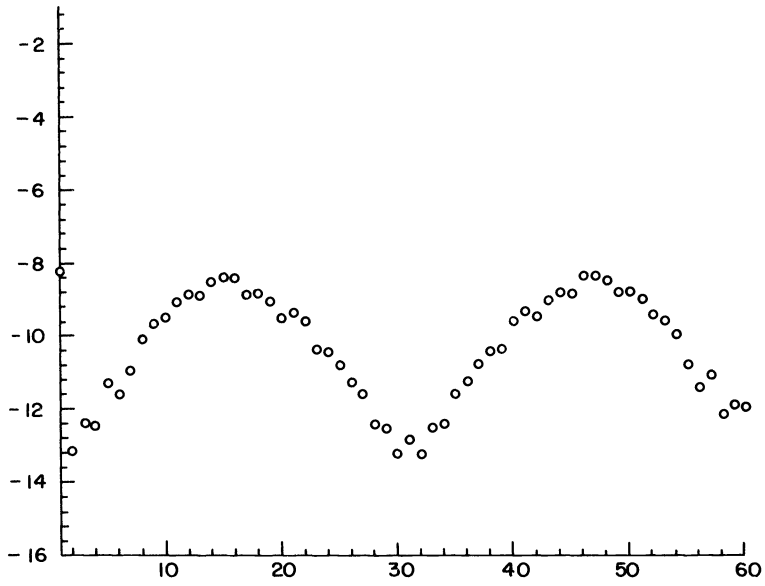


FIG. 7. *The spectrum of error at* $t = 4.0$ *in the case* $N = 60$, $\delta t = T_N/10$ *when repositioning was used, showing most rapid growth for* $p \sim 15$ *and* $p \sim 45$.

smoothing process. This delay in the onset of chaotic motion was due to suppression of the more unstable modes of discrete Helmholtz instability.

The disadvantage of smoothing is that it is not clear in general what the relationship is between the results achieved and the unknown exact solution of (1.1). As remarked in § 1, the initial value problem for (1.1) may not possess a solution for all time, and in this case the use of smoothing could yield an acceptable-looking solution where none in fact exists. However, this is perhaps too pessimistic. The actual problem of interest in aerodynamics is the evolution on a thin vortex layer of features the scale of which is large compared with the vortex layer thickness. To the extent that the growth of short Helmholtz waves is suppressed by smoothing, smoothing is turning the vortex sheet into a thin vortex layer; however, no precise effective thickness can be defined for either smoothing method. Nevertheless, gross features of smoothed calculations can probably be accepted as real. In contrast, the spurious wave produced on the uniform circle in a repositioning calculation (Fig. 6) shows that features involving few vortices cannot be accepted as real.

To emphasize these points, smoothing was tried for the initial value problem in which

$$z(\Gamma, 0) = \exp\{i(\Gamma - 0.5 \sin \Gamma)\}; \tag{5.1}$$

this represents a circular vortex sheet whose strength $\gamma(\Gamma)$ is given by

$$\gamma = (1 - 0.5 \cos \Gamma)^{-1}; \tag{5.2}$$

of course, there is no guarantee that a solution exists for $t > 0$.

The nonuniformity of γ means that the sheet distorts from the circular for $t > 0$. In Fig. 8 are displayed the results at $t = 0.5$ of calculations using 80 and 160 vortices and both types of smoothing. The gross features agree, but the details of the small rolling-up feature differ. Repositioning was more successful at preventing chaotic motion in this case. The linear smoothing calculation failed at $t = 0.6$ (when $N = 160$, it failed earlier),

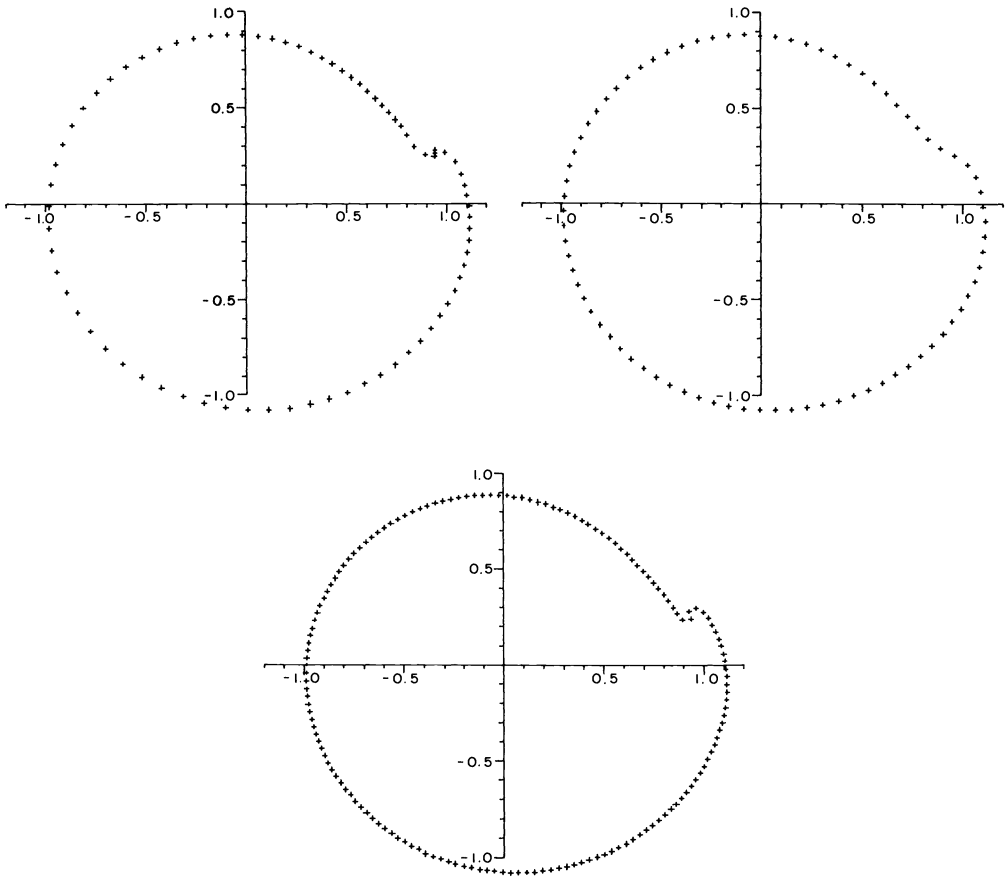


FIG. 8. *Comparison of results with at* $t = 0.5$ *with* (a) *linear smoothing,* $N = 80$ *and* $\delta t = 0.01$; (b) *repositioning,* $N = 80$ *and* $\delta t = 0.01$; *and* (c) *repositioning,* $N = 160$ *and* $\delta t = 0.01$.

whereas the repositioning results for $N = 80$ remained smooth up to $t = 1.0$, when the integration was stopped. This is surprising in view of the rather better performance of linear smoothing in the case of the uniform circle. However, the test of linear smoothing used equal strength vortices, unequally spaced, while that of repositioning used unequal strength vortices, equally spaced. Now the growth rate of the most unstable wave on a uniform array of point vortices at spacing h representing a straight uniform vortex sheet of strength γ is $\pi\gamma/4h$. Clearly, it is an advantage to prevent small values of h arising in a calculation, and repositioning achieves this.

The values of the invariants

$$I_1 = \sum_{r=1}^{N} \Delta\Gamma_r,$$

$$I_2 = \sum_{r=1}^{N} \Delta\Gamma_r z_r$$

$$I_3 = \sum_{r=1}^{N} \Delta\Gamma_r z_r z_r^*,$$

and the Hamiltonian

$$I_4 = -\frac{1}{2\pi} \sum_{\text{all pairs}} \Delta\Gamma_r \, \Delta\Gamma_s \log |z_r - z_s|,$$

at $t = 0$ and $t = 0.5$ were compared in the case $N = 80$ and $\delta t = .01$. As remarked, linear smoothing conserves I_1 and I_2 exactly. Repositioning does not conserve them exactly, but the variation in I_1 was only 0.0002%, and in I_2, 0.002%. Neither method conserves I_3, linear smoothing producing a .06% variation and repositioning producing a .04% variation.

The results for I_4 proved difficult to interpret. This is because the numerical values of I_4 happened to be small, being initially 0.0355 in the linear smoothing run and 0.0114 in the repositioning run. Thus the Hamiltonian for the vortex sheet (5.1), which can be shown to be $0.210\cdots$, was used to form percentage variations which then turned out to be 10% for linear smoothing and 1% for repositioning. (The discrepancy between the values of the Hamiltonians of the discrete and continuous systems is $O(\log N/N)$.) Clearly repositioning is causing less variation in the invariants than linear smoothing in this case.

Appendix: Proof of Van der Vooren's formula for closed vortex sheets. The objective is to establish (1.3). To do this the behavior of the integrand near the singularity must be determined. As in § 1, define

(A1) $$z_p = z(\Gamma_p, t),$$

where

(A2) $$\Gamma_p = p\Delta\Gamma,$$

and note that Taylor's theorem gives

(A3) $$z(\Gamma, t) = z_s + (\Gamma - \Gamma_s)\left(\frac{\partial z}{\partial \Gamma}\right)_s + \frac{1}{2}(\Gamma - \Gamma_s)^2\left(\frac{\partial^2 z}{\partial \Gamma^2}\right)_s + \cdots.$$

Thus

(A4) $$\frac{1}{z_s - z(\Gamma, t)} = -\frac{1}{(\Gamma - \Gamma_s)\left(\frac{\partial z}{\partial \Gamma}\right)_s} + \frac{1}{2}\frac{\left(\frac{\partial^2 z}{\partial \Gamma^2}\right)_s}{\left(\frac{\partial z}{\partial \Gamma}\right)^2} + \psi(\Gamma - \Gamma_s),$$

where ψ is an analytic function of $\Gamma - \Gamma_s$ which vanishes at $\Gamma = \Gamma_s$. This suggests writing

(A5) $$\int_0^{\Gamma_e} \frac{d\Gamma}{z_s - z(\Gamma, t)} = \int_0^{\Gamma_e} \left\{\frac{1}{z_s - z(\Gamma, t)} + \frac{1}{(\Gamma - \Gamma_s)(\partial z/\partial \Gamma)_s}\right\} d\Gamma - \int_0^{\Gamma_s} \frac{d\Gamma}{(\Gamma - \Gamma_s)(\partial z/\partial \Gamma)_s},$$

since the second integral on the right can be evaluated explicitly and any convenient integration formula can be used to evaluate the first integral on the right.

Since $z(\Gamma, t)$ is periodic, trapezoidal integration is a natural choice. The application of this integration rule is simplest if the periodicity of the integrand in the integral on the left of (A5) is invoked to change the range of integration $(\Gamma_s - \frac{1}{2}\Gamma_e, \Gamma_s + \frac{1}{2}\Gamma_e)$. Then the

second integral on the right vanishes and the first integral on the right is

$$\sum_{p\neq s}\frac{\Delta\Gamma}{z_s-z_p}+\sum_{p\neq s}\frac{\Delta\Gamma}{(p-s)\left(\frac{\partial z}{\partial\Gamma}\right)_s}+\frac{1}{2}\Delta\Gamma\frac{\left(\frac{\partial^2 z}{\partial\Gamma^2}\right)_s}{\left(\frac{\partial z}{\partial s}\right)_s^2}$$

The middle term vanishes[6] and (1.3) follows. Note that the error is exponentially small if the exact values of the derivatives are used. This is because the integrand in the first integral on the right, while not periodic because of the second term, is such as to make no contribution to the error as estimated by the Euler–Maclaurin formula.

Acknowledgments. The author has benefited from discussions with Mr. S. P. Fiddes, Professor P. G. Saffman and Mr. J. H. B. Smith. Professor H. B. Keller made some valuable comments on the first draft of the paper.

REFERENCES

G. R. BAKER (1980), *A test of the method of Fink and Soh for following vortex sheet motion*, J. Fluid Mech., 100, pp. 209–220.

M. DHANAK (1980), Imperial College Ph.D. Thesis.

S. FIDDES (1980), unpublished.

P. T. FINK AND W. K. SOH (1978), *A new approach to roll-up calculations of vortex sheets*, Proc. Roy. Soc. Ser. A, 362, pp. 195–209.

T. H. HAVELOCK (1931), *The stability of motion of rectilinear vortices in ring formation*, Phil. Mag., 11, pp. 617–633.

E. ISAACSON AND H. B. KELLER (1966), *Analysis of Numerical Methods*, John Wiley, New York.

H. LAMB (1932), *Hydrodynamics*, Cambridge University Press, London.

M. S. LONGUET-HIGGINS AND E. L. COKELET (1976). *The deformation of steep surface waves on water* I. *A numerical method of computation*, Proc. Roy. Soc Ser. A, 350, pp. 1–26.

B. MASKEW (1977), *Subvortex technique for close approach to a vortex sheet*, J. Aircraft, 14, pp. 188–193.

D. W. MOORE AND R. GRIFFITH-JONES (1974), *The stability of an expanding circular vortex sheet*, Mathematika, 21, pp. 128–133.

D. W. MOORE (1976), *The stability of an evolving two-dimensional vortex sheet*, Mathematika, 23, pp. 35–44.

——— (1978), *The equation of motion of a vortex layer of small thickness*, Stud. Appl. Math., 58, pp. 119–140.

——— (1979), *The spontaneous appearance of a singularity in the shape of an evolving vortex sheet*, Proc. Roy. Soc. Ser. A, 365, pp. 105–119.

LORD RAYLEIGH (1945), *Theory of Sound*, Dover, New York.

L. ROSENHEAD (1931), *The formation of vortices from a surface of discontinuity*, Proc. Roy. Soc. Ser. A, 134, pp. 170–192.

P. G. SAFFMAN AND G. R. BAKER (1979), *Vortex interaction*, Ann. Rev. Fluid Mech., 11, pp. 95–122.

T. SARPKAYA AND R. L. SHOAFF (1979), *An inviscid model of two-dimensional vortex shedding for transient and asymptotically steady separated flow over a cylinder*, AIAA J., 17, pp. 1193–1200.

A. I. VAN DER VOOREN (1965), *A numerical investigation of the rolling-up of vortex sheets*, unpublished report.

[6] if the number of points is odd. The result is true when N is even, but the proof is more complicated.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0007 $01.00/0

CONVERGENCE OF RANDOM METHODS FOR A REACTION-DIFFUSION EQUATION*

OLE H. HALD†

Abstract. A reaction-diffusion equation is solved on a grid by a fractional step method, which combines a deterministic solution of an ordinary differential equation with a random walk solution of the heat equation. We prove that the expected value of the computed solution tends to the solution of the reaction-diffusion equation and that the variance tends to zero. The rate of convergence is proportional to the mesh length.

Key words. convergence, random walk, reaction-diffusion

Introduction. Nonlinear partial differential equations can be solved by finite differences, finite elements, Fourier methods, pseudo-spectral methods and random methods. For Navier–Stokes equations the convergence has been established for at least one method in each category except the last. Here the solution is approximated by vortex blobs in two dimensions [2] and by vortex filaments or vortex segments in three dimensions [5], [7]. Another random walk technique is the vortex sheet method [4], which is used to approximate the solution of Prandtl boundary layer equations. The equations are solved by a fractional step technique. The nonlinear part of the equations is approximated by a system of ordinary differential equations, and the solution of this system is used to transport the computational elements. The linear part of the equations, which is due to viscosity, is then solved by a random walk technique. It is very difficult to prove the validity of these methods because of the interaction between the linear and the nonlinear parts of the equations and the combination of deterministic and random techniques, and previous attempts [18] have been unsatisfactory. However, there exist partial results. Thus Chorin et al. [3] have established the consistency of the vortex method, even including the creation of the vorticity at the boundary, and the convergence of the vortex method has been proved by Del Prete and Hald [11], [12] for flows of an inviscid fluid in a domain without boundary conditions. For three dimensions Beale and Majda have shown that the vortex segment method converges under similar assumptions (private communication). Finally, Beale and Majda have proved the convergence of a fractional step scheme for Navier–Stokes equations in two dimensions by leaving out the space discretization (private communication).

In this paper we will consider a model problem, namely the reaction-diffusion equation, and prove the convergence of a random walk method. The equation describes the temperature distribution in an infinitely long tube filled with combustible gas mixture. As the gas is ignited, the release of chemical energy will raise the temperature and flames will propagate through the tube. This can be described by a parabolic equation with a nonlinear source term. This equation admits traveling wave solutions, and the existence, uniqueness and stability of these solutions has recieved considerable attention, starting with Kolmogorov, Petrovskii and Piskunov [15]. For a recent survey and new results, see Lin [16].

Chorin [6] has proposed a numerical algorithm for solving the reaction-diffusion equation. The basic idea is to approximate the temperature gradient by a distribution of particles. The masses of the particles are determined by solving a system of ordinary differential equations. The positions of the particles are determined by a grid-free

* Received by the editors September 15, 1980.

† Department of Mathematics, University of California, Berkeley, California 94720. The author is an Alfred P. Sloan Research Fellow. This work was supported in part by the U.S. Office of Naval Research under grant N00014-76-C-0316, and the National Science Foundation under grant MCS79-05791.

random walk technique. I have been unable to prove the convergence of this algorithm. However, if the random walk is restricted to a mesh, and some technical assumptions are satisfied, then the convergence can be established, as proved below. The main difference between Chorin's method and ours is that Chorin uses only one particle per mesh point, while we create so many particles at each mesh point that the computed solution is very close to the solution of a finite difference equation. However, our method is not a Monte Carlo solution of a finite difference equation. Chorin's algorithm is of interest because it is a model for the boundary layer technique, but for the reaction-diffusion equation itself there exist more efficient algorithms; see Dwyer et al. [8].

The reaction-diffusion equation has been treated from a probabilistic point of view by McKean [17] and Bramson [1]. The goal is to determine the speed of the traveling wave, and the solution of the differential equation is expressed as the expected value of a functional of a branching Brownian motion. This approach restricts the class of reaction-diffusion equations which can be studied. In particular, it excludes the Hodgkin–Huxley equation and reaction-diffusion equations with source terms based on Arrhenius' expression for the chemical reaction rate. Our convergence proof includes these cases, which are of interest from a physical point of view.

1. The method. In this section we will present the reaction-diffusion equation and a random walk algorithm. The reaction-diffusion equation can be written as a nonlinear parabolic equation of the form

$$u_t = \nu u_{xx} + f(u), \qquad u(x, 0) = g(x), \tag{1}$$

where $-\infty < x < \infty$, ν is the viscosity and f is continuously differentiable on $[0, 1]$ with $f(0) = f(1) = 0$. This equation has two stationary solutions, namely $u \equiv 0$ and $u \equiv 1$. Thus if $0 \leqq g(x) \leqq 1$ then it follows from the maximum principle [10, p. 43] that $0 \leqq u(x, t) \leqq 1$ for all x and all $t > 0$. We assume now that $u \to 0$ as $x \to \infty$ and write $u(x, t) = \int_x^\infty \xi(s, t)\, ds$. To find an approximate solution of (1) we use the fractional step scheme

$$u_t = f(u), \tag{2}$$

$$\xi_t = \nu \xi_{xx}. \tag{3}$$

Here (3) is derived by differentiating $u_t = \nu u_{xx}$ with respect to x. Why do we use the temperature gradient instead of the temperature? Because we want the variance of the computed solution to be small. Thus, if $u_t = \nu u_{xx}$ is solved on a grid by a random walk method with one particle per mesh point, then the variance at a fixed point does not go to zero. On the other hand, if $\xi_t = \nu \xi_{xx}$ is solved by a random walk method, either on a grid or by using Gaussian distributed random variables, and $u(x, t)$ is approximated by the mass of the particles to the right of the point x, then the variance of the computed solution u is proportional to the mesh length, provided we have one particle per mesh point initially.

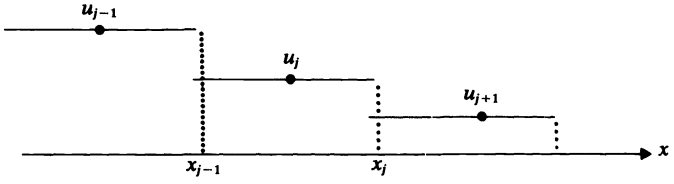


FIG. 1. *Initial data.*

To solve (2) and (3) numerically, we introduce a mesh $x_j = jh$ and approximate u and ξ by mesh functions $u_j = u(x_j - h/2)$ and $\xi_j = \xi(x_j)$; see Fig. 1. Since (2) is an ordinary differential equation, we solve it by Euler's method,

$$v_j = u_j + kf(u_j). \tag{4}$$

Here k is the time step. This method is easy to analyze and sufficiently accurate for our purpose. Our next step is to solve the diffusion equation by a random walk technique [9, p. 323]. We get the initial data by numerical differentiation,

$$\xi_j = \frac{v_j - v_{j+1}}{h}. \tag{5}$$

At x_j we place a particle with mass $\kappa_j = h\xi_j$. Let $X = X^{(i)}$ be a random variable which assumes the values -1, 0 and 1 with probabilities $\nu/2$, $1-\nu$ and $\nu/2$; here $\nu > 0$ is the viscosity. If $X = 1$ then the particle at x_j jumps to x_{j+1}. If $X = -1$ then it jumps to x_{j-1}. Finally, if $X = 0$ then the particle stays at x_j; see Fig. 2.

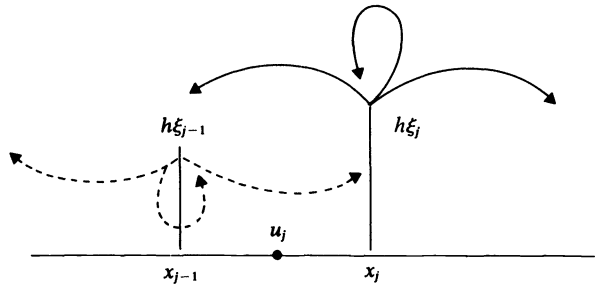


FIG. 2. *Random solution of the heat equation.*

We introduce now the indicator function for X: we set $I_i(X) = 1$ if $X = i$ and 0 otherwise. The random walk solution of (3) is then given by

$$h\xi_j = I_1(X^{(j-1)})\kappa_{j-1} + I_0(X^{(j)})\kappa_j + I_{-1}(X^{(j+1)})\kappa_{j+1}. \tag{6}$$

This amounts to redistributing the mass at the mesh points. Finally we compute the solution at the next time step by

$$u_j = \sum_{l=j} h\xi_l. \tag{7}$$

We see that the mass of a particle is found by a deterministic method, and the position by a random walk technique. The particles do not exist for long. After each time step the total mass at a mesh point is given to one particle. I cannot prove the convergence of this algorithm, because I cannot show that the variance of u_j tends to zero. In Chorin's algorithm [6] the particles exist for a long time, but if the mass of a particle becomes too large, then the particle is split into two and each gets half of the mass. We will use a similar idea. Let $\alpha > 0$. If $\xi_j \neq 0$, then we determine a positive integer N_j such that

$$N_j - 1 < \frac{|h\xi_j|}{\alpha} \leqq N_j. \tag{8}$$

At x_j we place N_j particles with mass $\kappa_j = h\xi_j/N_j$ and let each particle jump as before. Thus we create many particles where they are needed most, namely where the temperature gradient is large. The random walk solution of (3) is now given by

$$h\xi_j = \sum_{i=-1}^{1} \sum_{l=1}^{N_{j-i}} I_i(X^{(j-i)})\kappa_{j-i}, \tag{9}$$

where the random variables $X_l^{(j)}$ assume the values -1, 0 and 1 with probabilities $\nu/2$, $1-\nu$ and $\nu/2$. It would be easier to set all $N_j = 1/\alpha$, but the algorithm would become more expensive. If α is small, then we expect that the new value of ξ_j is close to the weighted average of the old ξ_{j-1}, ξ_j and ξ_{j+1} with weights $\nu/2$, $1-\nu$ and $\nu/2$ and that the variance of u_j is small. This is true if $\alpha = O(k^2)$. Equation (9) can be interpreted differently. We let the particle at x_j jump N_j times, and share the mass $h\xi_j$ between x_{j-1}, x_j and x_{j+1} according to the number of hits. Note that, because N_j depends on the solution and because f is a nonlinear function, this method is not equivalent to using (4), (5), (6) and (7) many times and then taking the average.

2. Main result. In this section we will state and discuss our main theorem. We consider the algorithm (4), (5), (8), (9) and (7) with initial data $u_j^0 = g(x_j - h/2)$. Since u_j^n is a random variable we express the convergence of the method in terms of the expected value of u_j^n and the variance of u_j^n. We have

THEOREM. *Assume that f and g are four times continuously differentiable functions, and that $f(0) = f(1) = 0$, $g(x) = 1$ for $x \leqq a$, $g(x) = 0$ for $x \geqq b$ and that $g'(x) \leqq 0$. Let $\nu \leqq 1$, $\alpha = k^2$ and $k/h^2 = \frac{1}{2}$. Let $M = \max_{[0,1]}(1, |f|, |f'|, |f''|)$ and assume that h is less than $(2/M)^{1/2}$, $((b-a)/2)^{1/3}$ and $(1+T^2)^{-1/4}$. If $nk = t \leqq T$, then*

$$|E(u_j^n) - u(x_j - h/2, t)| \leqq (e^{Mt} - 1)[C_0\nu h + C_1 k + C_2 \nu h^2],$$

$$\operatorname{var}(u_j^n) \leqq C_0 \nu h,$$

where $C_0 = 2 \max |g'| \exp(17MT)$. The constants C_1 and C_2 are independent of ν, but depend upon T and the derivatives of f and g. Let $N = \sum_j N_j$ be the total number of particles. Then

$$\frac{1}{k^2} \leqq N \leqq \frac{4}{k^2}.$$

Remark. If f is differentiable in [0, 1] and g is a bounded uniformly continuous function, then the reaction-diffusion equation has a classical solution ; see John [13]. However, this solution is not particularly smooth, and it is difficult to estimate the truncation error for the difference scheme

$$\frac{u_j^{n+1} - u_j^n}{k} = \nu \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2} + \frac{\nu}{2} f(u_{j+1}) + (1-\nu) f(u_j) + \frac{\nu}{2} f(u_{j-1}). \tag{10}$$

On the other hand, if the assumptions in the theorem are satisfied then we can use (10) to prove the existence of a smooth solution to the differential equation; for the approach see John [14, p. 185]. This solution will satisfy (10) except for a truncation error of order $O(k) + O(\nu h^2)$. This is almost as good as saying that u_{tt} and u_{xxxx} exist and are uniformly bounded.

That g should be constant at $\pm\infty$ is not essential. It simply means that at each time step we have only a finite number of particles. This condition has to be satisfied in all practical calculations. The requirements at $\pm\infty$ can be replaced by the assumption that $0 \leqq g \leqq 1$ and that the four derivatives are uniformly bounded. The assumption on g' is imposed for technical reasons and it is not natural. If it is not satisfied then the algorithm may have unphysical solutions and it becomes difficult to give a priori bounds for the computed solution; see Fig. 3. This phenomenon has not been observed in grid-free calculations because a particle with negative mass has only a small probability of jumping ahead of the wave. But the possibility cannot be excluded. Lin [16] has shown that, whether or not $g(x)$ is monotone, the solution $u(x, t)$ converges at an exponential rate toward a traveling wave solution which is monotone. In the proof Lin assumes that

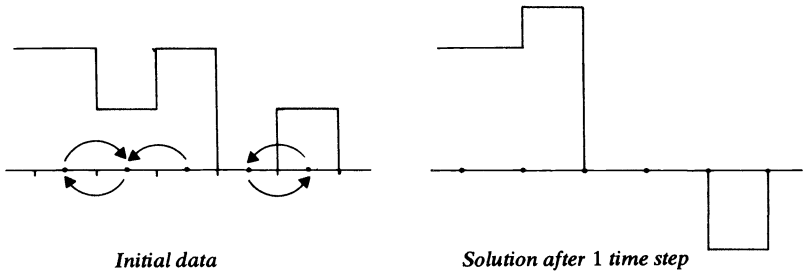


FIG. 3

$f(u) \leqq 0$ for $0 < u \leqq u_i$ and $f(u) > 0$ for $u_i < u < 1$, that $\int_0^1 f > 0$ and that $f'(1) < 0$. These requirements are satisfied for the Hodgkin–Huxley equation and for reaction-diffusion equations with positive ignition temperature. Thus our assumption on g' is not altogether artificial.

If $w(x+ct)$ is a traveling wave solution of (1) with $\nu = 1$, then $w(x/\sqrt{\nu}+ct)$ satisfy (1) with viscosity ν. Thus the gradient of the traveling wave is proportional to $1/\sqrt{\nu}$. It is therefore surprising that in (10) we can estimate the truncation error in the spacial direction by νh^2. The reason is that, although the solution $u(x, t)$ tends toward the traveling wave at exponential rate, the gradient of u can at most increase exponentially provided $g(x)$ is smooth. Thus as ν tends to zero it takes longer and longer time before the traveling wave is well approximated. If $g(x) = 1$ for $x < 0$ and 0 for $x > 0$, then we have the opposite conclusion, but this case is not covered by our theory.

We observe that the error in the solution u is proportional to the variance of the random walk and not to the standard deviation. This result is better than Chorin's conjecture for random solutions of Navier–Stokes equations with small viscosity; see [2]. However, the conjecture may still be right for Navier–Stokes equations, because the displacement of a particle in one dimension has a different effect on the solution than the displacement of a vortex blob in two dimensions.

If the mesh length is sufficiently small, then the number of particles is bounded independent of time. This is interesting because it is not obvious that (8) limits the creation of new particles. We conclude that the error due to the random method is proportional to the viscosity and inversely proportional to the fourth root of the number of particles. The dependence on the viscosity is presumably right, but is likely that a more sophisticated combinatoric analysis would give a dependence on the number of particles, which is comparable to the estimates for the heat equation. Anyway, the claim made by Milinazzo and Saffman [18], that more and more particles are needed as the viscosity tends to zero, is not true for our case.

3. The proof. The proof consists of three parts. We will first show that the variance of the computed solution tends to zero as the mesh length tends to zero, and that the expected value of the temperature gradient is bounded. Actually, the temperature gradient may become as large as $1/h$ in any calculation, but this is very unlikely to happen. The second part of the proof uses the standard arguments for the heat equation, with small modifications. Finally we show that the total number of particles is bounded.

We observe first that the computed solution is uniformly bounded and monotonically decreasing. At least, this is satisfied initially. Assume that $0 \leqq u_j^n \leqq 1$ and $u_j \geqq u_{j+1}$, that $u_j = 1$ for $jh \leqq a - nh$ and that $u_j = 0$ for $jh \geqq b + nh$. Since $kM \leqq 1$, we can

estimate the solution $v = v_j^n$ of (4) by

$$v = u + k[f(u) - f(0)] \geqq u + kf'(\theta u)u \geqq 0,$$

$$1 - v = 1 - u + k[f(1) - f(u)] \geqq [1 + kf'(u + \theta(1-u))](1-u) \geqq 0,$$

where $|\theta| \leqq 1$. The same argument shows that v_j is monotone. Indeed,

$$v_j - v_{j+1} = u_j - u_{j+1} + k[f(u_j) - f(u_{j+1})] \geqq (1 - kM)(u_j - u_{j+1}) \geqq 0.$$

This implies that all ξ_j in (5) are nonnegative and that $\sum h\xi_j = 1$. Here we have used the fact that $v_j = u_j$ for $jh \leqq a - nh$ and $jh \geqq b + nh$. Since the random walk is just a redistribution of the masses, it follows from (7) that u_j^{n+1} is monotone and $0 \leqq u_j^{n+1} \leqq 1$, and we see that $u_j^{n+1} = 1$ for $jh \leqq a - (n+1)h$ and $u_j^{n+1} = 0$ for $jh \geqq b + (n+1)h$. This completes the induction step.

We will now study the relationship between $u = u_j^{n+1}$ and $v = v_j^n$. By combining (5), (7) and (9) we find that

$$u_j = \frac{\nu}{2} v_{j+1} + (1-\nu)v_j + \frac{\nu}{2} v_{j-1} + C_{j-1} + D_j, \tag{11}$$

where

$$C_j = \sum_{l=1}^{N_j} \frac{1}{N_j}[I_1(X_l^{(j)}) - E(I_1(X_l^{(j)}))](v_j - v_{j+1}), \tag{12}$$

$$D_j = \sum_{i=0}^{1} \sum_{l=1}^{N_j} \frac{1}{N_j}[I_i(X_l^{(j)}) - E(I_i(X_l^{(j)}))](v_j - v_{j+1}).$$

Note that the expected value of $I_1(X)$ is $\nu/2$ and the expected value of $I_0(X) + I_1(X)$ is $1 - \nu/2$. We will show that $E(C_j) = E(D_j) = 0$, and that var (C_j) and var (D_j) are less than $\alpha\nu/2E(v_j - v_{j+1})$. Thus we conclude from (11) that

$$E(u_j) = \frac{\nu}{2} E(v_{j+1}) + (1-\nu)E(v_j) + \frac{\nu}{2} E(v_{j-1}). \tag{13}$$

It is more complicated to estimate the variance of u_j. Let $u_j = V + C + D$. Then it follows from (11) that

$$\text{var}\,(u_j) = \text{var}\,(V) + 2\,\text{cov}\,(V, C) + 2\,\text{cov}\,(V, D) + \text{var}\,(C) + \text{var}\,(D) + 2\,\text{cov}\,(C, D);$$

see [9, p. 216]. Since $\text{cov}\,(V, C) \leqq \sqrt{\text{var}\,(V)} \cdot \sqrt{\text{var}\,(C)}$ we see that

$$2\,\text{cov}\,(V, C) \leqq 2\sqrt{\text{var}\,(V)}\sqrt{\frac{\alpha\nu}{2} E(v_{j-1} - v_j)}$$

$$\leqq \frac{\sqrt{\alpha}}{2}(\text{var}\,(V) + 2\nu E(v_{j-1} - v_j)).$$

By using this result and the fact that 2 cov (C, D) is less than var (C) + var (D), we arrive at

$$\text{var}\,(u_j) \leqq (1 + \sqrt{\alpha})\,\text{var}\,(V) + (\sqrt{\alpha}\nu + \alpha\nu)E(v_{j-1} - v_{j+1})$$

$$\leqq (1+k)\left[\frac{\nu}{2}\,\text{var}\,(v_{j+1}) + (1-\nu)\,\text{var}\,(v_j) + \frac{\nu}{2}\,\text{var}\,(v_{j-1})\right] + 2k\nu E(v_{j-1} - v_{j+1}). \tag{14}$$

We have now expressed the expected value and the variance at time step $n+1$ in terms of quantities at the previous half step. Before going on we will verify our claims for C_j and D_j. Let $Y = v_j - v_{j+1}$ and $Z_l = I_1(X_l^{(j)}) - E(I_1(X_l^{(j)}))$ and drop all indices j. The integer N depends upon the random variable Y. Let Y assume the values $y_1, \cdots, y_m$ with probabilities $p_1, \cdots, p_m$, and let $n_1, \cdots, n_m$ be the corresponding values of N. We can then rewrite (12) as

$$C = \sum_{i=1}^{m} \frac{1}{n_i}(Z_1 + \cdots + Z_{n_i}) y_i I_{y_i}(Y).$$

Here $I_y(Y)$ is the indicator function for the set $\{y\}$. It is 1 if $Y = y$ and 0 otherwise. This technique is called conditioning with respect to Y. Let $B_i = Z_1 + \cdots + Z_{n_i}$. Since $E(B_i I_{y_i}(Y))$ is equal to $E(B_i)p_i$ and $E(Z_l) = 0$, we see that

$$E(C) = \sum_{i=1}^{m} \frac{y_i}{n_i} \sum_{l=1}^{n_i} E(Z_l)p_i = 0.$$

To estimate var (C) we observe that the functions I_{y_i} are mutually exclusive. Thus cov $(B_i I_{y_i}, B_j I_{y_j})$ vanishes for all $i \neq j$. Since the Z_l are independent and var $(Z_l) = (\nu/2)(1-\nu/2)$, we conclude that

$$\begin{aligned}
\text{var } (C) &= \sum_{i=1}^{m} \left(\frac{y_i}{n_i}\right)^2 \sum_{l=1}^{n_i} \text{var } (Z_l)p_i \\
&= \sum_{i=1}^{m} \frac{y_i^2}{n_i^2} \cdot n_i \cdot \frac{\nu}{2}\left(1-\frac{\nu}{2}\right) p_i \\
&\leqq \frac{\alpha\nu}{2} \sum_{i=1}^{m} y_i p_i \\
&= \frac{\alpha\nu}{2} E(v_j - v_{j+1}).
\end{aligned}$$

Here the inequality $y_i/n_i \leqq \alpha$ is a consequence of (8). The proof for D_j is similar. The rest of our proof is based on (4), (13) and (14). Note that (13) and (14) depend only on the expected value and variance of v_j^n, and that so far we have not used the nonlinearity of f. Let $v = v_j^n$ and $u = u_j^n$ and let us say that the random variable u assumes the values $u_1, \cdots, u_m$ with probabilities $p_1, \cdots, p_m$. Then it follows from (4) that

$$E(v) = E(u) + kE(f(u)), \tag{15}$$

$$\text{var } (v) = \text{var } (u) + 2k \text{ cov } (u, f(u)) + k^2 \text{ var } (f(u)). \tag{16}$$

Let $\mu = E(u) = \sum u_r p_r$. By using Taylor's formula we see that

$$\begin{aligned}
E(f(u)) &= \sum_r \left[f(\mu) + f'(\mu)(u_r - \mu) + \frac{1}{2}f''(\mu + \theta_r(u_r - \mu))(u_r - \mu)^2\right] p_r \\
&= f(E(u)) + \theta\frac{M}{2} \text{var } (u),
\end{aligned} \tag{17}$$

$$\begin{aligned}
\text{cov } (u, f(u)) &= \sum_r (u_r - \mu)[f(\mu) + f'(\mu + \theta_r(u_r - \mu))(u_r - \mu)]p_r \\
&\leqq M \text{ var } (u).
\end{aligned}$$

Since var $(f(u)) = E(f^2(u)) - [E(f(u))]^2$, we find from (17) that

$$\text{var}\,(f(u)) = f^2(E(u)) + \theta\frac{4M^2}{2}\,\text{var}\,(u) - (f(E(u)) + \theta\frac{M}{2}\,\text{var}\,(u))^2$$

$$\leqq 3M^2\,\text{var}\,(u),$$

where we have used the fact that $(f^2)'' \leqq 4M^2$. By combining these results with (15) and (16) and using the fact that $kM \leqq 1$, we arrive at

$$E(v) = E(u) + kf(E(u)) + \theta\frac{kM}{2}\,\text{var}\,(u),$$

$$\text{var}\,(v) \leqq (1 + 5kM)\,\text{var}(u).$$

We can now connect the expected value and variance of u at time step $n+1$ with the same quantities at the previous time step. By inserting the last two estimates in (13) and (14) and using Taylor's formula we conclude that

$$\begin{aligned} E(u_j^{n+1}) &= \frac{\nu}{2}E(u_{j+1}^n) + (1-\nu)E(u_j^n) + \frac{\nu}{2}E(u_{j-1}^n) \\ &\quad + k\left[\frac{\nu}{2}f(E(u_{j+1}^n)) + (1-\nu)f(E(u_j^n)) + \frac{\nu}{2}f(E(u_{j-1}^n))\right] \\ &\quad + \theta\frac{kM}{2}\left[\frac{\nu}{2}\,\text{var}\,(u_{j+1}^n) + (1-\nu)\,\text{var}\,(u_j^n) + \frac{\nu}{2}\,\text{var}\,(u_{j-1}^n)\right], \end{aligned} \tag{18}$$

$$\begin{aligned} \text{var}\,(u_j^{n+1}) &\leqq (1 + 13kM)\left[\frac{\nu}{2}\text{var}\,(u_{j+1}^n) + (1-\nu)\,\text{var}\,(u_j^n) + \frac{\nu}{2}\,\text{var}\,(u_{j-1}^n)\right] \\ &\quad + 4k\nu E(u_{j-1}^n - u_{j+1}^n). \end{aligned}$$

To prove that the variances tend to zero, the last inequality must be combined with an estimate for $E(u_{j-1} - u_{j+1})$. By using (18) we get the required estimate,

$$\begin{aligned} E(u_{j-1}^{n+1} - u_{j+1}^{n+1}) &\leqq (1 + kM)\left[\frac{\nu}{2}E(u_{j-2} - u_j) + (1-\nu)E(u_{j-1} - u_{j+1}) + \frac{\nu}{2}E(u_j - u_{j+2})\right] \\ &\quad + \theta\frac{kM}{2}\left[\frac{\nu}{2}\,\text{var}\,(u_{j-2}) + (1-\nu)\,\text{var}\,(u_{j-1}) + \nu\,\text{var}\,(u_j)\right. \\ &\quad \left. + (1-\nu)\,\text{var}\,(u_{j+1}) + \frac{\nu}{2}\,\text{var}\,(u_{j+2})\right]. \end{aligned}$$

Let $\varepsilon^n = \max_j E(u_{j-1}^n - u_{j+1}^n)$ and $\delta^n = \max_j \text{var}\,(u_j^n)$. From the last two inequalities we conclude that

$$\nu\varepsilon^{n+1} + \delta^{n+1} \leqq (1 + 17Mk)(\nu\varepsilon^n + \delta^n).$$

By using this estimate over and over we find that $\nu\varepsilon^n + \delta^n$ is less than $\exp(17Mkn) \cdot (\nu\varepsilon^0 + \delta^0)$. Since $\delta^0 = 0$ and ε^0 is less than $2h \max|g'|$, we have shown that the variance tends to zero as long as $nk \leqq T$. This completes the first part of the proof.

In the second part we will compare the expected value of the computed solution with the exact solution. Let $\phi_j^n = E(u_j^n)$ and let $C_0 = 2 \max|g'| \exp(17MT)$. We can

then rewrite (18) as

$$\phi^{n+1} = \frac{\nu}{2}\phi_{j+1} + (1-\nu)\phi_j + \frac{\nu}{2}\phi_{j-1} + k\left[\frac{\nu}{2}f(\phi_{j+1}) + (1-\nu)f(\phi_j) + \frac{\nu}{2}\ f(\phi_{j+1})\right] + \theta\frac{kM}{2}C_0\nu h, \tag{19}$$

where $|\theta| \leqq 1$. This shows that the expected value of the computed solution satisfies a finite difference approximation to the reaction-diffusion equation, except for a "truncation error" of order $O(\nu h)$. Let $\psi_j^n = u(x_j - h/2, nk)$, where $u(x, t)$ is the solution of the differential equation. Since f and g are four times continuously differentiable, we know that u, u_t and u_{xxx} are uniformly Lipschitz continuous. Thus if $k/h^2 = \frac{1}{2}$ then ψ satisfies (10) except for a truncation error, which can be bounded by $C_1 k + C_2 \nu h$, where C_1 and C_2 depend on T and the first four derivatives of f and g but are independent of ν. This implies that

$$\psi^{n+1} = \frac{\nu}{2}\psi_{j+1} + (1-\nu)\psi_j + \frac{\nu}{2}\psi_{j-1} + k\left[\frac{\nu}{2}f(\psi_{j+1}) + (1-\nu)f(\psi_j) + \frac{\nu}{2}f(\psi_{j-1})\right] + \theta[C_1 k + C_2 \nu h^2]. \tag{20}$$

The rest is straightforward. Let e_j^n be the error $\phi_j^n - \psi_j^n$, and set $\varepsilon^n = \max_j |e_j^n|$. By combining (19) and (20) and using the mean value theorem we see that

$$\varepsilon^{n+1} \leqq (1 + kM)\varepsilon^n + kM(C_0 \nu h + C_1 k + C_2 \nu h^2).$$

Since $\varepsilon^0 = 0$, the result stated in the theorem follows by induction. This completes the second part of the proof.

Finally we will show that the total number of particles is bounded. We observe that after the nth time step ξ is different from zero in at most $(b-a)/h + 2 + 2n$ mesh points. Thus it follows from (8) that

$$N - \left(\frac{b-a}{h} + 2 + 2n\right) < \frac{\sum h\xi_j}{\alpha} \leqq N. \tag{21}$$

Since $\sum h\xi_j = 1$ and $\alpha = k^2$, we get the lower bound on N and see that $1/k \leqq \sqrt{N}$. By using this result and $\alpha = h^4/4$ we conclude from (21) that

$$N - 2T\sqrt{N} \leqq \frac{b-a}{h} + 2 + \frac{4}{h^4}.$$

By solving this inequality as a quadratic equation and using our initial assumptions for h we arrive at the upper bound given in the theorem. This completes the proof.

4. Open problems. The technique presented in this paper cannot be applied to random walk solutions of Navier–Stokes equations without serious modifications. First, we work on a grid, while the random solutions of Navier–Stokes equations are grid-free. Secondly, we use many particles per mesh point, while every vortex blob is represented by one particle. Finally, we estimate the variance at a time step in terms of the variance of the previous time step. It is not clear whether this is possible for random walk solutions of Navier–Stokes equations. From an abstract point of view, the problem is to make sense of Trotter's product formula for nonlinear equations which are solved by a combination of deterministic and random methods.

On the positive side, our technique can be extended to systems of differential equations with different time constants. While the monotonicity assumption does hold

for traveling wave solutions for some reaction-diffusion equations (see [16]), it is not true generally. The solutions of the heat equation can be replaced by random methods which include more mesh points, and higher order methods can be used to solve the system of ordinary differential equations. Thus we should be able to take larger time steps.

Acknowledgments. The author thanks Alexandre J. Chorin, F. Alberto Grünbaum and Andrew Majda for helpful discussions.

REFERENCES

[1] M. D. BRAMSON, *Maximal displacement of branching Brownian motion*, Comm. Pure Appl. Math., 31 (1978), pp. 531–581.

[2] A. J. CHORIN, *Numerical study of slightly viscous flow*, J. Fluid Mech., 57 (1973), pp. 785–796.

[3] ———, T. J. R. HUGHES, M. F. MCCRACKEN AND J. E. MARSDEN, *Product formulas and numerical algorithms*, Comm. Pure Appl. Math., 31 (1978), pp. 205–256.

[4] ———, *Vortex sheet approximation of boundary layers*, J. Comp. Phys., 27 (1978), pp. 428–442.

[5] ———, *Vortex models and boundary layer instability*, this Journal, 1 (1980), pp. 1–21.

[6] ———, *Numerical methods for use in combustion modeling*, Proc. Int. Conf. Num. Meth. in Science and Engineering, Versailles, 1979.

[7] V. DEL PRETE, *Numerical Study of Vortex Breakdown*, Report LBL-8503, Lawrence Berkeley Laboratory, Berkeley, CA, 1978.

[8] H. A. DWYER, R. J. KEE AND B. R. SANDERS, *An adaptive grid method for problems in fluid mechanics and heat transfer*, AIAA Computational Fluid Dynamics Conference, Williamsburg, VA, 1979.

[9] W. FELLER, *An Introduction to Probability Theory and Its Applications*, vol. 1, 2nd ed., John Wiley, New York, 1957.

[10] A. FRIEDMAN, *Partial Differential Equations of Parabolic Type*, Prentice-Hall, Englewood Cliffs, NJ, 1964.

[11] O. H. HALD AND V. MAUCERI DEL PRETE, *Convergence of vortex methods for Euler's equations*, Math. Comp., 32 (1978), pp. 791–809.

[12] O. H. HALD, *Convergence of vortex methods for Euler's equations*, II, SIAM J. Numer. Anal., 16 (1979), pp. 726–755.

[13] F. JOHN, *On the integration of parabolic equations by difference methods*, Comm. Pure Appl. Math., 5 (1952), pp. 155–211.

[14] ———, *Partial Differential Equations*, 3rd ed., Springer-Verlag, New York, 1978.

[15] A. KOLMOGOROV, I. PETROVSKII AND N. PISKUNOV, *Étude de l'equation de la diffusion avec croissance de la quantité de la matière et son application a un problème biologique*, Moscow University Bull. Math., 1 (1937), pp. 1–25.

[16] S. S. LIN, *Theoretical Study of a Reaction-Diffusion Model for Flame Propagation in a Gas*, Ph.D. thesis, University of California, Berkeley, CA, 1979.

[17] H. P. MCKEAN, *Application of Brownian motion to the equation of Kolmogorov–Petrovskii–Piskunov*, Comm. Pure Appl. Math., 28 (1975), pp. 323–331.

[18] F. MILINAZZO AND P. G. SAFFMAN, *The calculation of large Reynolds number two-dimensional flow using discrete vortices with random walk*, J. Comp. Phys., 23 (1977), pp. 380–392.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0008 $01.00/0

A STUDY OF NUMERICAL METHODS FOR REACTION-DIFFUSION EQUATIONS*

ROLF D. REITZ†

Abstract. Numerical methods are applied to one-dimensional unsteady reaction-diffusion equations to seek traveling wave solutions. These equations describe flame propagation in certain combustion systems. Model scalar reaction-diffusion equations which admit traveling waves as exact solutions are formulated, and one of these is solved with twelve different numerical integration schemes for a test problem. The diffusion terms are differenced using the explicit method introduced by Saul'yev [*Integration of Equations of Parabolic Type by the Method of Nets*, Pergamon, New York, 1964], which, it is shown, can be formally accurate to $O(\Delta t^3/\Delta x^2)$. Both implicit and explicit techniques for the reaction terms are tested. The results of the study with these schemes indicates that numerical diffusion can reduce accuracy significantly, that numerical dispersion truncation errors can reduce accuracy if they are sufficiently large, and that an accurate representation of the reaction terms in the difference equations is important to retain overall accuracy. In addition to the above test problem, results are given using one of the explicit methods for the computation of a propagating ozone decomposition flame. The results show good agreement with the fourth-order accurate results of Margolis [J. Comput. Phys., 27 (1978), pp. 410–427] which indicates that more efficient lower order numerical methods can be sufficiently accurate for practical computations.

Key words. reaction-diffusion equations, numerical methods, combustion

Introduction. Traveling wave solutions to nonlinear reaction-diffusion equations are of interest in biological applications and in the theory of flame propagation. In combustion applications recent studies have focused on mathematical models which incorporate detailed chemical kinetic mechanisms, e.g., Westbrook et al. [16]. This trend has increased the number of equations which need to be solved and has stimulated research in numerical methods for reacting gases.

In this paper we present a study of several numerical methods as applied to the solution of one-dimensional reaction-diffusion equations. The twelve numerical schemes investigated in the study use variations of the method introduced by Saul'yev [14, p. 52] for the diffusion terms. These methods have the advantage of being explicit and can be extended for use in multidimensional computations. Moreover, certain of the schemes are unconditionally stable. The various schemes studied also include implicit and explicit methods for the evaluation of the reaction terms. Variation of parameters in the methods presented here allows the effect of different truncation errors on accuracy to be explored.

The numerical schemes are first applied to the solution of a scalar model equation which admits exact traveling wave solutions, and the study permits comparisons to be made between exact and computed solutions. The results provide insight into the performance of numerical methods for more complicated systems of equations. Results are next given using one of the explicit numerical schemes for the computation of a coupled system of reaction-diffusion equations for the propagating ozone decomposition flame. The results are compared with those of other authors [2], [8], [9] for this problem.

Model formulation. We consider propagating wave solutions to one-dimensional reaction-diffusion equations of the form

$$\frac{\partial u^{(k)}}{\partial t} = \frac{\partial}{\partial x}\left(d^{(k)}\frac{\partial u^{(k)}}{\partial x}\right) + F^{(k)}(u^{(1)}, \cdots, u^{(K)}),$$

* Received by the editors August 11, 1980. This work was supported by the U.S. Department of Energy under contract EY-76-C-02-3077.

† Department of Mechanical and Aerospace Engineering, Princeton University, Princeton, New Jersey 08544.

subject to the boundary conditions

$$\frac{\partial u^{(k)}}{\partial x}(\infty, t) = \frac{\partial u^{(k)}}{\partial x}(-\infty, t) = 0, \qquad k = 1, \cdots, K. \tag{1}$$

These equations model one-dimensional laminar flame propagation in an initially stagnant premixed mixture of reactants if constant pressure combustion is assumed, with Lewis number = 1, and if a Lagrangian coordinate transformation is introduced. This coordinate transformation eliminates the convective terms in the governing conservation equations and leaves the steady flame speed unchanged [11].

In formulating model equations we consider the scalar case $K = 1$ and $d^{(1)} = d =$ constant. In combustion applications $u = u^{(1)}$ could be taken to represent nondimensional temperature such that $0 \leqq u \leqq \alpha$, where α is the adiabatic flame temperature. This model represents a global irreversible one-step chemical kinetic mechanism with a stoichiometrically premixed mixture of reactants. In this case $F(u)(>0)$ is usually modeled with the nonlinear Arrhenius expression [17, p. 95]. Spalding [15] proposed a simpler algebraic expression,

$$F(u) = \beta u^n (\alpha - u)^m, \tag{2}$$

which approximates the Arrhenius function for large positive n, but no exact solutions have previously been found for this model.

It is noted here that exact traveling wave solutions can be found for $n = 2$ and $m = 1$ or, more generally, for F given by

$$F(u) = \beta u^{m+1}(\alpha - u^m), \qquad m > 0. \tag{3}$$

The solutions are

$$u(x - St) = [\alpha/(1 + e^{mS(x-St)/d})]^{1/m}, \tag{4}$$

and they correspond to a class of stable minimal wave speed solutions, with exponential decay for $x \to \infty$, which are discussed, for example, by Lin [7]. In (4) the wave speed is

$$S = \alpha(\beta d/m + 1)^{1/2}.$$

An additional model equation is $n = m = 1$ in (2). This problem (Fisher's equation) has been analyzed theoretically by Kolmogorov et al. [6] and numerically by Gazdag and Canosa [4] in studies which show the existence of a minimum wave speed. McKean [10] has shown that the final steady wave speed is determined by the nature of the *initial* data for this equation. In particular, for initial data which decay exponentially ahead of the wave, i.e.,

$$u(x, 0) = a\, e^{-bx}, \qquad x \to \infty, \tag{5a}$$

where a and b are constants, the asymptotic steady wave speed is

$$S = b + \frac{1}{b}. \tag{5b}$$

Ablowitz and Zeppetella [1] have found, for the particular wave speed $S = 5/\sqrt{6}$, that an exact traveling wave solution of the Fisher equation is given by (here $\alpha = \beta = d = 1$)

$$u\left(x - \frac{5}{\sqrt{6}}t\right) = (1 + e^{(x - 5/\sqrt{6}\,t)/\sqrt{6}})^{-2}. \tag{6}$$

Numerical method. We describe a numerical method for reaction-diffusion equations which will be shown to be accurate to $O(\Delta t^3/\Delta x^2)$ under certain conditions. The method is given in (7) below. Variations of the method, and in particular several implicit treatments of the reaction terms, are obtained by varying the parameters $\gamma, \delta, \alpha, q, \beta$ in the equations. The diffusion terms in (1) are differenced using the explicit method first introduced by Saul'yev [14, p. 52]. He was only able to show that the numerical method was accurate to $O(\Delta t^2/\Delta x^2)$. The finite difference approximation to the continuum solution $U^n_{k,j} \cong u^{(k)}(x_j, n\,\Delta t)$ is computed at grid points x_j from the set of equations

$$\begin{aligned}\frac{V_j - U^n_{k,j}}{\Delta t} = \frac{1}{2\Delta x^2}\{&(d^n_{k,j+1} + d^n_{k,j})(U^n_{k,j+1} - U^n_{k,j}) \\ &-[\gamma(\overline{d^{n+1}_{k,j}} + \overline{d^{n+1}_{k,j-1}})(V_j - V_{j-1}) \\ &\quad + (1-\gamma)(d^n_{k,j} + d^n_{k,j-1})(U^n_{k,j} - U^n_{k,j-1})]\} \\ &+ \delta F_k(V_j) + \alpha F_k(U^n_{l,j+q\Delta x}) + \beta F_k(U^{n-1}_{l,j}),\end{aligned} \tag{7a}$$

$$\begin{aligned}\frac{W_i - U^n_{k,i}}{\Delta t} = \frac{1}{2\Delta x^2}\{&-(d^n_{k,i} + d^n_{k,i-1})(U^n_{k,i} - U^n_{k,i-1}) \\ &+[\gamma(\overline{d^{n+1}_{k,i+1}} + \overline{d^{n+1}_{k,i}})(W_{i+1} - W_i) \\ &\quad + (1-\gamma)(d^n_{k,i+1} + d^n_{k,i})(U^n_{k,i+1} - U^n_{k,i})]\} \\ &+ \delta F_k(W_i) + \alpha F_k(U^n_{l,i+q\Delta x}) + \beta F_k(U^{n-1}_{l,i}),\end{aligned} \tag{7b}$$

$$U^{n+1}_{k,j} = \frac{V_j + W_j}{2}, \tag{7c}$$

where

$$j = 1, 2, \cdots, M, \qquad i = M - j + 1,$$
$$k = 1, 2, \cdots, K, \qquad l = 1, 2, \cdots, K,$$

Δx is the mesh spacing and M is the number of computational points. In the present discussion we take

$$\delta = q = 0, \quad \alpha = \tfrac{3}{2}, \quad \beta = -\tfrac{1}{2}. \tag{7d}$$

In this case, (7a, b) consists of two predictor sweeps of the mesh (in opposite directions) followed by a corrector step in (7c), with the reaction terms being given by an explicit Adams–Bashforth extrapolation formula. Computationally, the operation count is similar to that of the back substitution steps in the LU decomposition of the matrices in certain implicit methods.

The difference equations (7) are still implicit in the diffusion coefficients $\overline{d^{n+1}_{k,j}}$. We will assume that these can be computed from quantities whose values are already known at time level $n+1$; in fact we will later assume that they are constants. In addition, equations (7) are also implicit in the boundary conditions, but this is not of concern due to the specification of Neumann boundary conditions as in (1) [13, p. 95].

The numerical method represented by (7) has leading terms, given by a formal truncation error analysis, which are similar to those of the second-order accurate Crank–Nicolson [3] method. (Notice that the Crank–Nicolson method is obtained by replacing V_j and W_j by $U^{n+1}_{k,j}$ in (7) with $\gamma = \delta = \alpha = \frac{1}{2}$ and $q = \beta = 0$.) Expanding terms in (7) in a Taylor series about the continuum solution $u^{(k)}(x_j, (n+1)\Delta t)$ shows that the

continuum analogue of the difference equations is

$$
\begin{aligned}
&u_t-(du_x)_x-F(u)\\
&\quad=\frac{\Delta t}{2}(u_t-(du_x)_x-F(u))_t-\frac{\Delta t}{2}(1-\gamma)(du_x)_{xt}-\frac{2d\gamma}{\Delta x}(V-W)_x\\
&\qquad+O(\gamma\Delta x(V-W)_{xxx},\Delta t^2,\Delta x^2,\Delta t\,\Delta x),
\end{aligned}
\tag{8}
$$

where the superscript (k) has been dropped for simplicity and the subscripts t and x indicate differentiation. The second and third terms on the right-hand side of (8) and the $O(\Delta x(V-W)_{xxx})$ term (which will be seen to be $O(\Delta t^2)$), represent a departure from the first-order Crank–Nicolson truncation error form. The term involving $(V-W)_x$ may be estimated by subtracting the Taylor expansion of (7b) from that of (7a), and a regular perturbation analysis with small parameter Δt shows that

$$
V-W=-\frac{2\Delta t^2\gamma}{\Delta x}(du_x)_t+\frac{\Delta t^3\gamma}{\Delta x}((du_x)_{tt}-\gamma(d(du_x)_{xt})_x)+\text{h.o.t.}
$$

Substituting this in (8) gives

$$
\begin{aligned}
&u_t-(du_x)_x-F(u)\\
&\quad=\frac{\Delta t}{2}(u_t-(du_x)_x-F(u))_t+\frac{\Delta t}{2}(2\gamma^2r-(1-\gamma))(du_x)_{xt}\\
&\qquad-\frac{\Delta t^2\gamma^2r}{2}((du_x)_{ttx}-\gamma(d(du_x)_{xt})_{xx})+\text{h.o.t.},
\end{aligned}
\tag{9}
$$

where $r=d\Delta t/\Delta x^2$. From this it is seen that if $d=$ const. and

$$
\gamma=\frac{\sqrt{1+8r}-1}{4r},
\tag{10}
$$

the scheme is accurate to $O(\Delta t^3/\Delta x^2)$. (The negative root leads to a more restrictive stability condition; see (11) below.) Formal second-order accuracy only occurs under the additional constraints $F(u)=0$ and $\gamma=1$, which in view of (10) implies that $r\approx 0$.

The condition for diffusional stability of the method in (7) is [14]

$$
\frac{1}{r}\geqq 2(1-\gamma),
\tag{11}
$$

and it is thus unconditionally stable for $\gamma\geqq 1$. However, the Lipschitz timestep constraint

$$
\Delta t\leqq\frac{\text{const}}{\max\limits_j\left|\dfrac{\partial F_k(U_{l,j})}{\partial U_{m,j}}\right|}
\tag{12}
$$

is present, and recently Hoff [5] has indicated that this may make an unconditionally stable scheme for the diffusion operator only conditionally stable for the combined reaction-diffusion system.

Discussion of results. We first present the results of a parameter study of the numerical method described in (7a, b, c). The parameters used are listed in Table 1. The various schemes include implicit and explicit methods for the evaluation of reaction

TABLE 1

Parameter study of numerical method (7a, b, c) (Imp, CN, Exp, *analogous implicit, Crank–Nicolson and explicit methods*).

Scheme	Diffusion		Reaction Term					Leading Truncation Error*
	γ	Class	δ	α	q	β	Class	
I	2	Imp	1	0	0	0	Imp	$\frac{\Delta t}{2}u_{tt}+4r\,\Delta t\phi-2r\,\Delta t^2L^{2,1}$
II	1	CN	1	0	0	0	Imp	$\frac{\Delta t}{2}(u_t-du_{xx})_t+r\Delta t\phi-\frac{r\Delta t^2}{2}L^{1,1}$
III	0	Exp	1	0	0	0	Imp	$\frac{\Delta t}{2}(u_t-2\,du_{xx})_t$
IV	2	Imp	0	1	0	0	Exp	$\frac{\Delta t}{2}(u_t-2F)_t+4r\,\Delta t\phi-2r\,\Delta t^2L^{2,0}$
V	1	CN	0	1	0	0	Exp	$\frac{\Delta t}{2}(u_t-du_{xx}-2F)_t+r\,\Delta t\phi-\frac{r\,\Delta t^2}{2}L^{1,0}$
VI	0	Exp	0	1	0	0	Exp	$\frac{\Delta t}{2}(u_t-2\,du_{xx}-2F)_t$
VII	2	Imp	$\frac{1}{2}$	$\frac{1}{2}$	0	0	CN	$\frac{\Delta t}{2}(u_t-F)_t+4r\,\Delta t\phi-2r\,\Delta t^2L^{2,1/2}$
VIIIa	1	CN	$\frac{1}{2}$	$\frac{1}{2}$	0	0	CN	$\frac{\Delta t}{2}G_t+r\,\Delta t\phi-\frac{r\,\Delta t^2}{2}L^{\gamma,\delta}$
VIIIb			0	$\frac{3}{2}$	0	$-\frac{1}{2}$	CN	
IX	0	Exp	$\frac{1}{2}$	$\frac{1}{2}$	0	0	CN	$\frac{\Delta t}{2}(u_t-2\,du_{xx}-F)_t$
X	1	CN	0	1	$-\frac{S\,\Delta t}{2\,\Delta x}$	0	Exp	$\frac{\Delta t}{2}G_t-\frac{\Delta t}{2}(F_t+SF_x)+r\Delta t\phi-\frac{r\Delta t^2}{2}L^{1,0}$
XI	0	Exp	0	1	0	0	Exp	$\Delta tG_t-\frac{\Delta t}{2}(u_{tt}-S^2u_{xx})$
XII	Eq. (10)	CN	$\frac{1}{2}$	$\frac{1}{2}$	0	0	CN	$\frac{\Delta t}{2}G_t-\frac{r\,\Delta t^2}{2}L^{\gamma,1/2}$

* $G(u)=(u_t-du_{xx}-F)$; $\phi(u)=du_{xxt}$; $L^{\gamma,\delta}(u)=d[(u_t-\gamma\,du_{xx})_{tx}+2\delta F_u u_{tx}]_x$.

terms. A single nonlinear reaction-diffusion model equation was used in these studies. We show later computational results for a coupled system of reaction-diffusion equations describing the propagating ozone decomposition flame.

The Fisher equation, chosen as the model equation in these studies, is written as

$$u_t = du_{xx} + u(1-u). \tag{13}$$

This yields traveling wave solutions with $0 \leqq u \leqq 1$. Notice that the quadratic nonlinear source term simplifies the computation of implicit methods ($\delta \neq 0$ in (7a, b)) since they can be solved explicitly.

We set $d = 1$ and used an equally spaced mesh with the domain $-50 \leqq x \leqq 400$. This domain was found to be sufficient to eliminate boundary effects. The initial condition

$U(x, 0)$ was given by the exact solution (6), which is a traveling wave with $S = 5/\sqrt{6}$ and $u(\infty, t) = 0$, $u(-\infty, t) = 1$.

In the results shown in Fig. 1, however, this initial condition was modified slightly in order to illustrate certain features of the model (13). This figure shows the numerical

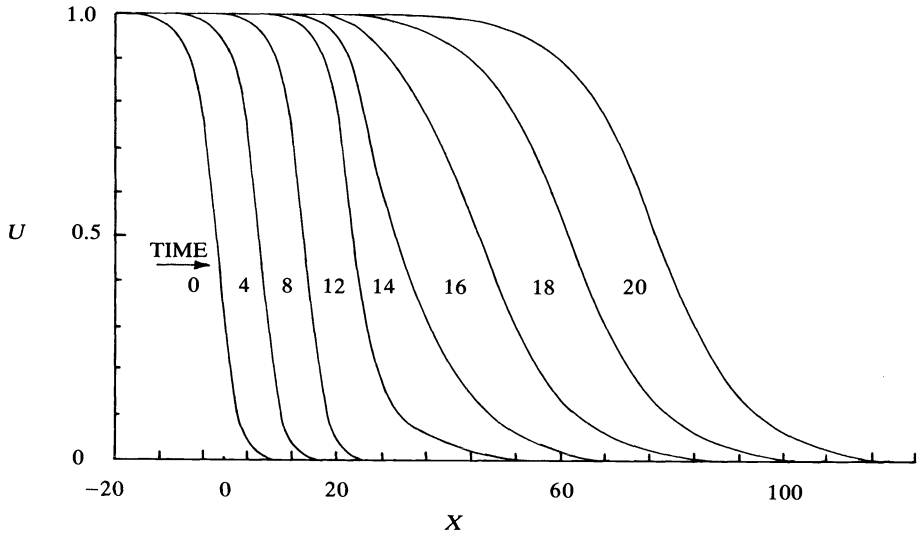


FIG. 1. *Numerical solution of Fisher's equation with initial data*: (6) *for* $u > 5\times10^{-6}$ *and* (5a) *for* $u \leqq 5\times10^{-6}$.

solution $U(x)$, for various times up to time $t = 20$, using the standard explicit scheme VI in Table 1 on a fine mesh with $\Delta x = 0.25$, $\Delta t = 0.01$. The initial data were given by (6) for $U > 5\times10^{-6}$, but for $U \leqq 5\times10^{-6}$ by (5a). The constant b in (5a) was set equal to 0.127, i.e., $b + 1/b \cong 8$, and the constant a was fixed by matching the initial data at the changeover point. Notice that with these conditions (5b) would predict a change in the wave speed from its initial value of $5/\sqrt{6}$ to a final speed $S \cong 8$.

The results shown in Fig. 1 indicate that the wave propagates to the right with its structure preserved up to $t \cong 8$. This is followed by a transition period for $10 \leqslant t \leqslant 16$ where the wave profile undergoes a change. Finally, for $t \gtrsim 18$ the wave appears to settle to a new structure. The corresponding change in the wave speed is shown in curve 1 of Fig. 2, which shows speed versus time for time up to $t = 30$. The wave speed was computed from the relation

$$S = \int_{-\infty}^{\infty} u(1-u)\,dx, \tag{14}$$

which is, however, only valid for steady wave propagation. The results in Fig. 2 confirm that a new steady state, as measured by changes in the wave speed, is reached with speed $\cong 8$. This agrees with (5b) for $b = 0.127$. Curve 2 in Fig. 2 summarizes a different set of results which were obtained under the same conditions as those given by curve 1, but where the initial condition was given only by (6). In this case the wave is seen to propagate with a speed which agrees well with the exact solution $S = 5/\sqrt{6}$ for the duration of the computation.

The sensitivity of the results to small perturbations in the initial data which was seen in Figs. 1 and 2, indicates that the Fisher equation could be an exacting test

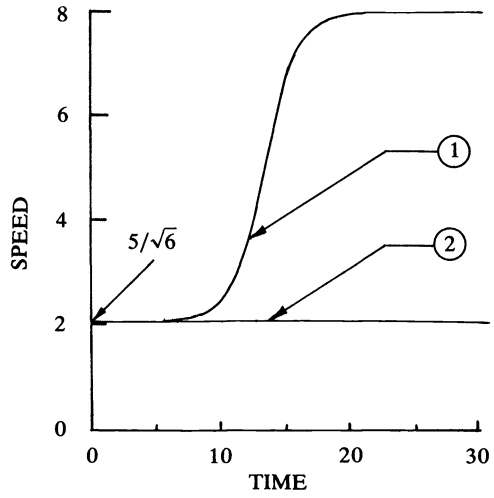


FIG. 2. *Computed wave speed versus time for Fisher's equation. Initial data curve* 1, (6) *and* (5a); *curve* 2, (6) *only.*

problem for numerical studies. Accordingly, the twelve numerical schemes which are listed in Table 1 were compared for this problem, as will now be discussed with the help of Figs. 3 and 4. These figures are plots of computed wave speed versus time with a coarse mesh ($\Delta x = 1$ and $\Delta t = 0.2$) and the initial data were specified with (6) only—a

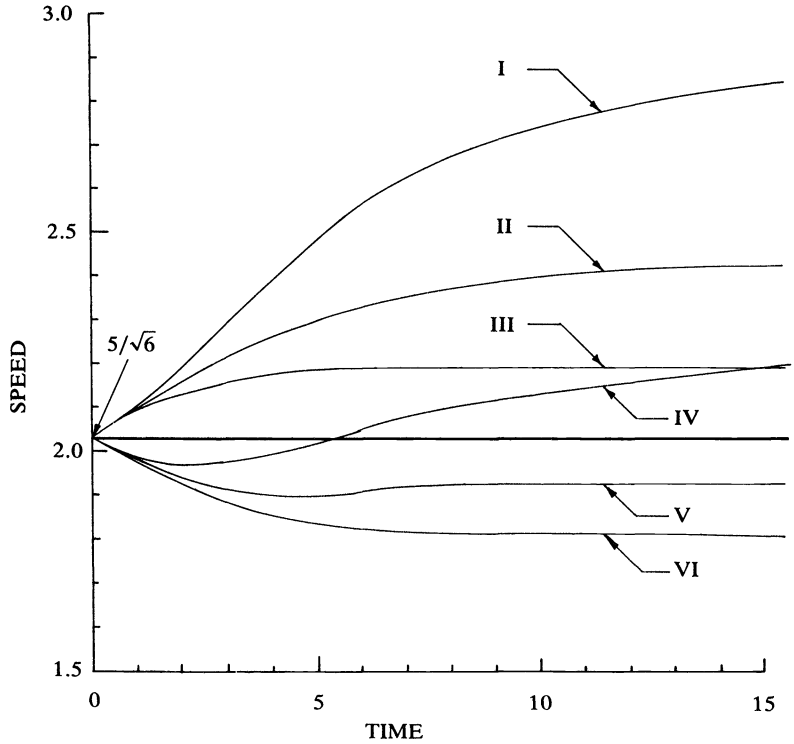


FIG. 3. *Computed wave speed versus time for Fisher's equation with exact solution* $S = 5/\sqrt{6}$. *Initial data,* (6). *Schemes* I–VI.

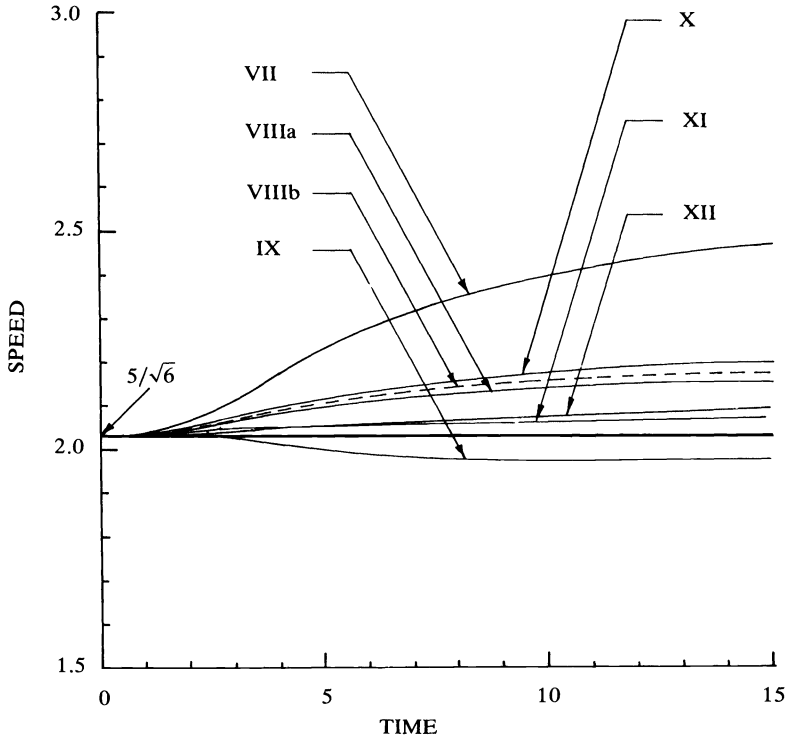


FIG. 4. *Computed wave speed versus time for Fisher's equation with exact solution* $S = 5/\sqrt{6}$. *Initial data,* (6). *Schemes* VII–XII.

steady propagating wave with speed $S = 5/\sqrt{6}$. For the schemes I to VI shown in Fig. 3 the error in the computed wave speed appears to grow, at least initially, linearly with time. The results of schemes VII to XII are shown in Fig. 4; error growth is more modest so that greater accuracy is associated with most of these schemes.

Parenthetically, notice that the schemes in Table 1 also resemble several other standard schemes: implicit (Imp), Crank–Nicolson (CN) and explicit (Exp). Each of these methods may be formed from (7a, b) by replacing V_j and W_j with $U_{k,j}^{n+1}$ in either the diffusion or reaction term or both terms, and by making the same choice of the parameters γ, δ, α, q and β as is shown in Table 1. The schemes in (7a, b, c) produce the same order Δt truncation errors as the corresponding Imp, CN and Exp schemes which are shown in the table. These error terms are listed first in the truncation error column of the table. However, equations (7) generate additional terms, proportional to $r\,\Delta t$ and $r\,\Delta t^2$, which are also given in the table. In the case of the standard explicit method (7a, b) become identical and reduce to the standard explicit scheme.

The underestimate of the steady wave speed which is seen in Fig. 3 for the standard explicit method VI ($\alpha = 1$, $\delta = \gamma = q = \beta = 0$) is primarily due to a first-order numerical diffusion error. This is demonstrated by comparing the results of this scheme with the results of scheme XI (Fig. 4), which is identical to scheme VI, except that the physical diffusion coefficient d was increased by a factor of $1+(S^2\Delta t/2d)$. (Here $S^{(n)}$ was computed explicitly using (14) as the computation proceeded.) This increase in d minimizes the numerical diffusion effect in steady wave propagation as is seen in the truncation error column of scheme XI in Table 1.

A penalty which is associated with the significant improvement in the accuracy of the explicit method which is seen in scheme XI is that the diffusional stability criterion $r \leqq \frac{1}{2}$ (compare (11) for $\gamma = 0$) becomes more restrictive for higher wave speeds. For $S \gg d/\Delta x$, the stability condition (11) is replaced by the C.F.L. condition $S\Delta t/\Delta x \leqq 1$. However, scheme XI is second-order accurate for the case of steady wave propagation, and it is therefore of interest to compare its performance with others in Table 1.

In scheme IX ($\delta = \alpha = \frac{1}{2}$, $\gamma = q = \beta = 0$) the leading truncation error is $-(\Delta t/2)du_{xxt}$. In contrast to scheme VI, this is a numerical dispersion error and not a diffusive error in steady propagation. A comparison of scheme IX (Fig. 4) with the results of scheme VI (Fig. 3) shows that this first-order dispersive error does not deteriorate accuracy as much as the diffusive error. In fact, the accuracy of the method IX is seen to be comparable to that of the second-order scheme XI.

The parameter γ in the diffusion terms of (7a, b) is given by (10) in scheme XII. Here $q = \beta = 0$ and $\delta = \alpha = \frac{1}{2}$. This optimal choice of the parameter γ results in a leading truncation error of the form $r\Delta t^2 L^{\gamma,1/2}(u)$. (The terms $L^{\gamma,\delta}(u)$ in Table 1 are derived from a Taylor series expansion of the methods followed by an analysis similar to that which led to (9).) In Fig. 4 it is seen that scheme XII performs as well as the second-order accurate scheme XI. From this it would be concluded that the $r\Delta t^2 L^{\gamma,\delta}(u)$ truncation error terms are small. The results which correspond to schemes VIII *a*, *b* and *X* are also shown in Fig. 4, and they are seen to give accurate results. The three schemes have the same leading truncation error. However, differences in their respective truncation errors do occur in the higher order terms (see Table 1), and the agreement in their results confirms that the effect of these $O(r\Delta t^2)$ terms is small.

The additional term $(\Delta t/2)(F_t + SF_x)$, which vanishes in steady propagation, in scheme X ($\gamma = \alpha = 1$, $\delta = \beta = 0$) comes from evaluating the reaction rate explicitly as $F(U^n_{j+q\Delta x})$ with $q = S\,\Delta t/2\,\Delta x$. (The reaction rate at $x_j + q\,\Delta x$ (here, $-1 < q < 0$) was determined by linear interpolation between the rates at x_j and x_{j-1} with $S^{(n)}$ given by (14).) This method of computing the reaction rate minimizes a truncation error, $-(\Delta t/2)F_t$, which arises from the similar but slightly less accurate scheme V ($\gamma = \alpha = 1$, $\delta = q = \beta = 0$). This may be seen by comparing scheme V in Fig. 3 with scheme X in Fig. 4. The reaction rate is evaluated implicitly in scheme VIIIa ($\gamma = 1$, $\delta = \alpha = \frac{1}{2}$, $q = \beta = 0$) and is computed explicitly, but with the same leading truncation error, in scheme VIIIb ($\gamma = 1$, $\delta = q = 0$, $\alpha = \frac{3}{2}$, $\beta = -\frac{1}{2}$. However, the methods VIIIa and VIIIb (dashed curve) are seen to produce similar results in Fig. 4. The methods VIIIa, b and X have a first-order numerical dispersion error $r\Delta t u_{xxt}$. A comparison of their results with those of the previously discussed scheme IX, which has a truncation error $-(\Delta t/2)du_{xxt}$, shows a corresponding change of sign in the error in the computed wave speed.

The first-order dispersive error is larger for scheme VII, and is $(\frac{1}{2}+4r)d\ \Delta t u_{xxt}$ ($\gamma = 2$, $q = \beta = 0$ and $\delta = \alpha = \frac{1}{2}$). With reference to Fig. 4, this scheme is seen to overestimate the steady wave speed significantly. Therefore, it would be concluded that large numerical dispersion errors can deteriorate the accuracy of a method.

The results shown in Fig. 3 also include schemes I to IV, which have not yet been discussed. Scheme I gave the poorest results in the study. This scheme has $\gamma = 2$ and the reaction rate is evaluated fully implicitly ($\delta = 1$, $\alpha = \beta = q = 0$). The scheme has a dispersive error $4r\ \Delta t\ du_{xxt}$, which is greater than that of schemes VIIIa, b and X but less than that of scheme VII. However, the method has, in addition, a numerical diffusion error $(\Delta t/2)U_{tt}$. A comparison between schemes VII, VIIIa, b and X (Fig. 4) and scheme I (Fig. 3) shows that this numerical diffusion term deteriorates the accuracy of the method greatly.

Finally, a comparison of the performance of schemes I to VI (Fig. 3) with that of the generally superior schemes VII to XII (Fig. 4) shows the importance of an accurate evaluation of the reaction term. Schemes VII to XII have first-order truncation errors $\sim\Delta t(U_{tt}-F_t)$. This form of truncation error would lead to a second-order method in the absence of physical diffusion, i.e., $d=0$, and it is seen to be preferable also for the case $d\neq 0$. In the following we describe results obtained using the explicit method VIIIb, which satisfies this latter requirement and which does not have a first-order numerical diffusion truncation error.

Fig. 5 summarizes results for the ozone decomposition flame propagation problem, mentioned earlier. This problem has also been analysed by Bledjian [2] and Margolis

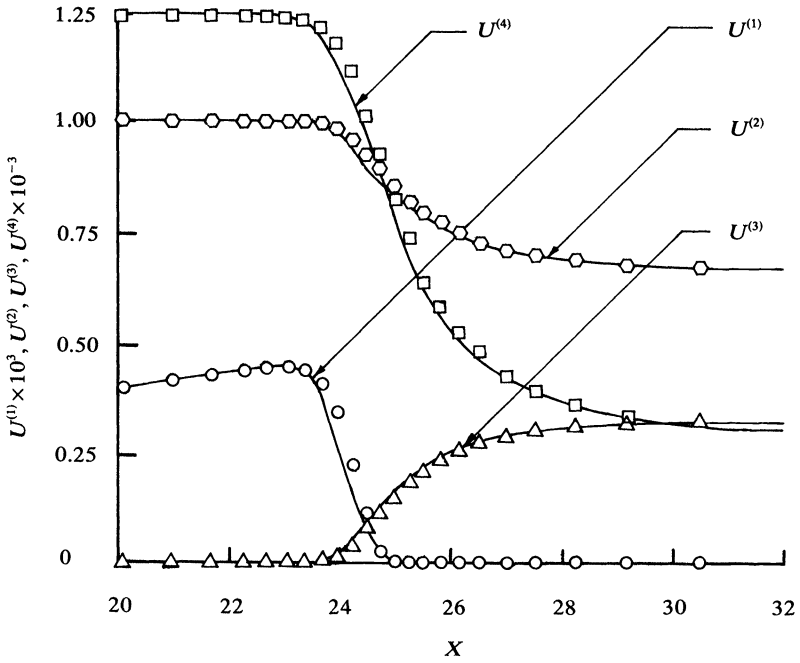


FIG. 5. *Profiles of temperature* ($U^{(4)}\square$), *and atomic oxygen* ($u^{(1)}\bigcirc$), *molecular oxygen* ($u^{(2)}\bigcirc$), *ozone* ($u^{(3)}\triangle$) *mass fractions at* $t=2.1$ ms *after ignition. Minimum* $\Delta x\simeq 0.25$, $\Delta t\simeq 0.3$ μs. (—— results of Margolis [9].)

[9], who used first- and fourth-order accurate methods of lines respectively. Lund [8] studied a similar ozone flame problem and used a first-order fully implicit method. Thus, results are available for comparison. The governing equations are as given in (1) with $K=4$ and $d^{(k)}$ assumed equal and constant. $u^{(1)}$, $u^{(2)}$ and $u^{(3)}$ are the mass fractions of atomic oxygen, molecular oxygen and ozone, respectively, and $u^{(4)}$ is the gas temperature. The initial data, nondimensionalization and reaction terms $F^{(k)}$ were as given by Margolis [9]. The problem models flame propagation in a mixture of $\frac{2}{3}O_2$ and $\frac{1}{3}O_3$(wt). The unburned gas temperature was 300°K, and the pressure (assumed constant through the flame in the model) was 0.821 atm.

Details of the steady wave structure are shown in Fig. 5. The symbols show the computed wave temperature and mass fraction profiles in a portion of the computational domain. The results were obtained using an adaptive grid method which concentrates grid points within the flame. The minimum grid spacing ($\Delta x\cong 0.25$) was arranged to coincide with the point of maximum temperature gradient in the method.

Further details of the adaptive grid method and more detailed ozone flame results are given in Reitz [12].

The solid curves show the results obtained by Margolis [9], who used an equally spaced mesh with $\Delta x = 0.2$. A comparison of the results shows excellent agreement between the two calculations. The present computed flame speed was 49.8 ± 0.1 cm/s, which agrees well with the value of Margolis [9], $S = 49.7$ cm/s. This agreement indicates that lower order methods can be sufficiently accurate for practical computations. The fact that the present results show improved accuracy over the results of Bledjian [2], who gave $S = 54.3$ cm/s, indicates that the first-order numerical dispersion error of scheme VIIIb reduces accuracy less than the first-order diffusive error in his method. This is in agreement with the findings of the model equation parameter study.

Finally, it is of interest to note that the time step used in the ozone flame computation ($\Delta t \cong 0.3$ μs), which is limited by the dominant eigenvalue of the reaction terms (12), is comparable to that used by Bledjian [2] ($\Delta t \sim 0.3$ μs) and by Lund [8] ($\Delta t \cong 0.4$ μs) (values of Δt were not given by Margolis [9]). In Lund's fully implicit method the timestep was controlled by specifying a maximum number of iterations for convergence of the solution each timestep. This agreement in the timesteps indicates that explicit methods can be competitive with implicit methods in certain combustion problems.

Conclusions. The model equation parameter study has shown that numerical diffusion effects can reduce the accuracy of a numerical method significantly. Numerical dispersion truncation errors were also found to be detrimental to accuracy, but only if they were sufficiently large. The study also points out that the reaction term should be accurately represented in the finite difference scheme. Generally, methods which are second-order accurate for the case of no physical diffusion, $d = 0$, were found to be best.

The favorable comparison of the ozone flame computations made with scheme VIIIb and the results of the fourth-order accurate method of Margolis [9] indicates that a lower order method is sufficiently accurate for practical computations. The fact that the results show improved accuracy over the first order results of Bledjian [2] confirms the finding in the model equation study that a first-order numerical dispersion error reduces accuracy less than a corresponding numerical diffusion error. Finally, the fact that the allowable timestep was found to be comparable to the timestep of the implicit method of Lund [8] suggests that explicit numerical methods could prove to be competitive in certain combustion problems.

Acknowledgments. The author would like to thank Professors S. Burstein, E. Isaacson and A. Majda for helpful discussions and comments.

REFERENCES

[1] M. J. Ablowitz and A. Zeppetella, *Explicit solutions of Fisher's equation for a special wave speed*, Dept. Mathematics Preprint, Clarkson College, Potsdam, NY, 1979.

[2] L. Bledjian, *Computation of time-dependent laminar flame structure*, Combust. Flame, 20 (1973), pp. 5–17.

[3] J. Crank and P. Nicolson, *A practical method for numerical integration of solutions of partial differential equations of heat-conduction type*, Proc. Cambridge Philos. Soc., 32 (1947), pp. 50–67.

[4] J. Gazdag and J. Canosa, *Numerical solution of Fisher's equation*, J. Appl. Probability, 11 (1974), pp. 445–457.

[5] D. Hoff, *Stability and convergence of finite difference methods for systems of nonlinear reaction-diffusion equations*, SIAM J. Numer. Anal., 15 (1978), pp. 1161–1177.

[6] A. KOLMOGOROV, I. PETROVSKY AND N. PISCOUNOFF, *Étude de l'équation de la diffusion avec croissance de la quantité de la matière et son application a un problème biologique*, Moscow University Bull. Math., 1 (1937), pp. 1–25.

[7] S. S. LIN, *Theoretical study of a reaction-diffusion model for flame propagation in a gas*, PhD thesis, Dept. Math., University of California, Berkeley, 1979.

[8] C. M. LUND, *A general computer program for calculating time-dependent phenomena involving one-dimensional hydrodynamics, transport and detailed chemical kinetics*, Lawrence Livermore Laboratory Report UCRL-52504, 1978.

[9] S. B. MARGOLIS, *Time-dependent solution of a premixed laminar flame*, J. Comput. Phys., 27 (1978), pp. 410–427.

[10] H. P. MCKEAN, *Application of Brownian motion to the equation of Kolmogorov-Petrovskii-Piskunov*, Comm. Pure Appl. Math., 28 (1975), pp. 323–331.

[11] R. D. REITZ, *The application of an explicit numerical method to a reaction-diffusion system in combustion*, Courant Institute Report COO-3077-162, New York University, 1979.

[12] ———, *Computations of laminar flame propagation using an explicit numerical method*, 18th International Symposium on Combustion, 1980, to appear.

[13] P. J. ROACHE, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1976.

[14] V. K. SAUL'YEV, *Integration of Equations of Parabolic Type by the Method of Nets*, Pergamon, New York, 1964.

[15] D. B. SPALDING, *One-dimensional laminar flame theory for temperature-explicit reaction rates*, Combust. Flame, 1 (1957), pp. 296–307.

[16] C. K. WESTBROOK, J. CREIGHTON, C. LUND AND F. L. DRYER, *A numerical model of chemical kinetics of combustion in a turbulent flow reactor*, J. Phys. Chem., 81 (1977), pp. 2542–2554.

[17] F. A. WILLIAMS, *Combustion Theory*, Addison-Wesley, Reading, MA, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 1, March 1981

0196-5204/81/0201-0009 $01.00/0

STABILITY AND MULTIPLICITY OF SOLUTIONS TO DISCRETIZATIONS OF NONLINEAR ORDINARY DIFFERENTIAL EQUATIONS*

WOLF-JÜRGEN BEYN† AND EUSEBIUS DOEDEL‡

Abstract. A large class of consistent and unconditionally stable discretizations of nonlinear boundary value problems is defined. The number of solutions to the discretizations is compared to the number of solutions of the continuous problem. We state conditions under which these numbers must agree for all sufficiently small mesh sizes. Various examples, including bifurcation problems, illustrate our theoretical results.

Key words. ordinary differential equations, bifurcation theory, discretization, collocation, stability theory, extraneous solutions

1. Introduction. We consider a large class of discretizations for the nonlinear boundary value problem

$$Nu \equiv u^{(n)} + f(x, u, u^{(1)}, \cdots, u^{(n-1)}) = 0, \qquad 0 \leqq x \leqq 1, \tag{1.1}$$

with boundary conditions

$$B^l u \equiv \sum_{k=0}^{n-1} b^0_{k,l} u^{(k)}(0) + \sum_{k=0}^{n-1} b^1_{k,l} u^{(k)}(1) = 0, \qquad 1 \leqq l \leqq n. \tag{1.1a}$$

The discrete approximations all satisfy the conditions for consistency and stability, even on nonuniform meshes without the assumption of a bounded mesh ratio. These approximations are defined in § 2, where we also summarize their convergence properties.

Then, in § 3, we consider the relation between the number of solutions to the discrete problem and the number of solutions to the continuous problem. In general these need not be the same, not even asymptotically, as the mesh size goes to zero. The results of [2] indicate that these numbers of solutions can be guaranteed to agree only under rather restrictive assumptions on the differential equation. Essentially, the result in [2] is that extraneous solutions must disappear, as the mesh size approaches zero, when the lower order part of the differential operator in (1.1) is a sublinear function of its main arguments. We recover this result in § 3 with simple proofs adapted to the special nature of the approximations considered in this paper.

A variety of examples are given in § 4. In addition to problems that do not satisfy the sublinearity condition and that exhibit extraneous solutions, no matter how small the mesh size, we also give examples of discretized bifurcation problems that have extraneous solution branches. One of these examples shows that even if a bifurcation problem does satisfy the restrictive assumption of sublinearity, it still may have extraneous solution branches for large mesh sizes. Our theoretical results then explain why these branches must disappear (or straighten out) as the mesh size decreases. A final example illustrates the effect that discretization may have on the bifurcation diagram associated with a Hopf bifurcation problem.

2. Definitions and basic convergence properties. The discretizations studied can be defined as follows. Introduce a mesh $\{0 = x_0 < x_1 < \cdots < x_J = 1\}$, with $h_j \equiv x_j - x_{j-1}$,

* Received by the editors April 21, 1980, and in revised form October 30, 1980. This work was supported in part by the U.S. Army under grant DAAG29-7B-G-0126 and by FCAC (Québec) under grant EQ-1438.

† Fakultät für Mathematik, Universität Konstanz, Konstanz, West Germany.

‡ Computer Science Department, Concordia University, Montréal, Québec, Canada.

$h \equiv (h_1, h_2, \cdots, h_J)$ and $|h| \equiv \max h_j$. To each mesh point x_j $(0 \leqq j \leqq J-n)$ associate a polynomial $p_j \in P_{n+m-1}$. Here P_d is the space of all polynomials of degree less than or equal to d. Define $\underline{p}_h \equiv \{p_j\}_{j=0}^{J-n}$. Let $P_h^{n,m} \equiv P_h^{n,m}[\mu_1, \mu_2, \cdots, \mu_n]$ denote the linear space of all $\underline{p}_h$ satisfying the boundary conditions (1.1a), in the sense that

$$\sum_{k=0}^{n-1} b_{k,l}^0 p_0^{(k)}(0) + \sum_{k=0}^{n-1} b_{k,l}^1 p_{J-n}^{(k)}(1) = 0, \qquad 1 \leqq l \leqq n, \tag{2.1}$$

as well as *matching conditions* of the form

$$p_j(x_{j+\mu_k}) = p_{j+1}(x_{j+\mu_k}), \qquad 1 \leqq k \leqq n, \quad 0 \leqq j \leqq J-n-1, \tag{2.2}$$

where $1 \leqq \mu_1 \leqq \mu_2 \leqq \cdots \leqq \mu_n \leqq n$. If not all integers μ_k are distinct, then we define this to imply that derivatives also match, in the obvious manner. The discrete method now consists of finding $\underline{p}_h \in P_h^{n,m}$ satisfying the collocation equations

$$Np_j(z_{j,i}) = 0, \qquad 1 \leqq i \leqq m, \qquad 0 \leqq j \leqq J-n, \tag{2.3}$$

where for each j the $z_{j,i}$ are distinct points in $[x_j, x_{j+\mu_n}]$. The $z_{j,i}$ are assumed to be *locally semiuniform*, i.e., $\min_{i_1 \neq i_2} |z_{j,i_1} - z_{j,i_2}| \geqq C|x_{j+\mu_n} - x_j|$, for some constant C which is independent of j and h. This assumption is not strictly necessary (e.g., [6]), but it simplifies the argument somewhat. Note that it does not impose any restrictions on the mesh, if the $z_{j,i}$ are chosen systematically with respect to $[x_j, x_{j+\mu_n}]$.

Examples. The orthogonal collocation methods correspond to taking $\mu_i = 1$, $1 \leqq i \leqq n$, and Gauss $z_{j,i}$ [1], [4]. Spline collocation methods can also be viewed as having $\mu_i = 1$, $1 \leqq i \leqq n$, but with $m = 2$, $z_{j,1} = x_j$, $z_{j,2} = x_{j+1}$ [17]. For a survey of such projection methods see [16]. The generalized finite difference methods of [5], [6], [13], [14], [15], [18] are obtained for general m and $\mu_k = k$, $1 \leqq k \leqq n$.

The essential characteristic of all methods included in the present framework is the fact that all derivatives up to order $n-1$ of each pair of consecutive polynomial components p_j and p_{j+1} match somewhere in $[x_{j+1}, x_{j+\mu_n}]$. But derivatives of order greater than $n-1$ are not required to match. If $\mu_k = 1$ $(1 \leqq k \leqq n)$, this is clear, since this is the case where all derivatives up to order $n-1$ of each p_j, p_{j+1} match identically at x_{j+1}. If not all μ_k are equal, this matching follows from repeated application of Rolle's theorem.

Introduce the following norm: If $\underline{w}_h \in P_h^{n,m}$, then

$$\|\underline{w}_h\|_p = \max_{0 \leqq k \leqq p} \max_{0 \leqq j \leqq J-n} \max_{x \in [x_j, x_{j+\mu_n}]} |w_j^{(k)}(x)|.$$

Also, for $w \in C^p[0, 1]$, we use the notation

$$\|\underline{w}_h - w\|_p \equiv \max_{0 \leqq k \leqq p} \max_{0 \leqq j \leqq J-n} \max_{x \in [x_j, x_{j+\mu_n}]} |w_j^{(k)}(x) - w^{(k)}(x)|.$$

The proof of the following lemma is very similar to that of [6, Lemma 2.1] and will be omitted.

LEMMA 2.1. *Let $\{h^\nu\}_{\nu=1}^\infty$ be a sequence of meshes with $|h^\nu| \to 0$ as $\nu \to \infty$. For each ν let $\underline{w}_{h^\nu} \in P_{h^\nu}^{n,m}$, with $\|\underline{w}_{h^\nu}\|_n \leqq C$. Then there is a subsequence $\{\underline{w}_{h^\nu}\}_{\nu=1}^\infty$ and a function $w \in C^{n-1}[0, 1]$ such that $\|\underline{w}_{h^\nu} - w\|_{n-1} \to 0$ as $\nu \to \infty$.*

Remark. For simplicity we do not notationally distinguish between sequence and subsequence.

Rewrite the collocation equations (2.3) in operator form as

$$N_h \underset{\sim}{p}_h = 0,$$

where N_h maps $P_h^{n,m}$ into $R_h^{n,m} \equiv \mathbb{R}^{m(J-n+1)}$. For $\underset{\sim}{r}_h \in R_h^{n,m}$, $\underset{\sim}{r}_h \equiv \{r_{j,i}: 0 \leqq j \leqq J-n, 1 \leqq i \leqq m\}$, let $\|\underset{\sim}{r}_h\|_\infty \equiv \max_{i,j} |r_{j,i}|$. (This is the usual max norm on $\mathbb{R}^{m(J-n+1)}$.) Let $L_h[\underline{w}_h]$ be the Fréchet derivative of N_h at $\underline{w}_h \in P_h^{n,m}$. Thus $L_h[\underline{w}_h]$ maps $P_h^{n,m}$ linearly into $R_h^{n,m}$ as follows: $L_h[\underline{w}_h]\underline{q}_h = \underset{\sim}{r}_h$, where

$$r_{j,i} = q_j^{(n)}(z_{j,i}) + \sum_{k=0}^{n-1} f_{y_k}(z_{j,i}, w_j(z_{j,i}), \cdots, w_j^{(n-1)}(z_{j,i}))q_j^{(k)}(z_{j,i}),$$

$$1 \leqq i \leqq m, \quad 0 \leqq j \leqq J-n.$$

Here the arguments of f are indicated as $f \equiv f(x, y_0, y_1, \cdots, y_{n-1})$. The induced operator norm is given by

$$\|L_h[\underline{w}_h]\| \equiv \max_{\|\underline{q}_h\|_n = 1} \|L_h[\underline{w}_h]\underline{q}_h\|_\infty.$$

Let $\underline{u}_h \equiv \{u_j(x)\}_{j=0}^{J-n} \in P_h^{n,m}$ interpolate a solution u of (1.1), (1.1a) at the mesh points and at certain additional points $t_{j,i} \in [x_j, x_{j+\mu_n}]$ as follows:

$$u_j(x_{j+\mu_k}) = u(x_{j+\mu_k}), \qquad 0 \leqq k \leqq n, \quad 0 \leqq j \leqq J-n,$$

$$u_j(t_{j,i}) = u(t_{j,i}), \qquad 1 \leqq i \leqq m-1, \quad 0 \leqq j \leqq J-n,$$

where repeated points, if any, denote Hermite interpolation. Also define

$$B_\varepsilon^n(\underline{u}_h) \equiv \{\underline{w}_h \in P_h^{n,m}: \|\underline{w}_h - \underline{u}_h\|_n \leqq \varepsilon\}.$$

We say that $f(x, y_0, y_1, \cdots, y_{n-1})$ has *Lipschitz continuous derivatives* with respect to $y_0, y_1, \cdots, y_{n-1}$ *in a* ρ*-neighborhood of* $v \in C^{n-1}[0, 1]$ if

$$\max_{0 \leqq k \leqq n-1} |f_{y_k}(x, \alpha_0, \cdots, \alpha_{n-1}) - f_{y_k}(x, \beta_0, \cdots, \beta_{n-1})| \leqq K_L \max_{0 \leqq l \leqq n-1} |\alpha_l - \beta_l|,$$

for all $\{\alpha_l, \beta_l\}$ with $\max_l |\alpha_l - v^{(l)}(x)| \leqq \rho$, $\max_l |\beta_l - v^{(l)}(x)| \leqq \rho$, and for all $x \in [0, 1]$. Here $K_L \equiv K_L[v]$ and $\rho \equiv \rho(v)$ are positive constants that do not depend on x.

LEMMA 2.2. *Let* f *have Lipschitz continuous partial derivatives with respect to* $y_0, y_1, \cdots, y_{n-1}$ *in a* ρ*-neighborhood of* u. *Here* $u \in C^{n+m}[0, 1]$ *is an exact solution of* (1.1), (1.1a). *Then for all sufficiently small* $|h|$ *we have*

$$\|L_h[\underline{v}_h] - L_h[\underline{w}_h]\| \leqq nK_L \|\underline{v}_h - \underline{w}_h\|_{n-1},$$

whenever

$$\underline{v}_h, \underline{w}_h \in B_{\rho/2}^{n-1}(\underline{u}_h).$$

A solution $u \in C^n[0, 1]$ of (1.1), (1.1a) is called an *isolated* solution if the linearized problem

$$L[u]v \equiv v^{(n)} + \sum_{k=0}^{n-1} f_{y_k}(x, u, u^{(1)}, \cdots, u^{(n-1)})v^{(k)} = 0,$$

subject to the boundary conditions $B^l v = 0$, $1 \leqq l \leqq n$, admits only $v(x) \equiv 0$ as solution. Below it will also be assumed that each $f_{y_k}(x, y_0, \cdots, y_{n-1})$ is continuous in x, in a ρ-neighborhood of u. By this is meant continuity in x for $|y_l - u^{(l)}(x)| \leqq \rho$, $0 \leqq l \leqq n-1$, $x \in [0, 1]$ with $\rho \equiv \rho[u]$ independent of x. We can now state the following stability result:

THEOREM 2.3. *Let f have Lipschitz continuous derivatives with respect to $y_0, y_1, \cdots, y_{n-1}$ as in Lemma 2.2 and let each f_{y_k} be continuous in x, all in a ρ-neighborhood of an isolated solution $u \in C^{n+m}[0,1]$ of (1.1), (1.1a). Then there are positive constants δ, ε and K such that*

$$\|\underset{\sim}{v}_h - \underset{\sim}{w}_h\|_n \leqq K\|N_h\underset{\sim}{v}_h - N_h\underset{\sim}{w}_h\|_\infty,$$

for all $\underset{\sim}{v}_h$, $\underset{\sim}{w}_h \in B_\varepsilon^n(\underset{\sim}{u}_h)$ and for all meshes h with $|h| \in (0, \delta]$.

For a proof of the statement above, one can first appeal to the general theory in [9]. This simplifies the problem to that of finding a corresponding stability result for the linearized problem. The linear case has been dealt with in [6], using a technique from [11]. Actually the discretizations in this work are slightly more general than those in [6], but this would be hardly noticeable in the proof.

A more direct approach is equally well possible and outlined below.

Sketch of proof. Suppose that the conclusion of the theorem does not hold. Then there exists a sequence of meshes $\{h^\nu\}_{\nu=1}^\infty$ and corresponding $\varepsilon^\nu > 0$, $K^\nu > 0$, and $\underset{\sim}{v}_{h^\nu}$, $\underset{\sim}{w}_{h^\nu} \in B_{\varepsilon^\nu}^n(\underset{\sim}{u}_{h^\nu})$, with $|h^\nu| \to 0$, $\varepsilon^\nu \to 0$ and $K^\nu \to \infty$ as $\nu \to \infty$, such that

$$\|\underset{\sim}{v}_{h^\nu} - \underset{\sim}{w}_{h^\nu}\|_n > K^\nu\|N_{h^\nu}\underset{\sim}{v}_{h^\nu} - N_{h^\nu}\underset{\sim}{w}_{h^\nu}\|_\infty.$$

By the generalized mean value theorem

$$N_{h^\nu}\underset{\sim}{v}_{h^\nu} - N_{h^\nu}\underset{\sim}{w}_{h^\nu} = L_{h^\nu}[\underset{\sim}{v}_{h^\nu}, \underset{\sim}{w}_{h^\nu}](\underset{\sim}{v}_{h^\nu} - \underset{\sim}{w}_{h^\nu}),$$

where

$$L_{h^\nu}[\underset{\sim}{v}_{h^\nu}, \underset{\sim}{w}_{h^\nu}] \equiv \int_0^1 L_{h^\nu}[t\underset{\sim}{v}_{h^\nu} + (1-t)\underset{\sim}{w}_{h^\nu}]\, dt.$$

Thus, if we let $\underset{\sim}{e}_{h^\nu} \equiv (\|\underset{\sim}{v}_{h^\nu} - \underset{\sim}{w}_{h^\nu}\|_n)^{-1}(\underset{\sim}{v}_{h^\nu} - \underset{\sim}{w}_{h^\nu})$, then $\|\underset{\sim}{e}_{h^\nu}\|_n = 1$ and $\|L_{h^\nu}[\underset{\sim}{v}_{h^\nu}, \underset{\sim}{w}_{h^\nu}]\underset{\sim}{e}_{h^\nu}\|_\infty < (K^\nu)^{-1}$. We may assume that $\varepsilon^\nu \leqq \rho/2$ for all ν, so that $\underset{\sim}{v}_{h^\nu}$, $\underset{\sim}{w}_{h^\nu} \in B_{\rho/2}^{n-1}(\underset{\sim}{u}_{h^\nu})$, and also that $|h^\nu|$ is sufficiently small for all ν, so that the estimate of Lemma 2.2 is valid. Therefore

$$\begin{aligned}
\|L_{h^\nu}[\underset{\sim}{u}_{h^\nu}]\underset{\sim}{e}_{h^\nu}\|_\infty &\leqq \|L_{h^\nu}[\underset{\sim}{u}_{h^\nu}]\underset{\sim}{e}_{h^\nu} - L_{h^\nu}[\underset{\sim}{v}_{h^\nu}, \underset{\sim}{w}_{h^\nu}]\underset{\sim}{e}_{h^\nu}\|_\infty + \|L_{h^\nu}[\underset{\sim}{v}_{h^\nu}, \underset{\sim}{w}_{h^\nu}]\underset{\sim}{e}_{h^\nu}\|_\infty \\
&< \int_0^1 \|L_{h^\nu}[t\underset{\sim}{u}_{h^\nu} + (1-t)\underset{\sim}{u}_{h^\nu}] - L_h^\nu[t\underset{\sim}{v}_{h^\nu} + (1-t)\underset{\sim}{w}_{h^\nu}]\|\, dt + (K^\nu)^{-1} \\
&\leqq \int_0^1 nK_L\|t(\underset{\sim}{u}_{h^\nu} - \underset{\sim}{v}_{h^\nu}) + (1-t)(\underset{\sim}{u}_{h^\nu} - \underset{\sim}{w}_{h^\nu})\|_{n-1}\, dt + (K^\nu)^{-1} \\
&\leqq nK_L\varepsilon^\nu + (K^\nu)^{-1} \to 0 \quad \text{as } \nu \to \infty.
\end{aligned}$$

Since $\|\underset{\sim}{e}_{h^\nu}\|_n = 1$ for all ν, it follows from Lemma 2.1 that there is a function $e(x) \in C^{n-1}[0,1]$ and a subsequence $\{h^\nu\}_{\nu=1}^\infty$ such that $\|\underset{\sim}{e}_{h^\nu} - e\|_{n-1} \to 0$ as $\nu \to \infty$. Using the techniques of [6] it is then not difficult to show, that in fact $e \in C^n[0,1]$, $e \not\equiv 0$ and $L[u]e = 0$. Also $e(x)$ satisfies the homogeneous boundary conditions (1.1a). This contradicts the assumption that u is an isolated solution. Thus the statement of the theorem must be true. □

Given the result of Theorem 2.3, it is easy to establish convergence. For this purpose define the *truncation error* $\underset{\sim}{\tau}_h \in R_h^{n,m}$ by

$$\underset{\sim}{\tau}_h \equiv N_h\underset{\sim}{u}_h.$$

Thus $\underset{\sim}{\tau}_h$ has components $\tau_{j,i} = Nu_j(z_{j,i})$. In general, if $u \in C^{n+m}[0,1]$ then $\underset{\sim}{\tau}_h$ satisfies an

estimate of the form

$$\|\underset{\sim}{\tau}_h\|_\infty \leqq M_1[u]|h|^m,$$

but the order can normally be improved by suitable choice of the collocation points [4], [5], [13], [15]. For example the order is at least $m+1$ if for each j the $z_{j,i}$ are chosen to coincide with the roots of

$$\left(\frac{d^n}{dx^n}\right)\left[\prod_{k=0}^{n}(x-x_{j+\mu_k})\prod_{k=1}^{m-1}(x-t_{j,k})\right].$$

Below we also need a bound on the error in interpolating an exact solution u of (1.1), (1.1a) by $\underline{u}_h \in P_h^{n,m}$ in the manner indicated before. For this we have

$$\|\underline{u}_h - u\|_n \leqq M_2[u]|h|^m,$$

provided again that $u \in C^{n+m}[0,1]$.

THEOREM 2.4. *Assume that the conditions of Theorem* 2.3 *are satisfied. Then there is a positive constant* δ *such that* $N_h\underline{p}_h = 0$ *has a unique solution*

$$\underline{p}_h \in P_h^{n,m} \quad \text{in } B_\varepsilon^n(\underline{u}_h) \quad \text{and such that}$$

$$\|\underline{p}_h - u\|_n \leqq (KM_1[u]+M_2[u])|h|^m, \quad \text{whenever } |h| \in (0,\delta].$$

Proof. The existence proof is identical to the corresponding part of the proof of [9, Thm. 3.6]. The error estimate is obtained as follows:

$$\begin{aligned}\|\underline{p}_h - u\|_n &\leqq \|\underline{p}_h - \underline{u}_h\|_n + \|\underline{u}_h - u\|_n\\ &\leqq K\|N_h\underline{p}_h - N_h\underline{u}_h\|_n + M_2[u]|h|^m\\ &\leqq K\|N_h\underline{u}_h\|_n + M_2[u]|h|^m\\ &\leqq (KM_1[u]+M_2[u])|h|^m.\end{aligned}$$

3. Multiplicity of solutions. Theorem 2.4 in the preceding section guarantees that every isolated solution of the continuous problem is approximated by a solution of the discrete problem. Such a discrete solution is then itself necessarily isolated for all sufficiently small $|h|$, because stability implies isolation. The proof of this fact is essentially the same as the proof of [9, Thm. 2.5], where it is given for the continuous problem. Not much can be said if the continuous problem has a solution that is not isolated. For example, in such a case it is possible that the discretization has no solutions that converge to the given nonisolated solution of the continuous problem. Most of these difficulties can be avoided however by considering solution branches instead of single solutions, and following these with a continuation procedure.

Below we state a global convergence theorem. It does not exclude the possibility of existence of extraneous solutions to the discrete problem, not even for small $|h|$. However, under some mild continuity assumptions, the theorem shows that if such extraneous solutions exist, they must tend to infinity in the appropriate norm as $|h| \to 0$. The proof of this fact bears resemblance to that of Thm. 2.3, although it deals directly with the nonlinear equation.

THEOREM 3.1. *Assume that the problem* (1.1), (1.1a), *has exactly* N *solutions* $u^{[l]}(x) \in C^{n+m}[0,1]$, $N<\infty$, *each of which is isolated. Let* f *have Lipschitz continuous derivatives* f_{y_k}, $0 \leqq k \leqq n-1$, *and let each* f_{y_k} *be continuous in* x; *all in a* $\rho[u^{[l]}]$-*neighborhood of each* $u^{[l]}$. *In addition assume that* f *is continuous in all its variables in a* ρ-*neighborhood of* every $v \in C^{n-1}[0,1]$; $\rho[v]>0$. *Then for each* $R>$

$\max_{1\leqq l\leqq N}\|u^{[l]}\|_n$, *there exists a* $\delta_R>0$ *such that the discrete problem* $N_h\underline{p}_h=0$ *has exactly* N *solutions* $\underline{p}_h^{[l]}\in P_h^{n,m}$ *in* $B_R^n(0)$, *for all* h *with* $|h|\in(0,\delta_R]$.

Proof. The assumptions above include those of Theorem 2.4. Hence each $u^{[l]}$ is approximated by a solution $\underline{p}_h^{[l]}$ of $N_h\underline{p}_h=0$. For all sufficiently small $|h|$ these $\underline{p}_h^{[l]}$ must be distinct, due to the isolated nature of each of the $u^{[l]}$. Also, since $R>\max_l\|u^{[l]}\|_n$, the $\underline{p}_h^{[l]}$ must eventually (for small $|h|$) lie in $B_R^n(0)$. Assume now that we can find a sequence of meshes $\{h^\nu\}_{\nu=1}^\infty$ with $|h^\nu|\to 0$ as $\nu\to\infty$, and for each ν solutions $\underline{p}_{h^\nu}\not\equiv\underline{p}_{h^\nu}^{[l]}$, $1\leqq l\leqq N$, of $N_{h^\nu}\underline{p}_{h^\nu}=0$, with $\|\underline{p}_{h^\nu}\|_n\leqq R$. Then by Lemma 2.1 there is a subsequence $\{\underline{p}_{h^\nu}\}_{\nu=1}^\infty$ and a function $p\in C^{n-1}[0,1]$, such that $\|\underline{p}_{h^\nu}-p\|_{n-1}\to 0$ as $\nu\to\infty$.

Let $s\in[0,1]$, and for each ν let $j\equiv j_\nu$ be such that $s\in[x_j, x_{j+\mu_n}]$. For each fixed j let the polynomials $\psi_i\equiv\psi_{j_\nu,i}$ denote the Lagrange interpolating coefficients for the points $z_i\equiv z_{j_\nu,i}$, $i=1,2,\cdots,m$. Since the collocation points are assumed to be locally semi-uniform throughout this work, we have

$$\max_{i,j}\max_{[x_j,\,x_{j+\mu_n}]}|\psi_{j,i}(x)|\leqq K_\psi,$$

for some constant K_ψ that does not depend on h. Write $w^*\equiv(w, w^{(1)},\cdots,w^{(n-1)})$. Consider now the following estimate:

$$\begin{aligned}
&|p_j^{(n)}(s)+f(s,p^*(s))|\\
&\quad=\Big|-\sum_{i=1}^m\psi_i(s)f(z_i,p_j{}^*(z_i))+f(s,p^*(s))\Big|\\
&\quad=\Big|\sum_{i=1}^m\psi_i(s)\{f(s,p^*(s))-f(z_i,p_j{}^*(z_i))\}\Big|\\
&\quad\leqq mK_\psi\max_i\{|f(s,p^*(s))-f(z_i,p^*(s))|\\
&\qquad\qquad+|f(z_i,p^*(s))-f(z_i,p_j{}^*(s))|+|f(z_i,p_j{}^*(s))-f(z_i,p_j{}^*(z_i))|\}.
\end{aligned}$$

The added continuity assumptions can now be used to show that the right-hand side of this final inequality tends to zero uniformly in s as $\nu\to\infty$. From integration, letting $\nu\to\infty$ and differentiation it follows that $p\in C^n[0,1]$, $Np=0$ and $B^lp=0$. Again the details proceed much like those given in the proof of [6, Thm. 2.2]. Thus $\underline{p}_{h^\nu}$ tends to another solution p of the continuous problem. This new solution p must be distinct from the other N solutions, for otherwise the stability result of Thm. 2.3 is violated for one of the $u^{[l]}$. Hence we have derived a contradiction. □

Under rather restrictive assumptions on the form of the differential operator it is sometimes possible to derive a priori bounds of the form $\|\underline{p}_h\|_n\leqq K$ for all solutions of $N_h\underline{p}_h=0$ and for all meshes h with $|h|$ sufficiently small. It then follows from Theorem 3.1 that extraneous solutions must disappear as $|h|\to 0$. This is the case, for example, in the following class of problems:

THEOREM 3.2. *Assume that the conditions of Theorem* 3.1 *hold. Also assume that for all* h *with* $|h|$ *sufficiently small (say* $|h|\in(0,\delta]$, $\delta>0$*) we have*

(i) $N_h=L_h+G_h$, L_h *linear, invertible*, $\|L_h^{-1}\|\leqq K_s$, *and*

(ii) *For each* $\varepsilon>0$ *there exists a positive number* $K(\varepsilon)$ *such that*

$$\|G_h\underline{p}_h\|_\infty\leqq K(\varepsilon)+\varepsilon\|\underline{p}_h\|_n \text{ for all solutions } \underline{p}_h \text{ of } N_h\underline{p}_h=0. \tag{3.1}$$

Here K_s, $K(\varepsilon)$ *and* ε *are independent of* h. *Then there is a* $\delta_1>0$ *such that the discrete problem* $N_h\underline{p}_h=0$ *has exactly* N *solutions for all meshes* h *with* $|h|\in(0,\delta_1]$.

Proof. We need only establish that all solutions of $N_h \underline{p}_h = 0$ are uniformly bounded. But this is trivial since

$$\|\underline{p}_h\|_n \leqq \|L_h^{-1} G_h \underline{p}_h\|_n \leqq K_s \|G_h \underline{p}_h\|_\infty \leqq K_s[K(\varepsilon) + \varepsilon \|\underline{p}_h\|_n].$$

Thus if we choose $\varepsilon < 1/K_s$ then

$$\|\underline{p}_h\|_n < \frac{K_s K(\varepsilon)}{1 - K_s \varepsilon}.$$

COROLLARY 3.3. *In the problem* (1.1), (1.1a) *let* $Lu \equiv u^{(n)}$, *and assume that the linear homogeneous problem* $Lu = 0$, *subject to* (1.1a), *admits the trivial solution only. Also assume that*

$$|f(x, y_0, y_1, \cdots, y_{n-1})| \leqq C_1 + C_2 \{\max_{0 \leqq k \leqq n-1} |y_k|\}^\alpha, \qquad x \in [0, 1], \quad y_k \in \mathbb{R},$$

with $0 \leqq \alpha < 1$. *Further, let the conditions of Theorem* 3.1 *be met. Then the discretization* $N_h \underline{p}_h = 0$ *cannot have extraneous solutions on meshes* h *with* $|h|$ *sufficiently small.*

Proof. We need only verify that a bound of the form (3.1) holds. For this purpose we shall make use of the following elementary inequality. For each given $\hat{\varepsilon} > 0$ there exists a constant $M(\hat{\varepsilon})$ such that

$$r^\alpha \leqq M(\hat{\varepsilon}) + \hat{\varepsilon} r \quad \text{for all } r \geqq 0.$$

Now let $\varepsilon > 0$ be given. Then

$$\begin{aligned}
\|G_h \underline{p}_h\|_\infty &= \max_{i,j} |f(z_{j,i}, p_j^*(z_{j,i}))| \\
&\leqq C_1 + C_2 \max_{i,j} \{\max_{0 \leqq k \leqq n-1} |p_j^{(k)}(z_{j,i})|\}^\alpha \\
&\leqq C_1 + C_2 \|\underline{p}_h\|_{n-1}^\alpha \\
&\leqq C_1 + C_2 \|\underline{p}_h\|_n^\alpha \leqq K(\varepsilon) + \varepsilon \|\underline{p}_h\|_n, \quad \text{with } K(\varepsilon) \equiv C_1 + C_2 M\left(\frac{\varepsilon}{C_2}\right).
\end{aligned}$$

□

Note that Corollary 3.3 applies in particular if $\alpha = 0$; i.e., extraneous solutions must disappear as $|h| \to 0$ when the lower order part of the operator is bounded.

4. Examples. One can easily construct examples of discretizations that have extraneous solutions on a given mesh. It is somewhat more difficult to find a problem where the discretization has extraneous solutions on each of an infinite sequence of meshes with mesh size tending to zero. For stable discretizations of the type treated in [2] and in this work, such examples cannot be found for operators with a sublinear lower order part. The following example is given in [7]. The boundary value problem

$$\begin{aligned}
&u'' + u'|u'| = 0, \qquad x \in [0, 1], \\
&u(0) = u(1) = 1
\end{aligned} \tag{4.1}$$

has $u(x) \equiv 1$ as its unique solution, while the discretization

$$\begin{aligned}
&(u_{j+1} - 2u_j + u_{j-1})/h^2 + (u_j - u_{j-1})/h|(u_j - u_{j-1})/h| = 0, \qquad 1 \leqq j \leqq J-1, \\
&u_0 = u_J = 1
\end{aligned}$$

has in addition to $u_j = 1$ $(0 \leqq j \leqq J)$ also the zig-zag solution $u_j = (-1)^j$ for all uniform

meshes with J even. Of course this difference approximation is only first order accurate. But this is not essential, as the following example indicates.

Example 4.1. To approximately solve

$$\begin{aligned} &u''+2u[u']^2=0, \qquad x\in[0,1],\\ &u(0)=u(1)=1, \end{aligned} \tag{4.2}$$

consider the second order finite difference scheme

$$\begin{aligned} &(u_{j+1}-2u_j+u_{j-1})/h^2+2u_j[(u_{j+1}-u_{j-1})/2h]^2=0, \qquad 1\leqq j\leqq J-1,\\ &u_0=u_J=1. \end{aligned} \tag{4.3}$$

The discretization (4.3) has in addition to the constant solution $u_j=1$ $(0\leqq j\leqq J)$ also the solution

$$u_j=\begin{cases} 1 & \text{if } j=0 \text{ or } 3, \bmod 4,\\ -1 & \text{if } j=1 \text{ or } 2, \bmod 4, \end{cases}$$

provided $J=3 \bmod 4$. This can be verified directly by substitution. However, the continuous problem (4.2) has $u(x)\equiv 1$ as its unique solution. For if $v(x)$ is another solution of (4.1) with $v(s)\neq 1$ for some $s\in[0,1]$, then there exists a $t\in(0,1)$ such that $v(t)\neq 1$, $v'(t)=0$. Hence $v(x)$, as well as the constant function $u(x)\equiv v(t)$, $x\in[0,1]$, solve the initial value problem

$$\begin{aligned} &u''+2u[u']^2=0, \qquad x\in[0,1],\\ &u(t)=v(t), \qquad u'(t)=0. \end{aligned}$$

Therefore $v(x)\equiv v(t)$ for all $x\in[0,1]$, which contradicts the fact that $v(1)=1$. (A similar argument can be used for (4.1).) Thus the solution $u(x)\equiv 1$ is unique. It is also clearly isolated.

For nonlinear problems containing a parameter, it can be instructive to compare the bifurcation diagram of a discretization to that of the continuous problem as is done, e.g., in [3]. As a simple numerical example consider the following:

Example 4.2. Discretize the nonlinear boundary value problem

$$\begin{aligned} &u''+\lambda \sin(u+u^2+u^3)=0, \qquad x\in[0,1],\\ &u(0)=u(1)=0 \end{aligned}$$

by the 4th order accurate, 3 point Numerov formula, with uniform mesh and mesh size $h=\frac{1}{8}$. Using the continuation and branch switching techniques of [10], one obtains the bifurcation diagram of Fig. 4.1. In the diagram the vertical axis represents a discrete L_2-norm of the approximate solution. Observe that $u(x)\equiv 0$ is a solution for all λ. Bifurcation points have been circled. The branches originating from the secondary bifurcation points are actually double branches that coincide in the diagram due to some symmetry. If the mesh size is decreased to $h=\frac{1}{70}$, then the diagram changes to that of Fig. 4.2. This of course can be expected to be more like the actual diagram of the continuous problem. Extraneous solutions, which are abundant in Fig. 4.1, must disappear when the mesh size goes to zero, provided λ is such that the continuous problem has only isolated solutions. This follows from the boundedness of the lower order part of the continuous operator. However, we have observed that within the limits of the given bifurcation diagram, extraneous solutions do not disappear until the number of meshintervals J is taken greater than 50.

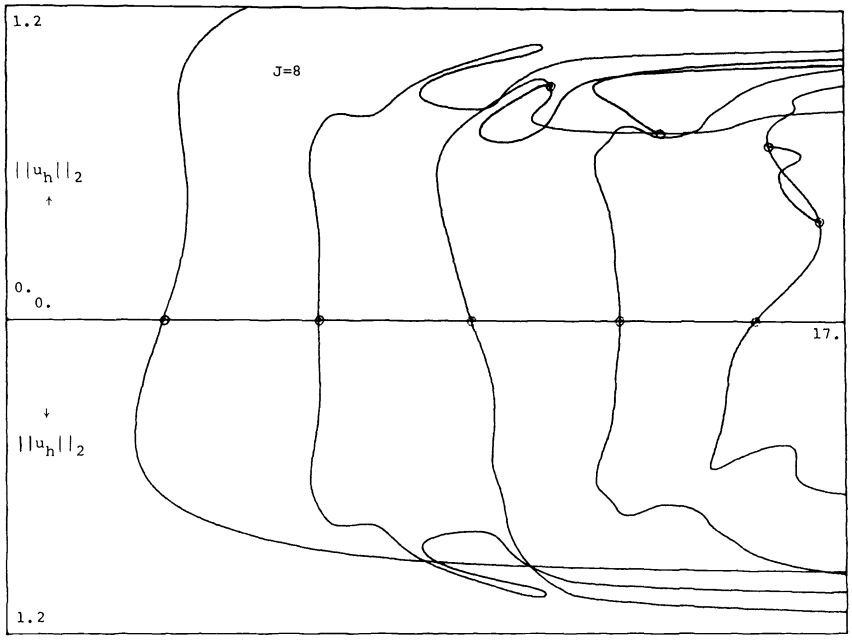


FIG. 4.1a. *Bifurcation diagram of the discretization in Example* 4.2 ($J = 8$).

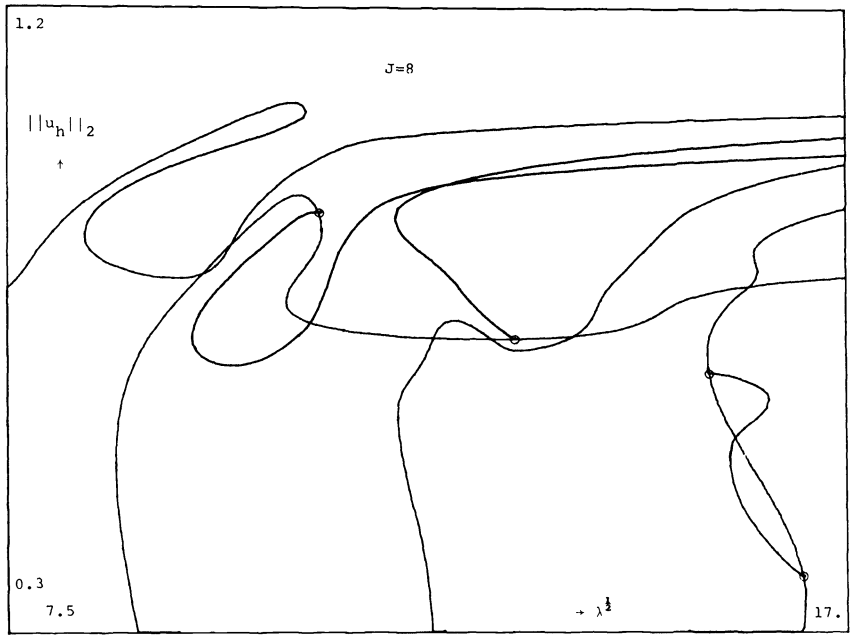


FIG. 4.1b. *Local enlargement of the bifurcation diagram in Fig.* 4.1a.

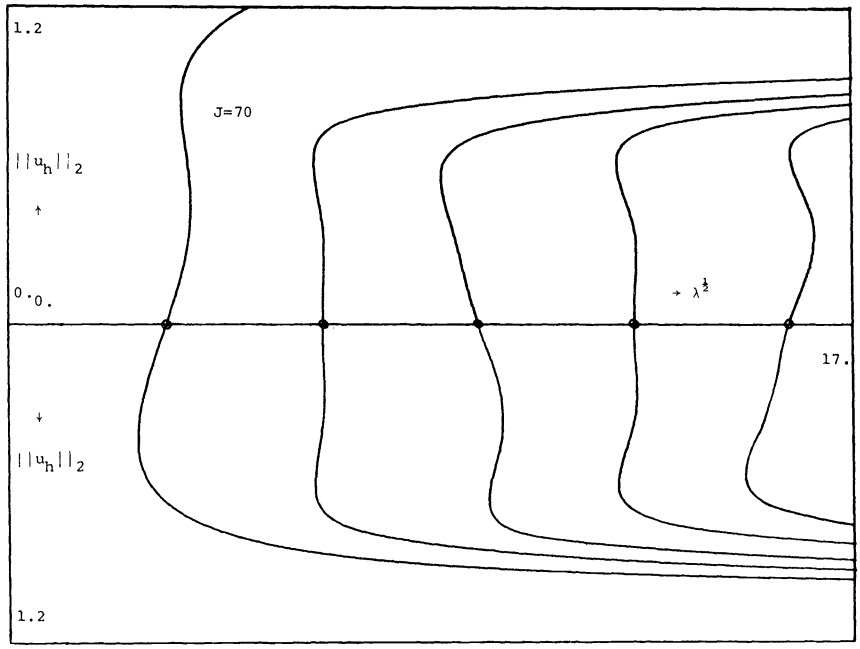


FIG. 4.2. *Bifurcation diagram of the discretization in Example* 4.2 ($J = 70$).

Next we give an analytic example where the bifurcation diagram of a discretization has an extraneous bifurcation point and associated extraneous solutions, no matter how small the mesh size.

Example 4.3. The solutions of the one parameter nonlinear problem

$$\begin{aligned} &v''+(1+[v(\tfrac{1}{2})]^2)\mu v = 0, \qquad x\in[0,1], \\ &v(0)=v(1)=0 \end{aligned} \tag{4.4}$$

are homeomorphically related to those of the linear problem

$$\begin{aligned} &u''+\lambda u = 0, \qquad x\in[0,1], \\ &u(0)=u(1)=0 \end{aligned} \tag{4.5}$$

through the transformation $\lambda = \mu(1+[v(\tfrac{1}{2})]^2)$, $u = v$. The same is true for their respective discretizations

$$\begin{aligned} &(v_{j+1}-2v_j+v_{j-1})/h^2+(1+v_{(1/2)J}^2)\mu v_j = 0, \qquad 1\leqq j\leqq J-1, \\ &v_0 = 0, \quad 4v_{J-1}-3v_J = 0, \quad J \text{ even}, \end{aligned} \tag{4.6}$$

and

$$\begin{aligned} &(u_{j+1}-2u_j+u_{j-1})/h^2+\lambda u_j = 0, \qquad 1\leqq j\leqq J-1, \\ &u_0 = 0, \quad 4u_{J-1}-3u_J = 0, \end{aligned} \tag{4.7}$$

both of which are consistent and stable for isolated solutions. We claim that (4.7) has an eigenvalue λ that is asymptotically (as $h\to 0$) given by $\lambda_E \cong -1/12h^2$. Thus, unlike the eigenvalues $\lambda_k = (k\pi)^2$ of (4.5), which are all positive, the discrete problem (4.7) admits

a negative eigenvalue. To prove this, we note that an eigenvalue of (4.7) has the form

$$u_j = c_1 z_1^j + c_2 z_2^j,$$

where $z_1(\lambda)$ and $z_2(\lambda)$ are the roots of the characteristic equation

$$z^2 - 2z + 1 + \lambda h^2 z = 0. \tag{4.8}$$

Now if z_1 is a root of (4.8) then so is z_1^{-1}. Moreover, the boundary condition $u_0 = 0$ implies that $c_1 = -c_2 \equiv c$. Thus

$$u_j = c[z^j - z^{-j}].$$

The second boundary condition yields the equation

$$4[z^{J-1} - z^{-(J-1)}] - 3[z^J - z^{-J}] = 0. \tag{4.9}$$

Notice that for large J (4.9) has a root $z \equiv z(\lambda)$ with $|z| > 1$. This root asymptotically satisfies $4z^{J-1} - 3z^J = 0$; i.e., $z = \frac{4}{3}$. The corresponding eigenvalue $\lambda_E \cong -1/12h^2$ is obtained from (4.8), and the associated eigenvector is $u_j = c[(\frac{4}{3})^j - (\frac{3}{4})^j]$. Note that $u_j \neq 0$ if $c \neq 0$ and $j \neq 0$.

The corresponding solution branch of the nonlinear discrete problem (4.6) is therefore asymptotically (as $h \to 0$) given by

$$(\mu(c), v_j(c)), \qquad -\infty < c < \infty,$$

where

$$\mu(c) = \frac{\lambda_E}{1 + c[(\frac{4}{3})^{J/2} - (\frac{3}{4})^{J/2}]^2}$$

and

$$v_j(c) = c[(\tfrac{4}{3})^j - (\tfrac{3}{4})^j].$$

(For related techniques see [8, 20].) The extraneous solution branch is shown in Fig. 4.3 for the case $J = 16$. This solution branch exists for all meshes with even J and is extraneous, since the continuous problem (4.4) has no nonzero solutions for negative μ. In particular, for any $\mu < 0$ one can choose h sufficiently small so that $\lambda_E < \mu$. For such μ the discretization (4.6) evidently has three solutions, unlike the continuous problem (4.4).

Finally we give a numerical example, where the objective is to determine time periodic solutions to an autonomous system containing a parameter. It is generally necessary to reformulate the problem as a boundary value problem on a fixed interval, say $[0, 2\pi]$. This allows the numerical computation to proceed past turning points and makes the computation of asymptotically unstable solutions possible. A general code for the bifurcation analysis of autonomous systems has been developed by the second author and will be described more completely in a forthcoming paper.

Example 4.4. The dynamic behavior of a single first order chemical reaction in a continuously stirred tank reactor can be modeled by the ordinary differential equation

$$\begin{aligned} u_1' &= -u_1 + B\,\mathrm{Da}\,(1-u_2)\,e^{u_1} - \beta u_1, \\ u_2' &= -u_2 + \mathrm{Da}\,(1-u_2)\,e^{u_1}, \end{aligned} \tag{4.10}$$

where B, β and Da are dimensionless parameters.

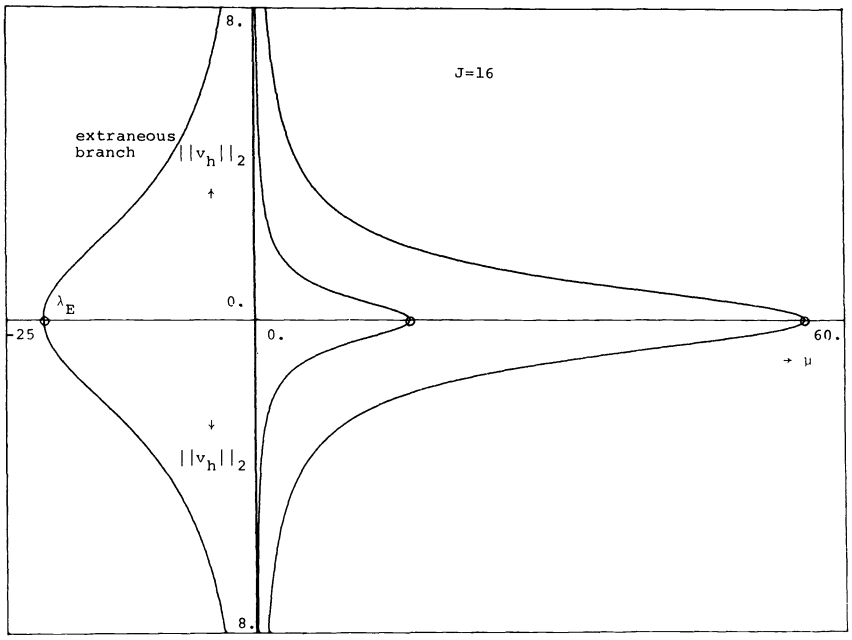


FIG. 4.3. *Graph of the extraneous branch in Example* 4.3.

For previous computations with initial value techniques, see [19]. Using vector notation and scaling the period to the interval $[0, 2\pi]$, we can write (4.10) as

$$u'(t) = \frac{\rho}{2\pi} f(u(t), \lambda), \qquad \lambda \equiv Da, \tag{4.11}$$

$$u(0) = u(2\pi). \tag{4.12}$$

Further, we use the specific values $\beta = 3$, $B = 14$. To remove the nonuniqueness, due to the fact that a periodic solution can be freely translated in time, we impose the orthogonality condition

$$(u(0) - u_0(0))^T f(u_0(0), \lambda_0) = 0. \tag{4.13}$$

Here (u_0, λ_0, ρ_0) denotes a given solution, while (u, λ, ρ) is the solution to be determined from (4.11), (4.12), (4.13) and

$$\|u - u_0\|^2 + (\lambda - \lambda_0)^2 + (\rho - \rho_0)^2 = \Delta s^2. \tag{4.14}$$

This procedure is repeated stepwise along a branch of periodic solutions. We discretize (4.11) by the method of collocation at Gauss points, using the piecewise polynomial space whose elements are globally continuous and quadratic polynomials in each mesh interval. With an adaptive mesh, 71 mesh points and two collocation points per mesh interval, the corresponding bifurcation diagram in [19] is recovered. (See Fig. 4.4.) The primary solution branch is a branch of steady states, while the secondary branch consists of periodic solutions. Along a significant portion of the periodic branch the solution changes very rapidly in a very small interval. Moreover, the location of this small interval does not remain fixed along the branch. The adaptive mesh is for this reason a necessity. For example, 71 *uniformly* spaced mesh points do not reproduce Fig. 4.4. A typical result is given in Fig. 4.5. Here an insufficient number of 10 uniformly distributed

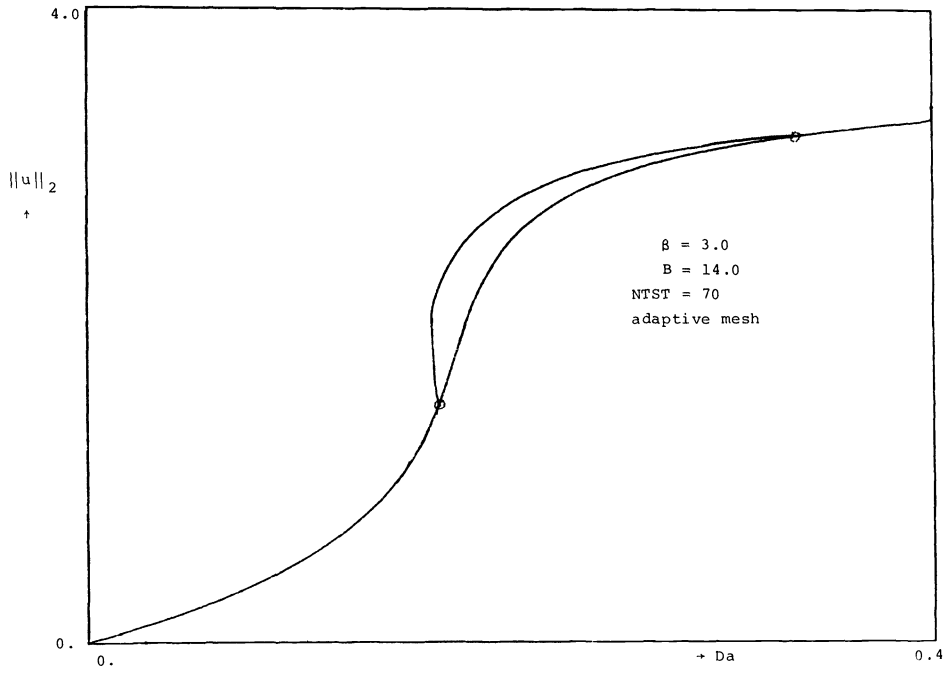


FIG. 4.4. *Bifurcation diagram for* (4.10).

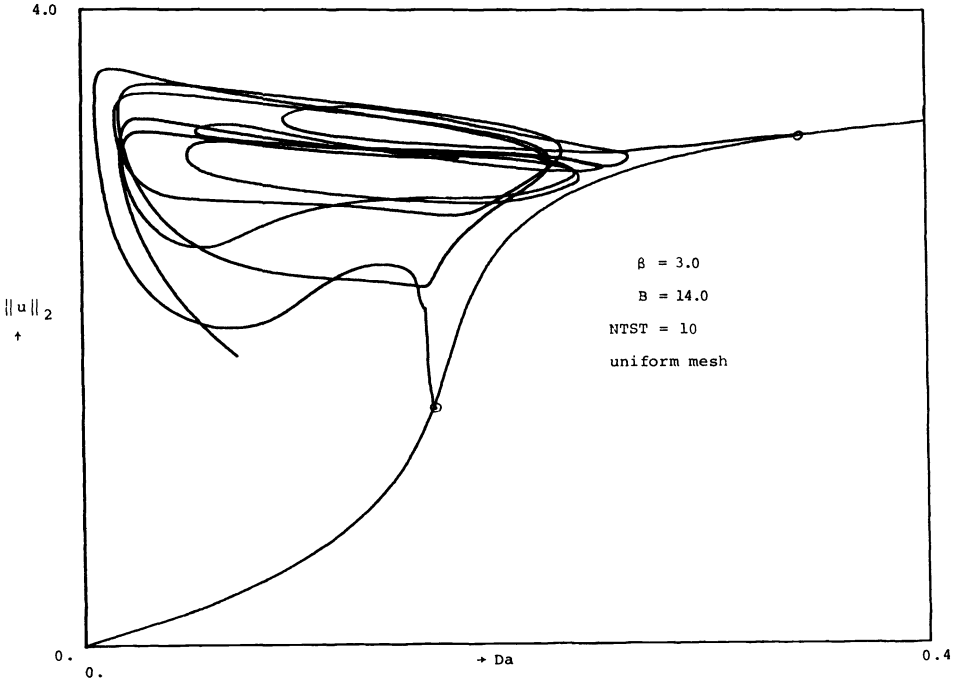


FIG. 4.5. *Effect of an inaccurate discretization on the bifurcation diagram of* (4.10).

mesh points is used. Note the abundance of extraneous solutions. The ability of the method to compute past turning points is especially well illustrated, although the resulting diagram may have little physical significance.

Remark. Although the side condition (4.13) is theoretically sound, i.e., local existence theorems can be based upon it, for practical numerical computation the integrated form

$$\int_0^{2\pi} (u(t)-u_0(t)^T) f(u_0(t), \lambda_0)\, dt = 0$$

is preferable. The modified condition minimizes motion of regions of rapid change in u when progressing along a branch of periodic solutions. This property significantly aids the efficiency of automatic mesh selection. Details will be presented elsewhere.

Acknowledgment. The second author wishes to thank Dr. Rita Meyer-Spasche of the Max-Planck Institut für Plasmaphysik for some stimulating discussions.

REFERENCES

[1] U. ASCHER, J. CHRISTIANSEN AND R. D. RUSSELL, *A collocation solver for mixed order systems of boundary value problems*, Math. Comp., 33 (1978), pp. 659–679.

[2] W.-J. BEYN, *On the convergence of the finite difference method for nonlinear ordinary boundary value problems*, in Constructive Methods for Nonlinear Boundary Value Problems and Nonlinear Oscillations, Proc. Oberwolfach Research Inst., Nov. 19–25, 1978, J. Albrecht, L. Collatz, K. Kirchgässner, eds., Birkhäuser-Verlag, Basel, 1979, ISNM 48, pp. 9–19.

[3] E. BOHL, *On the bifurcation diagram of discrete analogues for ordinary bifurcation problems*, Math. Meth. Appl. Sci., 1 (1979), pp. 566–571.

[4] C. DE BOOR AND B. SWARTZ, *Collocation at Gaussian points*, SIAM J. Numer. Anal., 10 (1973), pp. 582–606.

[5] E. J. DOEDEL, *Finite difference collocation methods for nonlinear two point boundary value problems*, SIAM J. Numer. Anal., 16 (1979), pp. 173–185.

[6] E. J. DOEDEL, *Some stability theorems for finite difference collocation methods on nonuniform meshes*, BIT, 20 (1980), pp. 58–66.

[7] R. GAINES, *Difference equations with boundary value problems for second order nonlinear ordinary differential equations*, SIAM J. Numer. Anal., 11 (1974), pp. 411–433.

[8] R. D. GRIGORIEFF, *Diskrete Approximation von Eigenwertproblemen*, II, *Konvergenzordnung*, Numer. Math., 24 (1975), pp. 415–433.

[9] H. B. KELLER, *Approximation methods for nonlinear problems with application to two point boundary value problems*, Math. Comp., 29 (1975), pp. 464–474.

[10] 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.

[11] H. O. KREISS, *Difference approximations for boundary and eigenvalue problems for ordinary differential equations*, Math. Comp., 26 (1972), pp. 605–624.

[12] T. R. LUCAS AND G. W. REDDIEN, *Some collocation methods for nonlinear boundary value problems*, SIAM J. Numer. Anal., 9 (1972), pp. 341–356.

[13] R. E. LYNCH AND J. R. RICE, *A high-order difference method for differential equations*, Math. Comp., 34 (1980), pp. 333–372.

[14] M. R. OSBORNE, *A method for finite difference approximation to ordinary differential equations*, Computer J., 7 (1964), pp. 58–65.

[15] M. R. OSBORNE, *Collocation, difference equations, and stitched function representations*, Proc. Dublin Conf. in Numerical Analysis, Dublin, 1974.

[16] G. W. REDDIEN, *A survey of projection methods*, SIAM Rev., 22 (1980), pp. 156–171.

[17] R. D. RUSSELL AND L. F. SHAMPINE, *A collocation method for boundary value problems*, Numer. Math., 19 (1972), pp. 1–28.

[18] B. SWARTZ, *Compact implicit difference schemes for a differential equation's side conditions*, Math. Comp., 35 (1980), pp. 733–746.

[19] A. UPPAL, W. H. RAY AND A. B. POORE, *On the dynamic behaviour of continuous stirred tank reactors*, Chem. Eng. Sci., 29 (1974), pp. 976–985.

[20] J. M. VARAH, *On the stability of boundary conditions for separable difference approximations to parabolic equations*, SIAM J. Numer. Anal., 14 (1977), pp. 1114–1125.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0001 $01.00/0

A GENERALIZED EIGENVALUE APPROACH FOR SOLVING RICCATI EQUATIONS*

P. VAN DOOREN†

Abstract. A numerically stable algorithm is derived to compute orthonormal bases for any deflating subspace of a regular pencil $\lambda B - A$. The method is based on an update of the QZ-algorithm, in order to obtain any desired ordering of eigenvalues in the quasitriangular forms constructed by this algorithm. As applications we discuss a new approach to solve Riccati equations arising in linear system theory. The computation of deflating subspaces with specified spectrum is shown to be of crucial importance here.

Key words. generalized eigenvalue problem, Riccati equation, optimal control, spectral factorization

1. Introduction. The computation of deflating subspaces with a specified spectrum has not received a great deal of attention until it was recently applied to the solution of the optimal control problem of a linear discrete time system [5], [15]. Before the development of reliable algorithms for the generalized eigenvalue problem [13], [16], these problems were often reduced to an equivalent standard eigenvalue problem and gave rise to the computation of invariant subspaces with a specified spectrum [8], [14], [17], [21]. The matrix involved in this standard eigenvalue problem does not consist of given data but has to be computed, which unfortunately requires inverses of possibly ill-conditioned matrices. In [5], [12], [15] the use of a generalized eigenvalue problem is recommended as a safer alternative, and attention is drawn to the absence of appropriate software for computing deflating subspaces of a regular pencil. In this paper we try to fill this gap, and we also exploit this new tool in a class of related problems arising in linear system theory. We thereby develop a new approach to tackle these problems in a numerically sound way.

In the rest of this section we briefly review some notions that we will need in later sections. The material covered here can be found, e.g., in [13], [18], [19], [20].

Notation will be as follows. We use uppercase for matrices and lowercase for vectors and scalars. $\mathbb{R}$ and $\mathbb{C}$ are the fields of real and complex numbers, respectively. We use A^* (resp. x^*) for the conjugate transpose of a complex matrix A (resp. vector x) and A' (resp. x') for the transpose of a real matrix A (resp. vector x). $\|\cdot\|_2$ denotes the spectral norm of a matrix and the Euclidean norm of a vector. A complex (real) square matrix A is called *unitary* (*orthogonal*) when $A^*A = I$ ($A'A = I$). When no explicit distinction is made between the complex and real case, we use the term unitary and the notation A^* for the real case as well.

Recently, more attention has been paid to the *generalized eigenvalue problem* (GEP):

$$Ax = \lambda Bx, \tag{1}$$

where B is not necessarily invertible but where the *pencil* $\lambda B - A$ is *regular*, i.e.,

$$\det(\lambda B - A) \not\equiv 0. \tag{2}$$

* Received by the editors July 15, 1980. This research was supported by the National Science Foundation under grant ENG78-10003 and by the U.S. Air Force under grant AFOSR-79- 0094.

† Departments of Electrical Engineering and Computer Science, Stanford University, Stanford, California 94305. Present address, Philips Research Lab., Av. Van Becelaere 2, Box 8, B-1170 Brussels, Belgium.

When the coefficients of the matrices A and B belong to $\mathbb{C}$, there exist unitary transformations Q and Z reducing the $n \times n$ pencil $\lambda B - A$ to the upper triangular form

$$(3) \qquad Q^*(\lambda B - A)Z = \lambda \hat{B} - \hat{A} = \lambda \begin{bmatrix} \hat{b}_{11} & \cdots & * \\ & \ddots & \vdots \\ 0 & & \hat{b}_{nn} \end{bmatrix} - \begin{bmatrix} \hat{a}_{11} & \cdots & * \\ & \ddots & \vdots \\ 0 & & \hat{a}_{nn} \end{bmatrix}.$$

The ratios $\lambda_i = \hat{a}_{ii}/\hat{b}_{ii}$ are called the *generalized eigenvalues* of the pencil $\lambda B - A$. The set $\{\lambda_1, \cdots, \lambda_n\}$ is called the *spectrum* of $\lambda B - A$ and is denoted by $\Lambda(B, A)$; it may contain repeated elements. Notice that λ_i may be infinite (when $\hat{b}_{ii} = 0$) but it is never undetermined (i.e., $\lambda_i = 0/0$), since $\hat{a}_{ii} = \hat{b}_{ii} = 0$ implies $\det(\lambda \hat{B} - \hat{A}) \equiv 0$ and hence $\det(\lambda B - A) \equiv 0$. As a consequence the matrix $\hat{a}_{ii}B - \hat{b}_{ii}A$ is singular. The vectors x_i satisfying

$$(4) \qquad (\hat{a}_{ii}B - \hat{b}_{ii}A)x_i = 0$$

are called *generalized eigenvectors* of $\lambda B - A$ corresponding to λ_i. If the eigenvalue $\lambda_i = \hat{a}_{ii}/\hat{b}_{ii}$ has a larger multiplicity than the number of independent solutions x_i of (4), then one can define *generalized principal vectors* $\tilde{x}_i$ of $\lambda B - A$ corresponding to λ_i. Since we do not need this concept in the sequel, we do not go into further details about it.

In the real case the decomposition (3) also exists but involves complex matrices Q, Z, $\hat{A}$ and $\hat{B}$ when $\Lambda(B, A)$ contains complex elements. Under orthogonal transformations Q and Z, $\lambda B - A$ can be transformed to the quasi upper triangular form

$$(5) \qquad Q'(\lambda B - A)Z = \lambda \hat{B} - \hat{A} = \lambda \begin{bmatrix} \hat{B}_{11} & \cdots & * \\ & \ddots & \vdots \\ 0 & & \hat{B}_{kk} \end{bmatrix} - \begin{bmatrix} \hat{A}_{11} & \cdots & * \\ & \ddots & \vdots \\ 0 & & \hat{A}_{kk} \end{bmatrix},$$

where the diagonal pencils $\lambda \hat{B}_{ii} - \hat{A}_{ii}$ have sizes $d_i = 1$ or 2 and the $\hat{B}_{ii}$ are upper triangular. If $d_i = 1$ then $\Lambda(\hat{B}_{ii}, \hat{A}_{ii})$ is real (possibly infinite). If $d_i = 2$ then $\Lambda(\hat{B}_{ii}, \hat{A}_{ii})$ contains two (finite) complex conjugate numbers. The spectrum of $\lambda B - A$ is the union of the sets $\Lambda(\hat{B}_{ii}, \hat{A}_{ii})$, as can be seen from an additional (unitary) reduction of (5) to (3). An algorithm has been derived recently to obtain decompositions of the type (3) and (5) in a numerically stable way [13]. When $B = I$, (1) boils down to the *standard eigenvalue problem* (SEP):

$$(6) \qquad Ax = \lambda x.$$

It is readily verified that the decompositions (5) and (3) then reduce to the classical Schur decompositions of the real or complex matrix A, respectively. We therefore call (3) and (5) *generalized Schur decompositions* of the regular pencil $\lambda B - A$. In the sequel we drop the term "generalized" when no confusion is possible from the context. The notion of eigenvector in the GEP can be extended to the notion of *deflating subspace* $\mathcal{X}$ of a regular pencil $\lambda B - A$, satisfying

$$(7) \qquad \dim(B\mathcal{X} + A\mathcal{X}) = \dim \mathcal{X},$$

where $\dim \mathcal{S}$ denotes the dimension of a subspace $\mathcal{S}$. Let $\mathcal{X}$ have dimension l, and suppose that the l first columns of the unitary matrices Q and Z, partitioned as

$$(8) \qquad Z = [\underbrace{Z_1}_{l} | \underbrace{Z_2}_{n-l}], \qquad Q = [\underbrace{Q_1}_{l} | \underbrace{Q_2}_{n-l}],$$

span the spaces $\mathcal{X}$ and $A\mathcal{X} + B\mathcal{X}$, respectively. Then it follows from (7) that $Q_2^* A Z_1 =$

$Q_2^* B Z_1 = 0$, or

$$Q^*(\lambda B - A)Z = \lambda \left[\begin{array}{c|c} \hat{B}_{11} & \hat{B}_{12} \\ \hline 0 & \hat{B}_{22} \end{array}\right] - \left[\begin{array}{c|c} \hat{A}_{11} & \hat{A}_{12} \\ \hline 0 & \hat{A}_{22} \end{array}\right] \begin{matrix} \} l \\ \} n-l \end{matrix}. \tag{9}$$

Conversely, if (8), (9) hold then the columns of Z_1 span a deflating subspace $\mathscr{X}$ according to (7). For $l = 1$, $\mathscr{X}$ is an eigenvector of $\lambda B - A$ corresponding to the eigenvalue $\Lambda(\hat{B}_{11}, \hat{A}_{11})$. For any l, $\Lambda(\hat{B}_{11}, \hat{A}_{11})$ is a subset of $\Lambda(B, A)$ and is denoted as $\Lambda(B, A)|_{\mathscr{X}}$ (the spectrum of $\lambda B - A$ restricted to $\mathscr{X}$). The deflating subspace $\mathscr{X}$ is uniquely determined by $\Lambda(B, A)|_{\mathscr{X}}$ when this subset is disjoint from the rest of $\Lambda(B, A)$ ($\mathscr{X}$ is then spanned by the eigenvectors and principal vectors corresponding to the spectrum $\Lambda(B, A)|_{\mathscr{X}}$). All this also holds for the real case. For the case $B = I$, the definition (7) of a deflating subspace reduces to the definition of an *invariant subspace* $\mathscr{X}$ of A, since $\dim(\mathscr{X} + A\mathscr{X}) = \dim \mathscr{X}$ is equivalent to $A\mathscr{X} \subset \mathscr{X}$. Notice also that in the SEP Q is equal to Z in (8), (9).

It follows now immediately from (9) that the Z-matrix in the Schur decomposition (3) yields orthonormal bases for deflating subspaces of dimension 1 to $n-1$, since the right-hand side of (3) has a block partitioning of the type (9) for $l = 1, \cdots, n-1$. This also holds for the "real" Schur decomposition (5) for those l that are conformable with the block partitioning in (5), namely

$$l = \sum_{j=1}^{i} d_j \quad \text{for } i = 1, \cdots, k-1. \tag{10}$$

In this paper we consider the computation of a deflating subspace $\mathscr{X}$ with prescribed spectrum $\Lambda(B, A)|_{\mathscr{X}} = \{\mu_1, \cdots, \mu_l\}$. From the above it follows that the l first columns of Z in (3) form an orthonormal basis for such a space $\mathscr{X}$ if and only if the sets $\{\lambda_i = \hat{a}_{ii}/\hat{b}_{ii} | i = 1, \cdots, l\}$ and $\{\mu_i | i = 1, \cdots, l\}$ are equal except for the ordering of their elements. In the real case, this also holds for the matrix Z in (5) when l satisfies (10). The complex elements in $\{\mu_i | i = 1, \cdots, l\}$ must therefore appear in conjugate pairs.

The problem thus reduces to obtaining decompositions of the type (3) and (5) but with prescribed ordering of the eigenvalues occurring on the diagonal. In the next section we show how to solve this problem by deriving a method to interchange the order of the eigenvalues in the decompositions (3) and (5), which were previously obtained by the QZ-algorithm. The method is proved to be numerically stable. In § 3 we apply this new tool to derive new methods for solving Riccati equations arising in linear system theory. In these methods, deflating subspaces with specified spectrum (namely, all the eigenvalues inside the unit circle or all the eigenvalues in the left half-plane) have to be computed. In § 4 we give some numerical examples.

2. Reordering. It is clear that the 1×1 and 2×2 diagonal blocks in the decompositions (3) and (5) can be reordered in an arbitrary way by using a method to interchange two consecutive blocks only. This idea was used, e.g., in the SEP to obtain standard Schur forms with an arbitrary ordering of the eigenvalues [8], [17], [21]. The method described hereafter can be viewed as a stable generalization of it to the GEP. (An unstable generalization was attempted in [15].) We thus want to find unitary transformations Q and Z such that

$$Q^* A Z = Q^* \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline 0 & A_{22} \end{array}\right] Z = \hat{A} = \left[\begin{array}{c|c} \hat{A}_{11} & \hat{A}_{12} \\ \hline 0 & \hat{A}_{22} \end{array}\right], \tag{11a}$$

$$Q^* B Z = Q^* \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline 0 & B_{22} \end{array}\right] Z = \hat{B} = \left[\begin{array}{c|c} \hat{B}_{11} & \hat{B}_{12} \\ \hline 0 & \hat{B}_{22} \end{array}\right], \tag{11b}$$

where $\Lambda(B_{11}, A_{11}) = \Lambda(\hat{B}_{22}, \hat{A}_{22})$ and $\Lambda(B_{22}, A_{22}) = \Lambda(\hat{B}_{11}, \hat{A}_{11})$, and where the dimensions d_1 and d_2 are either 1 or 2.

Moreover, we want the transformations Q and Z to be numerically stable. In order to prove this, we use a standard error analysis [23] of (possibly complex) transformations of the type

$$G^* y = G^* \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ 0 \end{bmatrix}, \tag{12a}$$

where G is the (possibly complex) Givens transformation

$$G = \begin{bmatrix} c & -\bar{s} \\ s & \bar{c} \end{bmatrix}, \qquad c\bar{c} + s\bar{s} = 1, \tag{12b}$$

constructed to annihilate y_2. Let $\tilde{c}, \tilde{s}$ (defining $\tilde{G}$) and $\tilde{y}_1$ be the computed versions of c, s and $\hat{y}_1$, respectively, and let ε be the machine precision of the computer; then a backward error analysis yields (for a standard construction of such transformations)

$$\tilde{G}^*(y + e_y) = \begin{bmatrix} \tilde{y}_1 \\ 0 \end{bmatrix}, \qquad \|e_y\|_2 \leqq 6 \cdot \varepsilon \|y\|_2. \tag{13}$$

Here we assume that the 0 element is not computed but put equal to zero. When performing the transformation $G^* z = \hat{z}$ for an arbitrary vector z, we have, similarly,

$$\tilde{G}^*(z + e_z) = \begin{bmatrix} \tilde{z}_1 \\ \tilde{z}_2 \end{bmatrix}, \qquad \|e_z\|_2 \leqq 6 \cdot \varepsilon \|z\|_2. \tag{14}$$

In the sequel $\mathcal{G}_{ij}$ denotes the class of matrices representing Givens transformations between columns or rows i and j. We prove that, by using transformations in this class for the reduction (11), the backward error can be bounded with respect to

$$\Delta = \max\{\|A\|_2, \|B\|_2\}. \tag{15}$$

Case I. $d_1 = d_2 = 1$. This may occur in both decompositions (3) and (5). We thus assume that the matrices can be complex. We have the following configuration:

$$Q^* A Z = Q^* \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{bmatrix} Z = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} \\ 0 & \hat{a}_{22} \end{bmatrix} = \hat{A}, \tag{16a}$$

$$Q^* B Z = Q^* \begin{bmatrix} b_{11} & b_{12} \\ 0 & b_{22} \end{bmatrix} Z = \begin{bmatrix} \hat{b}_{11} & \hat{b}_{12} \\ 0 & \hat{b}_{22} \end{bmatrix} = \hat{B}. \tag{16b}$$

We can assume without loss of generality that $|b_{22}| \geqq |a_{22}|$ (if this is not the case the role of A and B should be interchanged). A construction of Q and Z such that the order of the eigenvalues is interchanged follows then immediately from (8), (9). Indeed, we have $\Lambda(b_{22}, a_{22}) = \Lambda(\hat{b}_{11}, \hat{a}_{11})$ if the first column z_1 of Z is an eigenvector of $\lambda B - A$ corresponding to $\Lambda(b_{22}, a_{22})$ or

$$(a_{22} B - b_{22} A) Z = \left[\begin{array}{c|c} 0 & * \\ 0 & \end{array}\right]. \tag{17}$$

Notice that the last row of $H = (a_{22} B - b_{22} A)$ is zero:

$$H = \begin{bmatrix} x_1 & x \\ 0 & 0 \end{bmatrix}. \tag{18}$$

In order to obtain (17), we thus can choose a $Z \in \mathcal{G}_{12}$ annihilating x_1 in (18). It follows from (17) that Bz_1 and Az_1 are parallel, and (16) is then obtained by choosing a $Q \in \mathcal{G}_{12}$ annihilating x_2 in Bz_1:

$$Q^*BZ = Q^*\begin{bmatrix} x & x \\ x_2 & x \end{bmatrix} = \begin{bmatrix} x & x \\ 0 & x \end{bmatrix}. \tag{19}$$

The assumption $|b_{22}|/|a_{22}| = |\hat{b}_{11}|/|\hat{a}_{11}| \geqq 1$ implies that $\hat{b}_{11} \neq 0$, and Q^*Az_1 can then only be parallel to Q^*Bz_1 if Q^*AZ is indeed upper triangular.

We now prove the numerical stability of the method. As in (13), (14), computed elements are denoted by an upper tilde $\tilde{(\cdot)}$. Using the analysis (13) (14) above, it is easy to prove that all the ε_i, $i = 1, \cdots, 9$ below are of the order of the machine accuracy ε of the computer.

An error analysis of (17), (18) yields

$$(a_{22}B - b_{22}A + F)\tilde{Z} = \begin{bmatrix} 0 & x \\ 0 & 0 \end{bmatrix} \quad \text{for } \|F\|_2 = \varepsilon_1 \|a_{22}B - b_{22}A\|_2, \tag{20}$$

and of (19) yields

$$\tilde{Q}^*(B + E_b)\tilde{Z} = \begin{bmatrix} \tilde{b}_{11} & \tilde{b}_{12} \\ 0 & \tilde{b}_{22} \end{bmatrix} \quad \text{for } \|E_b\|_2 = \varepsilon_2 \Delta. \tag{21}$$

We prove that there also exists a backward error E_a such that

$$\tilde{Q}^*(A + E_a)\tilde{Z} = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ 0 & \tilde{a}_{22} \end{bmatrix} \quad \text{for } \|E_a\|_2 = \varepsilon_3 \Delta. \tag{22}$$

An error analysis of Q^*AZ using (14) yields

$$\tilde{Q}^*(A + E_c)\tilde{Z} = \begin{bmatrix} \tilde{a}_{11} & \tilde{a}_{12} \\ \tilde{a}_{21} & \tilde{a}_{22} \end{bmatrix} \quad \text{for } \|E_c\|_2 = \varepsilon_4 \Delta. \tag{23}$$

We only have to prove that $\tilde{a}_{21} \simeq \varepsilon \Delta$ in order to obtain (22) by putting $\tilde{a}_{21}$ equal to zero.

Let us therefore denote the (2, 1) elements of $\tilde{Q}^*(a_{22}B - b_{22}A)\tilde{Z}$, $\tilde{Q}^*B\tilde{Z}$ and $\tilde{Q}^*A\tilde{Z}$ by η_1, η_2 and η_3, respectively. They clearly satisfy the relation

$$a_{22} \cdot \eta_2 - b_{22} \cdot \eta_3 = \eta_1. \tag{24}$$

From (20), (21) and (23) it follows that

$$|\eta_1| \leqq \varepsilon_5\{|a_{22}| \|B\|_2 + |b_{22}| \|A\|_2\}, \tag{25a}$$

$$|\eta_2| \leqq \varepsilon_6 \Delta, \tag{25b}$$

$$|\tilde{a}_{21}| \leqq |\eta_3| + \varepsilon_7 \Delta. \tag{25c}$$

Using (25) and the assumption $|b_{22}| \geqq |a_{22}|$ in (24) we obtain

$$\begin{aligned} |\eta_3| &\leqq |\eta_2||a_{22}|/|b_{22}| + |\eta_1|/|b_{22}| \\ &\leqq \varepsilon_6 \Delta + \varepsilon_5\{\Delta + \Delta\} = \varepsilon_8 \Delta, \end{aligned} \tag{26a}$$

$$|\tilde{a}_{21}| \leqq (\varepsilon_8 + \varepsilon_7)\Delta = \varepsilon_9 \Delta. \tag{26b}$$

This shows the importance of the assumption $|b_{22}| \geqq |a_{22}|$ in order to guarantee the stability of the algorithm. In case $|b_{22}| < |a_{22}|$, Q is constructed to reduce A to triangular form instead of B, and a similar analysis is then possible.

Case II. $d_1 = 2$, $d_2 = 1$. We now have the following configuration (all matrices are real):

$$Q'AZ = Q'\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix} Z = \begin{bmatrix} \hat{a}_{11} & \hat{a}_{12} & \hat{a}_{13} \\ 0 & \hat{a}_{22} & \hat{a}_{23} \\ 0 & \hat{a}_{32} & \hat{a}_{33} \end{bmatrix} = \hat{A}, \tag{27a}$$

$$Q'BZ = Q'\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ 0 & b_{22} & b_{23} \\ 0 & 0 & b_{33} \end{bmatrix} Z = \begin{bmatrix} \hat{b}_{11} & \hat{b}_{12} & \hat{b}_{13} \\ 0 & \hat{b}_{22} & \hat{b}_{23} \\ 0 & 0 & \hat{b}_{33} \end{bmatrix} = \hat{B}. \tag{27b}$$

We assume that $|b_{33}| \geqq |a_{33}|$. If this is not the case, we can always interchange the role of A and B by transforming the first two columns of A and the last two columns of $\hat{A}$ in order to annihilate a_{21} and $\hat{a}_{32}$ and to create b_{21} and $\hat{b}_{32}$.

It follows again from (8), (9) that $\Lambda(b_{33}, a_{33}) = \Lambda(\hat{b}_{11}, \hat{a}_{11})$ if the first column z_1 of Z is an eigenvector of $\lambda B - A$ corresponding to $\Lambda(b_{33}, a_{33})$. Therefore we have (with R any invertible row transformation):

$$R'(a_{33}B - b_{33}A)Z = \left[\begin{array}{c|cc} 0 & & \\ 0 & * & \\ 0 & & \end{array}\right]. \tag{28}$$

Notice that the last row of $H = (a_{33}B - b_{33}A)$ is zero and that we can choose $R \in \mathcal{G}_{12}$ to annihilate the (2, 1) element of H. We then have

$$R'H = \begin{bmatrix} x_2 & x & x \\ 0 & x_1 & x \\ 0 & 0 & 0 \end{bmatrix}. \tag{29}$$

In order to obtain (28) we thus can choose $Z = Z_1 \cdot Z_2$, with $Z_1 \in \mathcal{G}_{23}$ and $Z_2 \in \mathcal{G}_{12}$ annihilating x_1 and x_2, respectively. Q is then constructed to have $\hat{B} = Q'BZ$ in upper triangular form. We therefore take $Q = Q_1 \cdot Q_2$, where $Q_1 \in \mathcal{G}_{23}$ is chosen to annihilate the (3, 2) element created by Z_1 (i.e., $Q_1'BZ_1$ is upper triangular) and where $Q_2 \in \mathcal{G}_{12}$ is chosen to annihilate the (2, 1) element created by Z_2 (i.e., $Q_2'Q_1'BZ_1Z_2$ is upper triangular). $\hat{B}$ now satisfies (27b). Since $|b_{33}|/|a_{33}| = |\hat{b}_{11}|/|\hat{a}_{11}| \geqq 1$, we have $\hat{b}_{11} \neq 0$ and because of (28), $Q'Az_1$ and $Q'Bz_1$ are parallel. This ensures that $\hat{A} = Q'AZ$ also satisfies (27a).

We now prove the numerical stability of the method. Using (13), (14) it can be checked that all the ε_i, $i = 1, \cdots, 4$ below are of the order of the machine accuracy ε. An error analysis of (28), (29) yields

$$\tilde{R}'(a_{33}B - b_{33}A + F)\tilde{Z}_1\tilde{Z}_2 = \begin{bmatrix} 0 & x & x \\ 0 & 0 & x \\ 0 & 0 & 0 \end{bmatrix} \quad \text{for } \|F\|_2 = \varepsilon_1 \|a_{33}B - b_{33}A\|_2, \tag{30}$$

and of the constructed product $Q_2'Q_1'BZ_1Z_2$ yields

$$\tilde{Q}_2'\tilde{Q}_1'(B + E_b)\tilde{Z}_1\tilde{Z}_2 = \begin{bmatrix} \tilde{b}_{11} & \tilde{b}_{12} & \tilde{b}_{13} \\ 0 & \tilde{b}_{22} & \tilde{b}_{23} \\ 0 & 0 & \tilde{b}_{33} \end{bmatrix} \quad \text{for } \|E_b\|_2 = \varepsilon_2 \Delta. \tag{31}$$

We prove that there also exists a backward error E_a such that

$$\tilde{Q}_2'\tilde{Q}_1'(A+E_a)\tilde{Z}_1\tilde{Z}_2=\begin{bmatrix}\tilde{a}_{11} & \tilde{a}_{12} & \tilde{a}_{13}\\ 0 & \tilde{a}_{22} & \tilde{a}_{23}\\ 0 & \tilde{a}_{32} & \tilde{a}_{33}\end{bmatrix} \quad \text{for } \|E_a\|_2=\varepsilon_3\Delta. \tag{32}$$

An error analysis of $Q_2'Q_1'AZ_1Z_2$ yields

$$\tilde{Q}_2'\tilde{Q}_1'(A+E_c)\tilde{Z}_1\tilde{Z}_2=\begin{bmatrix}\tilde{a}_{11} & \tilde{a}_{12} & \tilde{a}_{13}\\ \tilde{a}_{21} & \tilde{a}_{22} & \tilde{a}_{23}\\ \tilde{a}_{31} & \tilde{a}_{32} & \tilde{a}_{33}\end{bmatrix} \quad \text{for } \|E_c\|=\varepsilon_4\Delta. \tag{33}$$

We only have to prove that the elements $\tilde{a}_{i1}$, $i=2,3$ are ε-small, in order to obtain (32) by putting $\tilde{a}_{i1}=0$, $i=2,3$. This is easily proved using similar reasoning to (24)–(26). Here again the assumption $|b_{33}|\geqq|a_{33}|$ is crucial in the proof of backward stability. Therefore, in the case $|b_{33}|<|a_{33}|$, the roles of B and A have to be interchanged.

Case III. $d_1=1$, $d_2=2$. This case is dual to the previous case and can be reduced to it by pertransposition (transposition over the antidiagonal).

Case IV. $d_1=d_2=2$. A detailed configuration of (11) is then

$$Q'AZ=Q'\begin{bmatrix}a_{11} & a_{12} & a_{13} & a_{14}\\ a_{21} & a_{22} & a_{23} & a_{24}\\ 0 & 0 & a_{33} & a_{34}\\ 0 & 0 & a_{43} & a_{44}\end{bmatrix}Z=\begin{bmatrix}\hat{a}_{11} & \hat{a}_{12} & \hat{a}_{13} & \hat{a}_{14}\\ \hat{a}_{21} & \hat{a}_{22} & \hat{a}_{23} & \hat{a}_{24}\\ 0 & 0 & \hat{a}_{33} & \hat{a}_{34}\\ 0 & 0 & \hat{a}_{43} & \hat{a}_{44}\end{bmatrix}=\hat{A}, \tag{34a}$$

$$Q'BZ=Q'\begin{bmatrix}b_{11} & b_{12} & b_{13} & b_{14}\\ 0 & b_{22} & b_{23} & b_{24}\\ 0 & 0 & b_{33} & b_{34}\\ 0 & 0 & 0 & b_{44}\end{bmatrix}Z=\begin{bmatrix}\hat{b}_{11} & \hat{b}_{12} & \hat{b}_{13} & \hat{b}_{14}\\ 0 & \hat{b}_{22} & \hat{b}_{23} & \hat{b}_{24}\\ 0 & 0 & \hat{b}_{33} & \hat{b}_{34}\\ 0 & 0 & 0 & \hat{b}_{44}\end{bmatrix}=\hat{B}, \tag{34b}$$

where all the elements are real and B and $\hat{B}$ are invertible. In order to have $\Lambda(B_{22},A_{22})=\Lambda(\hat{B}_{11},\hat{A}_{11})$ the first two columns of Z must span the deflating subspace of $\lambda B-A$ corresponding to $\Lambda(B_{22},A_{22})$ or, equivalently, the two (complex) eigenvectors corresponding to the eigenvalues λ_2 and $\bar{\lambda}_2$ of $\Lambda(B_{22},A_{22})$. Such a Z also satisfies

$$(\lambda_2 I-B^{-1}A)(\bar{\lambda}_2 I-B^{-1}A)Z=\left[\begin{array}{cc|c}0 & 0 & \\ 0 & 0 & \\ 0 & 0 & *\\ 0 & 0 & \end{array}\right], \tag{35}$$

and could be constructed through (35). Unfortunately, this approach is not recommended from a numerical point of view because of the occurrence of B^{-1} and of the product $(\lambda_2 I-B^{-1}A)(\bar{\lambda}_2 I-B^{-1}A)$. An error analysis of (35) would yield a negligible relative error for this product but not for A and B individually.

A different approach is therefore recommended here, namely the double shift QZ-step. Implicitly, this is a double shift QR-step working on the matrix AB^{-1}, but the actual implementation avoids the construction of AB^{-1} and works instead directly on B and A [13]. For our 4×4 pencil (34) the scheme can be implemented economically with Givens rotations:

- Construct $Q_1\in\mathcal{G}_{23}$ and $Q_2\in\mathcal{G}_{12}$ according to the "double shift technique", and construct $Z_1\in\mathcal{G}_{23}$ and $Z_2\in\mathcal{G}_{12}$ such that $Q_2'Q_1'BZ_1Z_2$ is upper triangular.

$Q_2'Q_1'AZ_1Z_2$ and $Q_2'Q_1'BZ_1Z_2$ then look like

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x_4 & x & x & x \\ x_3 & x_5 & x & x \end{bmatrix} \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{bmatrix}. \tag{36}$$

• Construct $Q_3 \in \mathcal{G}_{34}$, $Q_4 \in \mathcal{G}_{23}$ and $Q_5 \in \mathcal{G}_{34}$, annihilating x_3, x_4 and x_5, respectively in (36). Construct $Z_3 \in \mathcal{G}_{34}$, $Z_4 \in \mathcal{G}_{23}$ and $Z_5 \in \mathcal{G}_{34}$ such that $Q'BZ$, with $Q = Q_1Q_2Q_3Q_4Q_5$ and $Z = Z_1Z_2Z_3Z_4Z_5$, is upper triangular. $Q'AZ$ is now upper Hessenberg and $Q'BZ$ upper triangular. This form is clearly maintained by a QZ-step. In order to obtain (34) we want moreover that $\hat{a}_{32} = 0$ and $\Lambda(\hat{B}_{22}, \hat{A}_{22}) = \Lambda(B_{11}, A_{11})$. According to the properties of the double shift method [13], this will be the case when $\{\lambda_1, \bar{\lambda}_1\} = \Lambda(B_{11}, A_{11})$ is chosen to determine the double shift (i.e., Q_1 and Q_2), and if in addition $a_{32} \neq 0$. Since in (34) the latter is not satisfied, we first perform a QZ-step with random shift such that $a_{32} \neq 0$, and we then perform a second QZ-step with double shift based on $\{\lambda_1, \bar{\lambda}_1\}$.

The numerical properties of the QZ-step are discussed in [13]. The algorithm is backward stable, but under the presence of rounding errors the element $\hat{a}_{32}$ may not be negligible. Several QZ-steps with double shift $\{\lambda_1, \bar{\lambda}_1\}$ are then performed, and $\hat{a}_{32}$ is shown to converge very fast to zero [13]. Only in pathological cases is more than one step required to obtain $|\hat{a}_{32}| \leqq \varepsilon\Delta$.

Operation count. The combination of a pair of left and right Givens transformations Q_i, Z_i, requires approximately $12n$ operations (1 operation = 1 addition + 1 multiplication). The number of operations for the different cases is then (for Case IV we assume only 2 QZ-steps are needed):

Case I: $12n$.

Case II and III: $32n$ (average).

Case IV: $120n$.

Since Cases II and III correspond to two interchanges of eigenvalues and Case IV to four interchanges, we finally have an average of $20n$ operations for interchanging two adjacent eigenvalues.

When a deflating subspace with a specified spectrum $\{\mu_1, \cdots, \mu_l\}$ has to be computed and a QZ-decomposition is already available, then at most $l \cdot (n-l) \leqq n^2/4$ such interchanges are required (namely when all μ_i, $i = 1, \cdots, l$ are in the bottom right corner). A reasonable estimate is thus $5n^3$ operations for computing a specific deflating subspace from a QZ-decomposition, while the latter requires approximately $25n^3$ operations. In order to obtain all possible orderings of eigenvalues in the QZ decomposition, and thus all possible deflating subspaces (if no eigenvalues are repeated), $n!$ such interchanges are required [9]. That is to be expected since it is a combinatorial problem.

3. Riccati equations. In this section we apply the above ideas to the solution of certain Riccati equations arising in linear system theory. We first briefly restate the four problems we will focus on, and we refer to the literature for a more complete discussion. We will everywhere assume that the matrices involved are real, since this is usually the case in practice. Extensions to the complex case are trivial.

Problem I. *Optimal control: continuous time case* [11][12][24]. Given the stabilizable system

$$\dot{x}(t) = A_{nn}x(t) + B_{nm}u(t), \tag{37}$$

find the control $u(t) = -Kx(t)$ minimizing the functional

$$J = \int_0^\infty [x'(t)Q_{nn}x(t) + u'(t)R_{mm}u(t)]\,dt, \tag{38}$$

where (A, Q) is detectable, $Q \geqq 0$ and $R \geqq 0$. When R is invertible this problem reduces to the computation of the unique nonnegative definite solution P of the algebraic Riccati equation

$$Q + A'P + PA - PBR^{-1}B'P = 0. \tag{39}$$

K is then equal to $R^{-1}B'P$. Equivalently [12], one can compute the invariant subspace $\mathscr{X}_s$ of the matrix

$$H = \begin{bmatrix} A & -BR^{-1}B' \\ -Q & -A' \end{bmatrix}, \tag{40}$$

where $\Lambda(H)|_{\mathscr{X}_s}$ contains all the stable eigenvalues (i.e., Re $(\lambda) < 0$) of H. If $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ is a basis for this subspace, then $P = X_2X_1^{-1}$.

Problem II. *Optimal control problem*: *discrete time case* [5][7][15]. Given the stabilizable system

$$x_{i+1} = F_{nn}x_i + G_{nm}u_i, \tag{41}$$

find the control $u_i = -Kx_i$ minimizing the functional

$$J = \sum_{i=0}^{\infty} [x_i'Q_{nn}x_i + u_i'R_{mm}u_i], \tag{42}$$

where (F, Q) is detectable, $Q \geqq 0$ and $R \geqq 0$.

When R is invertible [7], this problem can again be converted to the computation of the unique nonnegative definite solution P of the (discrete time) algebraic Riccati equation

$$P = F'PF - F'PG(R + G'PG)^{-1}G'PF + Q. \tag{43}$$

K is then equal to $(R + G'PG)^{-1}G'P$. This is also equivalent to solving for the "stable" deflating subspace $\mathscr{X}_s$ of the pencil [5][15]

$$\lambda \begin{bmatrix} I & GR^{-1}G' \\ 0 & F' \end{bmatrix} - \begin{bmatrix} F & 0 \\ -Q & I \end{bmatrix}, \tag{44}$$

where this time the stable eigenvalues are those inside the unit circle. If $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ is a basis for $\mathscr{X}_s$, then $P = X_2X_1^{-1}$.

Problem III. *Spectral factorization*: *continuous time case* [2]. Given an $m \times m$ "positive real" rational matrix $Z(s)$, i.e.,

$$Z(s) \text{ analytic} \quad \text{and} \quad Z(s) + Z^*(s) \geqq 0 \quad \text{in Re } (s) > 0, \tag{45}$$

find a "spectral factorization"

$$Z(s) + Z'(-s) = R(s) \cdot R'(-s), \tag{46}$$

where $R(s)$ has only stable poles and zeros (i.e., Re $(s) < 0$).

When $Z(s)$ is given by a minimal realization $C(sI_n - A)^{-1}B + D$ and $(D + D')$ is invertible, then this problem reduces to the computation of the unique positive definite

solution of the algebraic Riccati equation [2]

$$B(D+D')^{-1}B'+P[A-B(D+D')^{-1}C]' \\ +[A-B(D+D')^{-1}C]P+PC'(D+D')^{-1}CP=0. \tag{47}$$

This is again equivalent to the computation of the stable invariant subspace $\mathscr{X}_s$ of the matrix

$$H=\begin{bmatrix} A-B(D+D')^{-1}C & B(D+D')^{-1}B' \\ -C'(D+D')^{-1}C & -[A-B(D+D')^{-1}C]' \end{bmatrix}. \tag{48}$$

Problem IV. *Spectral factorization*: *discrete time case* [1][4]. Given an $m\times m$ "positive real" discrete time matrix $Z(z)$, i.e.,

$$Z(z) \text{ analytic} \quad \text{and} \quad Z(z)+Z^*(z)\geqq 0 \quad \text{for } |z|>1, \tag{49}$$

find a spectral factorization

$$Z(z)+Z'(z^{-1})=R(z)\cdot R'(z^{-1}) \tag{50}$$

where $R(z)$ has only stable poles and zeros (i.e., inside the unit circle). Again, when $Z(z)$ is given by a minimal realization $H(zI_n-F)^{-1}G+J$ and $(J+J')$ is invertible, the problem can be reduced to the computation of the unique positive definite solution P of the (discrete time) Riccati equation [4]

$$P=FPF'+(G-FPH')(J+J'-HPH')^{-1}(G'-HPF'). \tag{51}$$

In analogy to (43), (44), one can prove that this is equivalent to computing the stable deflating subspace $\mathscr{X}_s$ of

$$\lambda\begin{bmatrix} I & -G(J+J')^{-1}G' \\ 0 & F'-H'(J+J')^{-1}G' \end{bmatrix}-\begin{bmatrix} F-G(J+J')^{-1}H & 0 \\ -H'(J+J')^{-1}H & I \end{bmatrix}. \tag{52}$$

This, however, was not found in the literature.

Note that in order to be able to write down the Riccati equations, we need certain matrices to be invertible. This also holds for the equivalent SEP's and GEP's, since they are derived from the Riccati equations. Yet, if the matrices to be inverted happen to be badly conditioned, each of these approaches may encounter serious numerical difficulties when computing these inverses. We now present a way to circumvent this by an embedding technique. If D is invertible in the pencil

$$\begin{bmatrix} \lambda E-A & B \\ \underbrace{\lambda F-C}_{p} & \underbrace{D}_{m} \end{bmatrix}\begin{matrix}\}p \\ \}m\end{matrix} \tag{53}$$

then

$$\begin{bmatrix} I & -BD^{-1} \\ 0 & I \end{bmatrix}\cdot\begin{bmatrix} \lambda E-A & B \\ \lambda F-C & D \end{bmatrix}=\begin{bmatrix} \lambda(E-BD^{-1}F)-(A-BD^{-1}C) & 0 \\ \lambda F-C & D \end{bmatrix}. \tag{54}$$

Let U be an orthogonal transformation reducing $\left[\begin{smallmatrix} B \\ D \end{smallmatrix}\right]$ to $\left[\begin{smallmatrix} 0 \\ \tilde{D} \end{smallmatrix}\right]$ with $\tilde{D}$ $m\times m$ and invertible. Partition U conformably with (53); then we have

$$\begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}\cdot\begin{bmatrix} \lambda E-A & B \\ \lambda F-C & D \end{bmatrix}=\begin{bmatrix} \lambda\tilde{E}-\tilde{A} & 0 \\ * & \tilde{D} \end{bmatrix}. \tag{55}$$

Since the rows of $[U_{11}|U_{12}]$ and $[I|-BD^{-1}]$ both are a basis for the left null space of $\left[\begin{smallmatrix}B\\D\end{smallmatrix}\right]$, they are related by an invertible row transformation which clearly must be U_{11}:

$$U_{11}[I|-BD^{-1}]=[U_{11}|U_{12}] \tag{56}$$

From (54) and (55) it then follows that

$$U_{11}[\lambda(E-BD^{-1}F)-(A-BD^{-1}C)]=\lambda\tilde{E}-\tilde{A}. \tag{57}$$

Therefore the deflating subspaces of $\lambda\tilde{E}-\tilde{A}$ and of $\lambda(E-BD^{-1}F)-(A-BD^{-1}C)$ are the same. According to (7), deflating subspaces of a regular pencil are indeed not affected by an invertible row transformation on the pencil. This technique was also applied in [22] (with $E=I$ and $F=0$) for developing a stable way to compute the deflating subspaces of $\lambda I-(A-BD^{-1}C)$ or, in other words, the invariant subspaces of $A-BD^{-1}C$. This can now be applied to the above four problems. In each of them the pencil (53) takes the form (we always have $p=2n$)

Problem I:

$$\lambda\begin{bmatrix}I&0&0\\0&I&0\\0&0&0\end{bmatrix}-\begin{bmatrix}A&0&B\\-Q&-A'&0\\0&B'&R\end{bmatrix}. \tag{58}$$

Problem II:

$$\lambda\begin{bmatrix}I&0&0\\0&F'&0\\0&G'&0\end{bmatrix}-\begin{bmatrix}F&0&-G\\-Q&I&0\\0&0&R\end{bmatrix}. \tag{59}$$

Problem III:

$$\lambda\begin{bmatrix}I&0&0\\0&I&0\\0&0&0\end{bmatrix}-\begin{bmatrix}A&0&B\\0&-A'&C'\\C&-B'&D+D'\end{bmatrix}. \tag{60}$$

Problem IV:

$$\lambda\begin{bmatrix}I&0&0\\0&F'&0\\0&G'&0\end{bmatrix}-\begin{bmatrix}F&0&-G\\0&I&-H'\\H&0&-(J+J')\end{bmatrix}. \tag{61}$$

For each of these pencils a $2n\times 2n$ pencil $\lambda\tilde{E}-\tilde{A}$ can thus be derived via (55), and its stable deflating subspace $\mathscr{X}_s$ is the one required in the above four problems. This procedure does not involve the inversion of a possibly ill-conditioned matrix. Only orthogonal transformations are used as well in the construction of $\lambda\tilde{E}-\tilde{A}$ as in the computation of the deflating subspace $\mathscr{X}_s$. This guarantees the numerical stability of the method. Unfortunately this is not completely satisfactory yet, since the performed errors do not necessarily respect the structure of the pencils (58)–(61). An (unsuccessful) attempt to restrict the orthogonal transformations to those respecting the structure of the matrices they act upon can be found in the literature for Problems I and III, but in the formulation (40) and (48), respectively [14].

An important remark here is that in the new formulation (58)–(61) no inverses occur any more, and that perhaps this new formulation also gives the correct answer when these inverses do not exist. This would follow from limiting arguments if both the exact solution of the problem and the computed solution from the GEP's (58)–(61) are

continuous. This is true for the eigenvalue problem if the spectrum $\Lambda(\tilde{E}, \tilde{A})|_{\mathscr{X}_s}$ is separated from the rest of the spectrum $\lambda\tilde{E} - \tilde{A}$ [20], and this holds under some weak assumptions in each problem (stabilizability, detectability, positive realness). The continuity of the solution $R(s)$ of Problem III is discussed in [1, p. 243]. It also holds for the more general "minimal factorization problem" [3] for which the above embedding technique was originally derived [22]. It is therefore reasonable to assume that it also holds for the other three problems. This is still under current investigation.

During the elaboration of this research, the author's attention was drawn to the work of A. Emami-Naeini and G. Franklin [6]. Via an independent approach they arrive at the same form (59). No proof is provided, though, that the method also works for singular R. The authors of [6] are presently working on that problem.

4. Numerical examples. In this section we give two examples illustrating the reordering of eigenvalues in order to compute a certain deflating subspace with prescribed spectrum. We use a PDP 11-34 computer with double precision. The machine precision is then $\varepsilon \cong 1.5\ 10^{-17}$. Two routines are used for the reordering of the Schur form [25]. EXCHQZ exchanges two adjacent blocks in a real Schur form and ORDER uses this routine to reorder all the eigenvalues inside the unit circle to the top or bottom of the real Schur form, depending on the value of a parameter IFIRST. This last routine is easily adapted for any region which is symmetric with respect to the real axis. This condition is necessary because the pencils considered are real and complex conjugate eigenvalues need thus to stay together in the real Schur form.

Example I.

$$A - \lambda B = \left[\begin{array}{c|ccc|cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & .3 & .2 & 4 & 6 & 0 & 0 & 0 \\ 0 & -.2 & .3 & 0 & 0 & 0 & .5 & 0 \\ \hline 0 & 0 & 0 & .5 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 4 & 2.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & -10 & 4 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \end{array}\right] - \lambda \left[\begin{array}{c|ccc|cccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{62}$$

The first four eigenvalues $\{0, .3 - j.2, .3 + j.2, .5\}$ are outside the unit circle. The last four ones $\{\infty, 4 - j5, 4 + j5, 2\}$ are outside the unit circle. Calling ORDER with IFIRST = 1 interchanges the order of these sets of eigenvalues. The first four columns of the transformation Z required for this then span the unstable subspace $\mathscr{X}_u$ of $A - \lambda B$; see Table 1.

When again calling ORDER but now with IFIRST = −1, we retrieve the ordering of $A - \lambda B$ and the four first columns of the updated Z look like Table 2.

This is ε-close to the real stable deflating subspace $\mathscr{X}_s$ of $A - \lambda B$, which is spanned by $\left[\begin{smallmatrix} I_4 \\ 0_4 \end{smallmatrix}\right]$. This result is to be expected because of the numerical stability of our method and because the space $\mathscr{X}_s$ of $A - \lambda B$ is well-conditioned. When the gap between the spectra $\Lambda(B, A)|_{\mathscr{X}_s}$ and $\Lambda(B, A)|_{\mathscr{X}_u}$ is large, both spaces $\mathscr{X}_s$ and $\mathscr{X}_u$ are indeed well-conditioned (see [20]).

Example II. Consider Problem II with

$$F = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = [0]. \tag{63}$$

TABLE 1

0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.4472135954999579*d*	00
0.0000000000000000*d*	00	0.1074176896329408*d*	00	0.4536669603618238*d*	−01	0.0000000000000000*d*	00
0.0000000000000000*d*	00	−0.6787906375520325*d*	−01	−0.1235432403982217*d*	00	0.0000000000000000*d*	00
0.0000000000000000*d*	00	0.1029985626780152*d*	00	−0.1992532259759830*d*	00	0.0000000000000000*d*	00
0.1000000000000000*d*	01	0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.0000000000000000*d*	00
0.0000000000000000*d*	00	−0.1485334378807009*d*	00	−0.9610814937405690*d*	00	0.0000000000000000*d*	00
0.0000000000000000*d*	00	−0.9752861049841944*d*	00	0.1389224422842769*d*	00	0.0000000000000000*d*	00
0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.8944271909999159*d*	00

TABLE 2

0.1000000000000000*d*	01	0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.0000000000000000*d*	00
0.0000000000000000*d*	00	−0.9682688602071642*d*	00	−0.2499108127975237*d*	00	0.3620467925763759*d*	−16
0.0000000000000000*d*	00	−0.2499108127975237*d*	00	0.9682688602071642*d*	00	−0.1119431427711831*d*	−15
0.0000000000000000*d*	00	0.2168404344971009*d*	−17	0.1154675313697062*d*	−15	0.9999999999999999*d*	00
0.0000000000000000*d*	00	−0.3134519117986896*d*	−17	0.1040834085586084*d*	−16	0.4491253335510017*d*	−16
0.0000000000000000*d*	00	0.1040834085586084*d*	−16	0.1543903893619358*d*	−15	−0.5898059818321144*d*	−16
0.0000000000000000*d*	00	−0.3469446951953614*d*	−17	−0.6357761539454998*d*	−15	0.0000000000000000*d*	00
0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.0000000000000000*d*	00	0.0000000000000000*d*	00

The pencil (59) then looks like

$$\lambda \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{64}$$

An orthogonal row transformation can then be constructed in order to construct a deflated pencil $\lambda\tilde{E} - \tilde{A}$ following (55):

$$\lambda \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{65}$$

The QZ-algorithm permutes the two last columns of (65) to obtain the real Schur form

$$\lambda \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{66}$$

which displays the eigenvalues $\{\infty, \infty, 0, 0\}$. In order to obtain the stable subspace $\mathscr{X}_s$ of $\lambda\tilde{E} - \tilde{A}$ we reorder these eigenvalues and obtain, as a basis for $\mathscr{X}_s$,

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} 0 & -\sqrt{2}/2 \\ -\sqrt{2}/2 & 0 \\ 0 & -\sqrt{2}/2 \\ -\sqrt{2}/2 & 0 \end{bmatrix} + O(\varepsilon). \tag{67}$$

We then find, up to machine accuracy, the answer $P = I$. One can check that this is the correct answer to Problem II by using another method [7]. This example illustrates that the embedding technique gives a correct result even when R is singular. Moreover, the problem is perfectly well-conditioned as well for the construction of $\lambda\tilde{E} - \tilde{A}$, as for the computation of $\mathscr{X}_s$ and P.

We finally want to draw attention to the fact that the number of operations required for the construction of $\lambda\tilde{E} - \tilde{A}$ from the pencils (58)–(61) is comparable to the amount of work required to construct the pencils (40), (44), (48), (52). From then on, the new approach takes the same amount of computations for Problems II and IV and only slightly more (less than the double) for Problems I and III. The stability of the method and its better conditioning therefore make this new approach particularly attractive.

Acknowledgments. I want to thank A. Emami-Naeini, G. Franklin and A. Laub for drawing my attention to this problem and for several helpful discussions. A. Emami-Naeini also suggested Example II.

REFERENCES

[1] B. ANDERSON AND J. MOORE, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

[2] B. ANDERSON AND S. VONGPANITLERD, *Network Analysis and Synthesis. A Modern Systems Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1972.

[3] H. BART, I. GOHBERG, M. KAASHOEK AND P. VAN DOOREN, *Factorizations of transfer functions*, SIAM J. Control. Optim., 18 (1980), pp. 675–696.

[4] M. DENHAM, *On the factorization of discrete-time rational spectral density matrices*, IEEE Trans. Automat. Contr., AC-20 (1975), pp. 535–537.

[5] A. EMAMI-NAEINI AND G. FRANKLIN, *Design of steady state quadratic loss optimal digital controls for systems with a singular system matrix*, in Proceedings 13th Asilomar Conference on Circ. Syst. & Comp., Nov., 1979, pp. 370–374.

[6] ———, *Comments on "The numerical solution of the discrete time algebraic Riccati equation"*, IEEE Trans. Automat. Contr., AC-25 (1980), pp. 1015–1016.

[7] G. FRANKLIN AND J. POWELL, *Digital Control of Dynamic Systems*, Addison-Wesley, Reading, MA, 1979.

[8] G. GOLUB AND J. WILKINSON, *Ill-conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Rev., 18 (1976), pp. 578–619.

[9] S. JOHNSON, *Generation of permutations by adjacent transposition*, Math. Comp., 17 (1963), pp. 282–285.

[10] T. KAILATH, *Linear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[11] H. KWAKERNAAK AND R. SIVAN, *Linear Optimal Control Systems*, Wiley-Interscience, New York, 1972.

[12] A. LAUB, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Contr., AC-24 (1979), pp. 913–921.

[13] C. MOLER AND G. STEWART, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241–256.

[14] C. PAIGE AND C. VAN LOAN, *A Hamiltonian-Schur decomposition*, Internal Report, Department of Computer Science, Cornell University, Ithaca, New York, 1979.

[15] T. PAPPAS, A. LAUB AND N. SANDELL JR., *On the numerical solution of the discrete time algebraic Riccati equation*, IEEE Trans. Automat. Contr., AC-25 (1980), pp. 631–641.

[16] G. PETERS AND J. WILKINSON, *$Ax = \lambda Bx$ and the generalized eigenproblem*, SIAM J. Numer. Anal., 7 (1970), pp. 479–492.

[17] A. RUHE, *An algorithm for numerical determination of the structure of a general matrix*, BIT, 10 (1970), pp. 196–216.

[18] G. STEWART, *On the sensitivity of the eigenvalue problem $Ax = \lambda Bx$*, SIAM J. Numer. Anal., 9 (1972), pp. 669–686.

[19] ———, *Introduction to Matrix Computation*, Academic Press, New York, 1973.

[20] ———, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[21] ———, *Algorithm 506: HQR3 and EXCHNG. Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix*, ACM TOMS, 2 (1976), pp. 275–280.

[22] P. VAN DOOREN, *The generalized eigenstructure problem in linear system theory*, IEEE Trans. Automat. Contr., AC-26 (1981), to appear.

[23] J. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

[24] M. WONHAM, *On a matrix Riccati equation of stochastic control*, SIAM J. Control, 6 (1968), pp. 681–697.

[25] P. VAN DOOREN, *A generalized eigenvalue approach for solving Riccati equations*, Internal Report NA-80-02, Department of Computer Science, Stanford University, Stanford, CA, 1980.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0002 $01.00/0

ON THE IMPLICIT DEFLATION OF NEARLY SINGULAR SYSTEMS OF LINEAR EQUATIONS*

G. W. STEWART†

Abstract. In solving the linear system $Ax = b$ when A is nearly singular, it is often appropriate to work in a coordinate system in which the singularity can be easily removed. When A is large and sparse, an explicit transformation of the system cannot be made, since it will destroy the sparsity of the system. In this paper, an algorithm similar to the method of iterative refinement for linear systems is given and its numerical properties described.

Key words. rank degeneracy, singular value, sparse matrix, iterative refinement

1. Introduction. In this paper we shall consider the problem of solving the system of linear equations

$$Ax = b \tag{1.1}$$

when the matrix A is nearly singular. When A is or order n and rank $(A) = n-1$, this problem admits a simple mathematical solution. Let u and v be left and right null vectors of A (i.e., $A^T u = Av = 0$) of Euclidean norm unity, and let U and V be orthogonal matrices whose last columns are the null vectors u and v. Then it is easily verified that

$$U^T A V = \begin{bmatrix} A_{11} & 0 \\ 0 & 0 \end{bmatrix},$$

where A_{11} is a nonsingular matrix or order $n-1$. Write

$$V^T x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{1.2}$$

and

$$U^T b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \tag{1.3}$$

where x_1 and b_1 are of order $n-1$. For the system (1.1) to be consistent, the vector b_2 must be zero, in which case the vector

$$\tilde{x} = V \begin{bmatrix} A_{11}^{-1} b_1 \\ 0 \end{bmatrix}$$

is a solution of (1.1). In fact, it is the solution of smallest Euclidean norm.

In practice, the matrix A will seldom be exactly singular and the vectors u and v will be only approximate null vectors. This means that $U^T A V$ will have the form

$$U^T A V = \begin{bmatrix} A_{11} & a_{12} \\ a_{21}^T & a_{22} \end{bmatrix}, \tag{1.4}$$

* Received by the editors July 23, 1980, and in final form January 9, 1981.

† Department of Computer Science and Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742. This research was supported in part by the U.S. Office of Naval Research under contract W00014-76-C-0391.

where a_{12}, a_{21} and a_{22} are small. In many applications (for example, in the numerical treatment of bifurcation problems [4] or of nearly decomposable Markov chains [6]), it is appropriate to ignore these small quantities and compute the approximate solution $\tilde{x}$ given above. We shall call the results of this procedure the *deflated solution* of the system (1.1).

As long as A is a dense matrix whose elements can be contained in the high speed storage of a computer, the process just outlined presents no particular difficulties. When A is large and sparse, however, the process cannot be carried out, since the matrix A_{11} of (1.4) will in general no longer be sparse. Using sparse matrix techniques (e.g., see [2]), it may be possible to compute a solution of (1.1). Thus we may raise the following question: given approximate left and right null vectors u and v of A, can we use the ability to solve (1.1) to obtain the deflated solution of (1.1)? The answer is yes—usually; the rest of this paper is devoted to describing an algorithm for computing the deflated solution.

In what follows we shall use the vector 2-norm defined by $\|x\|^2 = x^T x$ and the subordinate matrix norm defined by

$$\|A\| = \sup_{\|x\|=1} \|Ax\|.$$

2. The algorithm. It will be convenient to phrase the algorithm in terms of projectors associated with the vectors u and v. Let

$$P_u = I - uu^T, \qquad P_v = I - vv^T.$$

The matrix P_u (P_v) projects a vector x onto the space orthogonal to the vector $u(v)$. It is easy to verify that the deflated solution $\tilde{x}$ of (1.1) is the unique vector satisfying

$$P_u A P_v \tilde{x} = P_u b \tag{2.1a}$$

and

$$P_v \tilde{x} = \tilde{x}. \tag{2.1b}$$

We have indicated that when A is large and sparse it may be impossible to solve (1.1) directly. However, under appropriate conditions we can solve it approximately. This suggests that we use some form of the method of iterative refinement for linear systems [3], [5], [6], a suggestion that leads directly to the following algorithm:

1. $x = 0$
2. loop

(2.2)

1. $r = P_u(b - Ax)$
2. Solve $Ad = r$
3. $x \leftarrow x + P_v d$

The conditions under which this algorithm works are stated in the following theorem.

THEOREM 2.1. *Let $U^T A V$ have the form* (1.4). *Suppose that A_{11} is nonsingular and that*

$$\delta = a_{22} - a_{21}^T A_{11}^{-1} a_{12} \neq 0. \tag{2.3}$$

Then A is nonsingular, so that algorithm (2.2) *is well defined. Let*

$$F \equiv \frac{A_{11}^{-1} a_{12} a_{21}^T}{\delta}. \tag{2.4}$$

If $\|F\| < 1$, then the sequence generated by algorithm (2.2) *converges to $\tilde{x}$.*

Proof. If we make the substitutions $x \leftarrow V^T x$, $b \leftarrow V^T b$ and $A \leftarrow U^T A V$, then x, b and A assume the same form as the right-hand sides of (1.2), (1.3) and (1.4). Moreover, P_u and P_v become diag $(I_{n-1}\ 0)$, so that operating on a vector x by either P_u or P_v amounts to setting the last component of x to zero.

To prove the first assertion of the theorem, note that

$$\begin{bmatrix} A_{11} & a_{12} \\ a_{21}^T & a_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ a_{21}^T A_{11}^{-1} & 1 \end{bmatrix} \begin{bmatrix} A_{11} & a_{12} \\ 0 & a_{22} - a_{21}^T A_{11}^{-1} a_{12} \end{bmatrix}. \tag{2.5}$$

If $\delta \neq 0$, then (2.5) exhibits A as the product of two nonsingular matrices.

To prove the second assertion of the theorem, note that any vector x generated by (2.2) satisfies $P_v x = x$, or equivalently,

$$x = \begin{bmatrix} x_1 \\ 0 \end{bmatrix}.$$

For such an x we have

$$r = \begin{bmatrix} b_1 - A_{11} x_1 \\ 0 \end{bmatrix}.$$

From (2.5) it is easily seen that

$$d = \begin{bmatrix} (I+F) A_{11}^{-1} (b_1 - A_{11} x_1) \\ -\delta^{-1} a_{21}^T A_{11}^{-1} (b_1 - A_{11} x_1) \end{bmatrix}.$$

Hence, if we denote by x^* the vector produced at step 2.3 of (2.2), we have

$$x_1^* = x_1 + (I+F)(\tilde{x}_1 - x_1), \qquad x_2^* = 0.$$

Hence

$$\tilde{x}_1 - x_1^* = F(\tilde{x}_1 - x_1),$$

or

$$\|\tilde{x}_1 - x_1^*\| \leqq \|F\| \|\tilde{x}_1 - x_1\|.$$

Since $\|F\| < 1$, the sequence generated by (2.2) must converge to $\tilde{x}$.

It is important to have some idea of when the condition $\|F\| < 1$ is satisfied. We shall now show that what is required is that u and v be good approximations to the singular vectors of A corresponding to the smallest singular value. Specifically, by appealing to the singular value decomposition [5, Chapt. 7], we may assume without loss of generality that A has the form

$$A = \begin{bmatrix} \Sigma & 0 \\ 0 & \sigma \end{bmatrix},$$

where

$$\Sigma = \text{diag}\,(\sigma_1, \sigma_2, \cdots, \sigma_{n-1})$$

and $\sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_{n-1} \geqq \sigma > 0$. The optimal approximate null vectors of A (in the

sense that $\|A_v\|$ and $\|A^T u\|$ are minimized) are the unit vectors $\mathbf{1}_n$ and $\mathbf{1}_n$. If we accept approximations

$$u^T = (e_u^T \quad 1), \qquad v^T = (e_v^T \quad 1),$$

then U and V are approximated up to second order terms in e_u and e_v by

$$U \cong \begin{bmatrix} I & e_u \\ -e_u^T & 1 \end{bmatrix}, \qquad V \cong \begin{bmatrix} I & e_v \\ -e_v^T & 1 \end{bmatrix}.$$

It is then easily verified that

$$U^T A V \cong \begin{bmatrix} \Sigma + \sigma e_u e_v^T & \Sigma e_v - \sigma e_u \\ e_u^T \Sigma - \sigma e_v^T & e_u^T \Sigma e_v^T - \sigma \end{bmatrix}.$$

Thus, if we set

$$\varepsilon = \max\{\|e_u\|, \|e_v\|\} < 1, \tag{2.6}$$

we have

$$\begin{aligned} \|A_{11}\| &\lessapprox \sigma_1 + \varepsilon^2 \sigma, \\ \|a_{12}\|, \|a_{21}\| &\lessapprox (\sigma_1 + \sigma)\varepsilon \lessapprox 2\|A_{11}\|\varepsilon. \end{aligned} \tag{2.7}$$

After some manipulation, we get

$$\|F\| \lessapprox \frac{4\varepsilon^2 \|A_{11}\| \kappa(A_{11})}{|a_{22}| - 4\varepsilon^2 \|A_{11}\| \kappa(A_{11})}$$

where $\kappa(A_{11}) = \|A_{11}\| \|A_{11}^{-1}\|$ is the condition number of A_{11}. Hence, if

$$\varepsilon \lessapprox \sqrt{\frac{1}{8} \kappa(A_{11})^{-1} \frac{|a_{22}|}{\|A_{11}\|}}, \tag{2.8}$$

then $\|F\| < 1$.

The condition (2.8) does not put severe restrictions on the accuracy of u and v. For example, if $|a_{22}|/\|A_{11}\| = 10^{-5}$ and $\kappa(A_{11}) = 10^2$, then (2.8) requires $\varepsilon < 1.12 \cdot 10^{-4}$ for convergence. Actually, in this case it is more realistic to replace the factor $\frac{1}{8}$ by $\frac{1}{2}$ to give $\varepsilon < 2.24 \cdot 10^{-4}$ (cf. (2.7)). In any case, we only require four-digit approximations to the minimal singular vectors to get convergence.

3. Practicalities. In this section we shall discuss some of the practical problems associated with the algorithm described in the last section. The first problem is, of course, to obtain approximate null vectors u and v. For the sparse systems contemplated in this paper, the vectors can be obtained by a variant of the inverse power method [8]. In brief, one starts with a vector u_0 and iterates according to the formulas

1. $A\tilde{v}_{i+1} = u_i$,
2. $v_{i+1} = \tilde{v}_i / \|\tilde{v}_i\|$,
3. $A^T \tilde{u}_{i+1} = v_{i+1}$,
4. $u_{i+1} = \tilde{u}_{i+1} / \|\tilde{u}_{i+1}\|$.

The iteration requires nothing more than that one be able to solve linear systems involving A and A^T. If A is within rounding error of a singular matrix, then at most two iterations will usually suffice to produce the vectors u and v. As A moves away from

singularity, more iterations will be required. In any case, the final result will be a pair u and v which satisfy (2.6), where ε is of the order of the rounding unit ε_M.

The computation of the projections required by the algorithm can be accomplished directly from the definitions of P_u and P_v. Alternatively, one can use Householder transformations as described in [5, Chapt. 6]. The latter method has the advantage that no matter how small $P_u x$ is, it will be almost exactly orthogonal to u.

In the first step of the iteration, the vector x is zero so that $r = P_u b$. If a_{22} is large enough, the solution produced by this step will be as accurate as the condition of $\tilde{x}$ warrants. If $a_{22} \leqq \varepsilon_M$, however, more iterations may be required, and it is because of this possibility that we have phrased our algorithm in the iterative form of (2.2). Seldom should more than two or three iterations be required.

We shall not give a formal analysis of the effects of rounding error on (2.2), since such an analysis closely parallels the treatments in [4] and [6] for iterative refinement. In summary, there are two factors affecting the performance: the errors made in solving the system $Ad = r$ and the errors made in calculating $b - Ax$ and the various projections. As far as the first is concerned, it follows from standard rounding error analyses [1], [7] that the computed vector d satisfies $(A + E)d = r$, where E is a small matrix whose size depends on the details of the way in which the solution was obtained. The presence of this matrix E may slow the convergence, but it will not affect the final accuracy of the results. In sparse matrix applications where some form of threshold pivoting has been used, E can quite easily be large enough to slow convergence, another reason for the iterative form of (2.2).

The accuracy to which $b - Ax$ and the projections are calculated limits the accuracy attainable in x. If these are calculated in single precision, then $\tilde{x}$ will be determined to a relative accuracy of order $\varepsilon_M \|A_{11}^{-1}\|$. To get more accurate answers, the vectors $b - Ax$ and the projections must be computed to higher precision. However, we note that doing this will in general not make sense unless u and v have also been determined to high precision.

REFERENCES

[1] J. R. BUNCH, *Analysis of sparse elimination*, SIAM J. Numer. Anal., 11 (1974), pp. 847–873.

[2] IAIN DUFF AND G. W. STEWART, eds., *Sparse Matrix Proceedings* 1978, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[3] C. B. MOLER, *Iterative refinement in floating point*, J. Assoc., Comput. Mach., 14 (1967), pp. 316–321.

[4] W. R. RHEINBOLDT, *Numerical methods for a class of finite dimensional bifurcation problems*, SIAM J. Numer. Anal., 15 (1978), pp. 1–11.

[5] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[6] ———, *Computable error bounds for aggregated Markov chains*, University of Maryland Computer Science Center Technical Report 901, 1980.

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

[8] ———, *The Algebraic Eigenvalue Problem*, Oxford University Press (Clarendon), London and New York, 1965.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0003 $01.00/0

BALANCING THE GENERALIZED EIGENVALUE PROBLEM*

ROBERT C. WARD†

Abstract. An algorithm is presented for balancing the A and B matrices prior to computing the eigensystem of the generalized eigenvalue problem $Ax = \lambda Bx$. The three-step algorithm is specifically designed to precede the QZ-type algorithms, but improved performance is expected from most eigensystem solvers. Permutations and two-sided diagonal transformations are applied to A and B to produce matrices with certain desirable properties. Test cases are presented to illustrate the improved accuracy of the computed eigenvalues.

Key words. eigenvalues, balancing, scaling, diagonal transformation, generalized eigenvalue, graded matrix, QZ algorithm

1. Introduction. Generalized eigenvalue problems consist of determining scalars λ and corresponding $n \times 1$ nonzero vectors x such that the equation

$$Ax = \lambda Bx \tag{1}$$

is satisfied, where A and B are given $n \times n$ matrices. These problems arise in such diverse areas as natural frequency and buckling analysis of structures, the application of molecular orbital theory in quantum chemistry, the investigation of regulation and servomechanism design by time-invariant control systems, and economic analysis of energy decisions.

The only efficient, stable, general purpose algorithms for solving generalized eigenvalue problems are the QZ-type algorithms introduced by Moler and Stewart [5] and extended by Ward [11]. These algorithms reduce A and B simultaneously to upper triangular matrices T_A and T_B such that

$$T_A = QAZ \quad \text{and} \quad T_B = QBZ,$$

where Q and Z are products of elementary unitary transformations. The problem $T_A w = \lambda T_B w$ is equivalent to $Ax = \lambda Bx$ in that they have the same eigenvalues and their eigenvectors are related by the equation $x = Zw$. If we denote the ith diagonal of T_A by α_i and the ith diagonal of T_B by β_i, the eigenvalues are given by α_i/β_i if there are no zero values of β_i. A zero value of β_i with a nonzero value of α_i merely implies that the corresponding eigenvalue is infinite. A zero value for both β_i and α_i implies that $\det(A - \lambda B) \equiv 0$; thus, for any value of λ there exists a nonzero vector x such that $Ax = \lambda Bx$. We will formally express the eigenvalues as α_i/β_i, always implying the above consequences when β_i is zero.

The QZ-type algorithms applied to generalized eigenvalue problems having a wide range in the magnitude of the elements in the A or B matrix frequently produce α's and β's which also have widely varying magnitudes. When the smaller α's and β's are primarily determined by the smaller elements in A and B, respectively, such as in graded matrices, those α's and β's normally have large relative errors independent of their sensitivities, as a result of interaction between the smaller and larger matrix elements. Similar inaccuracies in the solution of the standard eigenvalue problem $Ax = \lambda x$ have been greatly reduced by the development of an algorithm by Parlett and

* Received by the editors September 6, 1979. 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.

† Mathematics and Statistics Research Department, Computer Sciences Division, Union Carbide Corporation, Nuclear Division, Oak Ridge, Tennessee 37830.

Reinsch [7] for balancing the A matrix. Their algorithm determines a diagonal similarity transformation which reduces the magnitude range of its elements while it reduces the norm of A.

In this paper we present an algorithm for balancing the matrices in the generalized eigenvalue problem by permutations and two-sided diagonal transformations. The algorithm is specifically designed to precede the QZ-type algorithms, but improved accuracy is expected from most eigensystem solvers. The scaling strategy is discussed in § 2, and the details of the three-step algorithm are given in § 3. The test results are presented in § 4.

2. Determination of scaling strategy. Instead of solving the desired problem represented by (1), we are usually required to solve a "nearby" problem given by

$$(A+G)x = \lambda(B+H)x, \tag{2}$$

where G and H represent data error matrices for A and B, respectively. A QZ-type algorithm given this nearby problem computes the exact eigenvalues of

$$(A+G+S\varepsilon_0)\bar{x} = \bar{\lambda}(B+H+T\varepsilon_0)\bar{x}, \tag{3}$$

where $\|S\| \leqq f(n)\|A+G\|$, $\|T\| \leqq g(n)\|B+H\|$, ε_0 is the basic computer roundoff error and the functions f and g are polynomials of modest degree (see Ward [10]). Using a perturbation analysis similar to Wilkinson [12, pp. 64–69], we find that for a simple eigenvalue λ_k of (1) there exists an eigenvalue $\bar{\lambda}_k$ of (3) such that

$$\bar{\lambda}_k - \lambda_k = \frac{y_k^T(G-\lambda_k H)x_k}{y_k^T B x_k} + \frac{y_k^T(S-\lambda_k T)x_k}{y_k^T B x_k}\varepsilon_0 + O(\varepsilon_0^2), \tag{4}$$

where x_k and y_k are respectively the right and left unit-length eigenvectors associated with λ_k. Note that the first term in this error expression is a result of the data error matrices G and H and only the second term is a result of the computation errors made by the algorithm. Also note that there are two factors which affect the accuracy of $\bar{\lambda}_k$: the sizes of the perturbation matrices as reflected in the numerators above and the magnification of this effect by the eigenvalue sensitivity factor $(y_k^T B x_k)^{-1}$.

If we scale the rows and columns of (2), the resulting generalized eigenvalue problem becomes

$$[D_1(A+G)D_2](D_2^{-1}x) = \lambda[D_1(B+H)D_2](D_2^{-1}x),$$

where D_1 and D_2 are positive diagonal matrices. A QZ-type algorithm applied to the scaled problem computes exact eigenvalues of

$$[D_1(A+G)D_2 + U\varepsilon_0](D_2^{-1}\hat{x}) = \hat{\lambda}[D_1(B+H)D_2 + V\varepsilon_0](D_2^{-1}\hat{x}), \tag{5}$$

where $\|U\| \leqq \hat{f}(n)\|D_1(A+G)D_2\|$ and $\|V\| \leqq \hat{g}(n)\|D_1(B+H)D_2\|$. For the simple eigenvalue λ_k of (1), there exists a computed eigenvalue $\hat{\lambda}_k$ of the scaled problem such that

$$\hat{\lambda}_k - \lambda_k = \frac{y_k^T(G-\lambda_k H)x_k}{y_k^T B x_k} + \frac{y_k^T D_1^{-1}(U-\lambda_k V)D_2^{-1}x_k}{y_k^T B x_k}\varepsilon_0 + O(\varepsilon_0^2). \tag{6}$$

Comparing (4) and (6), we notice that scaling has no effect upon the error induced in the eigenvalue by the data error matrices G and H. Hence, in the remainder of this paper we will refer to the given problem (2) using the standard terminology $Ax = \lambda Bx$.

The scaling strategy should be chosen strictly upon the effect of the computation errors made by the algorithm; that is, D_1 and D_2 should be chosen such that $y_k^T D_1^{-1}(U-\lambda_k V)D_2^{-1}x_k$ is minimized. Unfortunately, U, V, λ_k, x_k and y_k are unknown

at the time that the scaling must be performed. The problem is further complicated by the fact that U and V are dependent upon D_1 and D_2. A scaling strategy based upon this minimization appears intractable.

If we assume that all the elements of U are of the same order of magnitude, which would be slightly larger than the maximum element in D_1AD_2, and similarly for V, (5) immediately suggests a potential scaling strategy. This strategy would be to scale all the elements in A and B to the same order of magnitude. Thus, the perturbations in the elements of the scaled matrices caused by the computational errors would be of the same relative magnitudes, preventing possible loss of extreme accuracy in the smaller matrix elements. It should be noted that the above assumption on U and V would be the usual situation, although exceptions can be easily contrived.

Let us consider the effect of this strategy upon the error in the computed eigenvalue as given by (6). If we denote the diagonal elements of D_1 by r_i and the diagonal elements of D_2 by c_j, the expression $|y_k^T D_1^{-1}(U-\lambda_k V)D_2^{-1}x_k|$ with our assumption on U and V now becomes bounded by

$$\theta[(\max_{i,j}|r_iA_{ij}c_j|)+|\lambda_k|\ (\max_{i,j}|r_iB_{ij}c_j|)]\left[\sum_{i=1}^{n}\frac{|(y_k)_i|}{r_i}\right]\left[\sum_{j=1}^{n}\frac{|(x_k)_j|}{c_j}\right],$$

where θ is a "small" positive scalar greater than 1. If a small eigenvalue λ_k is primarily determined by the smaller elements in A and B, the i components of y_k and the j components of x_k corresponding to the small A_{ij}'s and B_{ij}'s will be much larger than the other components of y_k and x_k, and the r_i's and c_j's corresponding to these small elements will be larger than the other diagonals of D_1 and D_2. Thus, the above expression is smaller than in the unscaled ($r_i = c_j = 1$) problem. All the components of the left and right eigenvectors corresponding to the large eigenvalues will likely be of the same order of magnitude, and the above expression will not drastically change from the unscaled problem.

Our scaling strategy is then to scale A and B so that the magnitude of each of their elements is as close to unity as possible. (Unity was chosen since scaling by a scalar has no effect upon generalized eigenvalue problems and it simplifies the resulting minimization problem as discussed in § 3.2.) This strategy results in equalizing the roundoff errors occurring during the eigensystem computation, which is similar to the strategy espoused by Dongarra et al. [3] for scaling the coefficient matrix before solving a system of linear equations.

3. Algorithmic details. The balancing algorithm consists of three distinct steps designed to increase either the efficiency or the accuracy of the eigensystem solver which follows. Each step will be discussed and analyzed separately. The Fortran code consists of a driver routine which calls separate routines for performing each of the three steps. A similar arrangement exists for back-transforming the computed eigenvectors to those of the original system.

3.1. Step 1. Reduce the order. The first step consists of determining permutation matrices P_1 and P_2 such that the permuted matrices have the block structure indicated below:

$$(7)\qquad P_1AP_2=\begin{bmatrix}A_{11} & A_{12} & A_{13}\\ 0 & A_{22} & A_{23}\\ 0 & 0 & A_{33}\end{bmatrix},\qquad P_1BP_2=\begin{bmatrix}B_{11} & B_{12} & B_{13}\\ 0 & B_{22} & B_{23}\\ 0 & 0 & B_{33}\end{bmatrix},$$

where A_{11}, A_{33}, B_{11}, and B_{33} are upper triangular matrices. This step is analogous to the first step in Parlett and Reinsch's algorithm. If we denote the diagonals of A_{11}, A_{33},

B_{11} and B_{33} by $\alpha_i^{(1)}$, $\alpha_i^{(3)}$, $\beta_i^{(1)}$ and $\beta_i^{(3)}$ respectively, the eigenvalues of $Ax = \lambda Bx$ are now given by $\alpha_i^{(1)}/\beta_i^{(1)}$, $\alpha_i^{(3)}/\beta_i^{(3)}$ and the eigenvalues of $A_{22}y = \lambda B_{22}y$. Thus, this first step attempts to reduce the order of the generalized eigenvalue problem to be solved. If no such permutation matrices exist, then $A_{22} = A$, $B_{22} = B$, and other submatrices do not exist. If permutations exist which transform both A and B into upper triangular form, then A_{22} and B_{22} do not exist and all the eigenvalues have been found.

The algorithm for determining the permutation matrices P_1 and P_2 can be best described by use of the $n \times n$ matrix W defined by $W_{ij} = |A_{ij}| + |B_{ij}|$, although the matrix W is never computed. The rows of W are first scanned for a row containing only one nonzero element. This element is permuted to the (n, n)-position, which determines the nth row of P_1 and the nth column of P_2. The scan is then repeated on the leading principal submatrix of order $n-1$, determining the $(n-1)$st row and column of P_1 and P_2 respectively. This process continues until either we have more than one nonzero element in each row of the reduced W or W is in upper triangular form. In the latter case we have completely determined P_1 and P_2. In the former case we perform similar scanning operations on the columns of the remaining principal submatrix of W. The single nonzero elements are permuted to the (1,1)-position, and the matrix is deflated from the top.

The operation count for step one is best stated in terms of maximum and minimum values. The minimum number of operations results when a reduction in the order of matrices is not possible by permutations. In this case $4n^2$ comparisons are required. The maximum number of operations occurs when both A and B can be permuted into upper triangular form. This reduction may require up to $\frac{2}{3}n^3 + O(n^2)$ comparisons and $9n^2 + O(n)$ replacements. Since eigensystem solvers require γm^3 operations, where m is the order of the matrices and γ is $O(10)$, the cost for this step is either minor or more than repaid by the savings in the computation of the eigensystem of $A_{22}y = \lambda B_{22}y$.

3.2. Step 2. Scale *A* and *B*. The second step involves scaling the A_{22} and B_{22} matrices by two-sided diagonal transformations, and is normally the major step in the balancing algorithm. We will drop the subscripts from A and B, denote their order by m, and discuss the computation of $m \times m$ diagonal matrices D_1 and D_2 such that the elements of D_1AD_2 and D_1BD_2 have magnitudes as close to unity as possible.

If we let D_1 and D_2 be expressed as integral powers of the computer radix ρ, denoted by r_i and c_j respectively, and look at logarithms to the base ρ, the scaling strategy results in the following minimization problem:

$$\min_{r_i, c_j} \sum_{i,j=1}^{m} [(r_i + c_j + \log_\rho |A_{ij}|)^2 + (r_i + c_j + \log_\rho |B_{ij}|)^2]. \tag{8}$$

The problem is similar to that studied by Curtis and Reid [2], but our technique for its solution differs.

By eliminating the requirement that the r_i's and c_j's be integers and differentiating, we find that the solution to the minimization problem is the solution to the following consistent, singular, system of linear equations:

$$\begin{bmatrix} F_1 & E \\ E^T & F_2 \end{bmatrix} \cdot \begin{bmatrix} r \\ c \end{bmatrix} = \begin{bmatrix} -g \\ -h \end{bmatrix}, \tag{9}$$

where F_1 and F_2 are $m \times m$ diagonal matrices whose elements are the number of nonzeros in the rows and columns (respectively) of A and B, g and h are the vectors of row and column sums of logarithms of nonzero elements in A and B and E is the sum of the incidence matrices of A and B. In the case where A and B contain only nonzero

elements, the system simplifies to

$$2\begin{bmatrix} mI & ee^T \\ ee^T & mI \end{bmatrix} \cdot \begin{bmatrix} r \\ c \end{bmatrix} = \begin{bmatrix} -g \\ -h \end{bmatrix}, \tag{10}$$

where e is the m-vector containing a one in each component. Much information can be determined about the linear system in (10). The Moore–Penrose generalized inverse of the coefficient matrix is given by

$$\frac{1}{2}\begin{bmatrix} \frac{1}{m}I - \frac{3}{4m^2}ee^T & \frac{1}{4m^2}ee^T \\ \frac{1}{4m^2}ee^T & \frac{1}{m}I - \frac{3}{4m^2}ee^T \end{bmatrix},$$

and the Moore–Penrose solution of (10) is given by

$$\begin{bmatrix} r \\ c \end{bmatrix} = -\frac{1}{2m}\begin{bmatrix} g - \frac{1}{2m}(e^T g)e \\ h - \frac{1}{2m}(e^T g)e \end{bmatrix}.$$

Moreover, the Hessian matrix corresponding to (8) is positive semidefinite, making the above solution a *global minimum.*

The balancing algorithm computes the solution to (9) by an iterative scheme which takes advantage of the above knowledge about the linear system in (10). Denoting the coefficient matrix in (9) and (10) by $M-N$ and M respectively and the right side vector by b gives an iteration of the form

$$x^{(k+1)} = x^{(k-1)} + \omega_{k+1}(\mu_k z^{(k)} + x^{(k)} - x^{(k-1)}),$$

where

$$Mz^{(k)} = b - (M-N)x^{(k)}.$$

The acceleration parameters ω_{k+1} and μ_k are determined by the conjugate gradient method. This iterative scheme has been developed and analyzed by Concus et al. [1]. Since integers r_i and c_j are desired, the iteration is stopped after the magnitude of the correction to r and c is bounded by 0.5.

The number of operations per iteration is given by $4m^2$ additions and $4m^2$ comparisons plus $O(m)$ operations. If A and B are reasonably dense (i.e., contain mostly nonzero elements), the iterates converges in 2 or 3 iterations, and only $O(m^2)$ operations are required for this second step. When A and B are very sparse, the iterative scheme usually converges within $\frac{1}{2}m$ iterations, although large sparse matrices have not been tested to determine the extent of this overestimate. Theoretically, the iteration will have converged to full accuracy after m iterations.

3.3. Step 3. Grade *A*/*B*. The scaled A_{22} and B_{22} matrices may still have some variation in the magnitudes of their elements. The third step attempts to arrange these magnitudes inside the matrices in order to reduce the size of the errors incurred during the eigensystem computation. The matrices $D_1A_{22}D_2$ and $D_1B_{22}D_2$ will be denoted as the A and B matrices in this section.

The QZ-type algorithms have the strong tendency of finding the eigenvalues in decreasing order starting at the top of the triangularized matrices (i.e., $|\alpha_1/\beta_1| \geqq |\alpha_2/\beta_2| \geqq \cdots \geqq |\alpha_m/\beta_m|$). The α_i's and β_i's are determined by the application of a sequence of unitary transformations to the rows and columns of A and B. Thus, it would be advantageous for the QZ-type algorithms if the (i, j)-elements of the A matrix decrease in magnitude and those of the B matrix increase in magnitude as $i+j$ increases.

This objective cannot always be accomplished by applying permutation matrices to A and B while still preserving the eigenvalues. However, we can determine permutation matrices such that

$$\left.\begin{aligned} \|A_i\|_1/\|B_i\|_1 &\geqq \|A_j\|_1/\|B_j\|_1 \\ \|A^i\|_1/\|B^i\|_1 &\geqq \|A^j\|_1/\|B^j\|_1 \end{aligned}\right\} \quad \text{for } i<j,$$

where the subscripts denote the rows of the permuted matrix and the superscripts denote the columns. This permutation has the effect of trying to make $|U_{ij}|$ and $|V_{ij}|$ in (5) approximately equal to $\theta|(D_1AD_2)_{ij}|$ and $\tau|(D_1BD_2|_{ij}|$ respectively, where θ and τ are "small" positive scalars greater than 1. That is, it tries to prevent the spreading of the errors associated with the large elements of D_1AD_2 and D_1BD_2.

Since permutation of the columns of a matrix does not affect the 1-norm of the rows and vice versa, the grading algorithm first determines the permutation matrix G_2 such that the ratio of the column norms of AG_2 to the corresponding column norms of BG_2 appear in nonincreasing order. Then, G_1 is determined such that the ratio of the row norms of G_1AG_2 to those of G_1BG_2 appear in nonincreasing order.

The actual number of operations required for this step is a function of n and m, the order of the original and reduced matrices respectively, and the norms of the rows and columns of B. The maximum number of operations is given by $2m^2$ additions and $6nm+6\frac{1}{2}m^2$ replacements plus $O(m+n)$ operations. Certainly, this number will be insignificant compared to the number of operations required by the eigensystem solver.

3.4. Summary. After completion of the three-step balancing algorithm, the problem $Ax=\lambda Bx$ has been transformed into the problem $A'v=\lambda B'v$, where

$$A' = \begin{bmatrix} A_{11} & A_{12}D_2G_2 & A_{13} \\ 0 & G_1D_1A_{22}D_2G_2 & G_1D_1A_{23} \\ 0 & 0 & A_{33} \end{bmatrix}$$

and

$$B' = \begin{bmatrix} B_{11} & B_{12}D_2G_2 & B_{13} \\ 0 & G_1D_1B_{22}D_2G_2 & G_1D_1B_{23} \\ 0 & 0 & B_{33} \end{bmatrix},$$

with the partitioning, A_{ij} and B_{ij} defined as in (7). If only the eigenvalues are desired, the problem has been reduced to considering only the matrices $G_1D_1A_{22}D_2G_2$ and $G_1D_1B_{22}D_2G_2$. If the eigenvectors x are also desired, they may be computed from the eigenvectors v by the equation

$$x = P_2 \begin{bmatrix} v_1 \\ D_2G_2v_2 \\ v_3 \end{bmatrix},$$

where v is partitioned consistent with A' and B'.

The first step does not rely on any assumptions concerning the eigensystem solver which follows or any information about the size of the matrix elements. Thus, the code implementing step one may be used to reduce the order of the matrices prior to calling any generalized eigensystem solver.

Step two attempts to increase the "involvement" of the smaller matrix elements in computing the eigenvalues. This step may hinder the *detection* of eigenvalue inaccuracies if the smaller elements have much larger relative errors. For example, consider the following eigenproblem on a 6 decimal digit computer:

$$\begin{bmatrix} 1 & 10^{-5} \\ 2 & 10^{-5} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} 1 & 0 \\ 0 & 10^{-5} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

where the user knows that there are errors of the order 10^{-6} in all the elements of A and B. The QZ algorithm applied to the unscaled matrices would return one eigenvalue on the order of $10^{-5}/10^{-5}$ with about one accurate digit, thus alerting the user to the potential inaccuracy of the eigenvalue and the closeness to $\det(A-\lambda B) \equiv 0$. The QZ algorithm applied to the scaled matrices would return more accurate eigenvalues to the given eigenproblem but would return α's and β's on the order of unity, providing the user with no additional information. In practice this problem has only occurred when the rank deficiency of A and B is the result of one row or column rather than a linearly dependent set of several rows or columns. That is, small α's and β's which result from cancellation have not been adversely magnified by the scaling. Also, this potential problem does not occur when the relative errors in the matrix elements are of similar magnitudes. The code has been written so that the scaling routine can be easily bypassed or replaced by a routine implementing a different strategy.

The third step has been specifically designed to precede the QZ-type algorithms. However, most eigensystem solvers involve matrix-vector products or row and column transformations which would produce more accurate results with graded matrices. Since the cost is low and improved accuracy is expected, it is recommended to use this step prior to calling any generalized eigensystem solver.

4. Test results. The balancing algorithm has been extensively tested on the IBM System computers at the Oak Ridge National Laboratory. The code has been analyzed for portability by the PFORT verifier described by Ryder [8] and for data flow and software errors by the DAVE code described by Osterweil and Fosdick [6]. The QZ algorithm in EISPACK described by Garbow et al. [4] has been modified to accept partitioned matrices of the form in (7) and was used as the eigensystem solver in all test cases. The modification involved only minimal changes to the subroutines QZHES and QZIT.

Three of the numerical experiments are discussed below. The computations were performed in double precision (relative accuracy of 16.8 decimal digits) on an IBM 3033 computer. A ρ-value of two was used in the second step instead of the actual computer radix 16, since better equalization of the element magnitudes occurs with smaller ρ-values and the additional rounding errors are minimal. The results are presented in tabular form with the first inaccurate digit in the eigenvalues underlined. Only the first few digits of the α's, β's and sensitivity factors are given, since their magnitudes are the primary interest. The signed integer superscript at the end of a number represents its decimal exponent.

Test Case 1. *Pseudo-random magnitudes.*

$$A_1 = \begin{bmatrix} -2.0^{+1} & -1.0^{+4} & -2.0^{+0} & -1.0^{+6} & -1.0^{+1} & -2.0^{+5} \\ 6.0^{-3} & 4.0^{+0} & 6.0^{-4} & 2.0^{+2} & 3.0^{-3} & 3.0^{+1} \\ -2.0^{-1} & -3.0^{+2} & -4.0^{-2} & -1.0^{+4} & 0.0^{+0} & 3.0^{+3} \\ 6.0^{-5} & 4.0^{-2} & 9.0^{-6} & 9.0^{+0} & 3.0^{-5} & 5.0^{-1} \\ 6.0^{-2} & 5.0^{+1} & 8.0^{-3} & -4.0^{+3} & 8.0^{-2} & 0.0^{+0} \\ 0.0^{+0} & 1.0^{+3} & 7.0^{-1} & -2.0^{+5} & 1.3^{+1} & -6.0^{+4} \end{bmatrix}$$

$$B_1 = \begin{bmatrix} -2.0^{+1} & -1.0^{+4} & 2.0^{+0} & -2.0^{+6} & 1.0^{+1} & -1.0^{+5} \\ 5.0^{-3} & 3.0^{+0} & -2.0^{-4} & 4.0^{+2} & -1.0^{-3} & 3.0^{+1} \\ 0.0^{+0} & -1.0^{+2} & -8.0^{-2} & 2.0^{+4} & -4.0^{-1} & 0.0^{+0} \\ 5.0^{-5} & 3.0^{-2} & 2.0^{-6} & 4.0^{+0} & 2.0^{-5} & 1.0^{-1} \\ 4.0^{-2} & 3.0^{+1} & -1.0^{-3} & 3.0^{+3} & -1.0^{-2} & 6.0^{+2} \\ -1.0^{+0} & 0.0^{+0} & 4.0^{-1} & -1.0^{+5} & 4.0^{+0} & 2.0^{+4} \end{bmatrix}$$

This test case was generated by starting with a well-balanced pair of matrices and scaling the rows and columns by pseudorandom powers of ten. The eigenvalues of $A_1x = \lambda B_1x$ are 1, 2, 3, 4, 5 and 6. In this test case all the eigenvalues are equally dependent upon the relative accuracy of each matrix element.

The output scaling vectors r and c computed by the balancing code were

$$r^T = [-9, 3, -3, 10, 0, -7],$$

$$c^T = [5, -5, 8, -11, 5, -8].$$

The first step had no effect upon the eigensystem computation since permutations could not isolate any eigenvalues. All the row and column norms of the scaled matrices were of the same order of magnitude; thus, step three also had little effect upon the computations. The results are presented in Tables 1 and 2. Since the eigenvalue sensitivities are based on error bounds, they may be pessimistic but will allow comparisons to be made between the original and balanced problems. The sensitivity

TABLE 1
Computational results from $A_1x = \lambda B_1x$.

Original problem	Balanced problem
0.9999999999986649	1.000000000000007
1.999999999997867	1.999999999999983
2.999999971458162	3.000000000000064
4.000000359881658	3.999999999993096
4.999998501541569	5.000000000024284
6.000001319640085	5.999999999980004
α/β	
$1.00^{+4}/1.00^{+4}$	$8.29^{-1}/8.29^{-1}$
$2.04^{+0}/1.02^{+0}$	$3.72^{-1}/1.86^{-1}$
$1.47^{-1}/4.89^{-2}$	$2.04^{+0}/6.81^{-1}$
$1.99^{+0}/4.97^{-1}$	$2.67^{+0}/6.68^{-1}$
$6.02^{-1}/1.20^{-1}$	$2.11^{+0}/4.22^{-1}$
$2.01^{+0}/3.35^{-1}$	$4.94^{-1}/8.23^{-2}$

TABLE 2
Eigenvalue sensitivities for $A_1x = \lambda B_1x$.

	$\lvert y^TBx\rvert^{-1}$		$\frac{\lVert A\rVert + \lvert\lambda\rvert\lVert B\rVert}{\lvert y^TBx\rvert}$	
Eigenvalues	Original	Balanced	Original	Balanced
1.0	1.43^{+4}	3.91^{+1}	4.77^{+10}	7.08^{+2}
2.0	1.58^{+5}	3.25^{+1}	8.65^{+11}	8.65^{+2}
3.0	4.33^{+6}	7.68^{+2}	3.29^{+13}	2.70^{+4}
4.0	7.12^{+7}	3.47^{+3}	6.91^{+14}	1.52^{+5}
5.0	1.69^{+8}	1.02^{+4}	2.00^{+15}	5.31^{+5}
6.0	1.06^{+8}	7.59^{+3}	1.48^{+15}	4.61^{+5}

factor $(y^TBx)^{-1}$ will be different for the two problems due to the normalization of the eigenvectors. This normalization did not appear in (4) and (6) since the eigenvectors also appeared in the numerator of these expressions.

Test Case 2. *Polynomial root solver.* The A_2 matrix is the lower Hessenberg companion matrix for a characteristic polynomial of order ten with the coefficients appearing in the last row and the integers one through ten as roots. The B_2 matrix is the identity matrix. Thus, the eigenvalues of $A_2x = \lambda B_2x$ are the integers one through ten and may be computed by standard eigenproblem solvers. This type of problem is frequently encountered and contains some ill-conditioned eigenvalues.

No reduction of the generalized eigenproblem was possible by permutations. The computed scaling vectors were

$$r^T = [6, 6, 6, 5, 3, 1, -2, -5, -10, -17],$$

$$c^T = [-5, -6, -6, -6, -4, -3, 0, 3, 7, 13].$$

The elements of the graded A_2 matrix ranged from 8.0^{+0} to 8.6^{-1}, and those of the graded B_2 matrix ranged from 6.25^{-2} to 2.0^{+0}. The results are presented in Tables 3 and 4.

The *QR* algorithm with balancing (subroutine *RG*) contained in EISPACK and described by Smith et al. [9] was also used to compute the eigenvalues of this test case. All the eigenvalues computed by the *QZ* algorithm using the balanced matrices were slightly more accurate than those computed by the *RG* code. The accuracy improvement ranged from 0.2 to 0.5 of a decimal digit. This difference may be the result of the *QR* balancing code using a ρ of 16, whereas the *QZ* balancing code used a ρ of 2.

Test Case 3. *Graded matrix.*

$$A_3 = \begin{bmatrix} 1.0^{+0} & 1.0^{+1} & & & & & & \\ 1.0^{+1} & 1.0^{+2} & 1.0^{+3} & & & & & \\ & 1.0^{+3} & 1.0^{+4} & 1.0^{+5} & & & & \\ & & 1.0^{+5} & 1.0^{+6} & 1.0^{+7} & & & \\ & & & 1.0^{+7} & 1.0^{+8} & 1.0^{+9} & & \\ & & & & 1.0^{+9} & 1.0^{+10} & 1.0^{+11} & \\ & & & & & 1.0^{+11} & 1.0^{+12} & \end{bmatrix}$$

B_3 is the identity matrix. In this test case, the small eigenvalues are primarily determined by the smaller matrix elements.

TABLE 3
Computational results from $A_2x = \lambda B_2x$.

Original problem	Balanced problem
Eigenvalues	
1.000000$\underline{1}$2369418	0.9999999999999$\underline{8}$09
1.999999$\underline{9}$86337335	2.000000000000$\underline{1}$343
3.000000$\underline{0}$1590798	2.999999999$\underline{7}$8608
4.000000$\underline{1}$2841566	4.000000000$\underline{1}$56160
5.000000$\underline{2}$2317214	4.999999999$\underline{3}$69756
5.999999$\underline{7}$8272570	6.000000001545631
6.999999$\underline{8}$7591551	6.999999997618605
7.999999999$\underline{8}$892347	8.000000002256280
8.999999999$\underline{7}$20611	8.999999998799755
10.00000000006$\underline{6}$648	10.00000000027369
α/β	
$1.00^{+0}/1.00^{+0}$	$7.31^{-1}/7.31^{-1}$
$2.00^{+0}/1.00^{+0}$	$1.15^{+0}/5.74^{-1}$
$3.00^{+0}/1.00^{+0}$	$1.49^{+0}/4.95^{-1}$
$4.00^{+0}/1.00^{+0}$	$1.64^{+0}/4.11^{-1}$
$5.00^{+0}/1.00^{+0}$	$2.08^{+0}/4.16^{-1}$
$6.00^{+0}/1.00^{+0}$	$2.20^{+0}/3.66^{-1}$
$7.00^{+0}/1.00^{+0}$	$2.09^{+0}/2.98^{-1}$
$8.00^{+0}/1.00^{+0}$	$2.22^{+0}/2.78^{-1}$
$9.00^{+0}/1.00^{+0}$	$2.20^{+0}/2.44^{-1}$
$1.00^{+1}/1.00^{+0}$	$2.32^{+0}/2.32^{-1}$

TABLE 4
Eigenvalue sensitivities for $A_2x = \lambda B_2x$.

	$\lvert y^TBx\rvert^{-1}$		$\dfrac{\lVert A\rVert+\lvert\lambda\rvert\lVert B\rVert}{\lvert y^TBx\rvert}$	
Eigenvalues	Original	Balanced	Original	Balanced
1.0	8.84^{+1}	8.55^{+1}	1.13^{+9}	1.15^{+3}
2.0	9.84^{+4}	3.86^{+3}	1.26^{+12}	5.96^{+4}
3.0	1.03^{+7}	6.18^{+4}	1.32^{+14}	1.08^{+6}
4.0	2.49^{+8}	4.87^{+5}	3.17^{+15}	9.46^{+6}
5.0	2.28^{+9}	2.11^{+6}	2.91^{+16}	4.53^{+7}
6.0	9.99^{+9}	5.41^{+6}	1.27^{+17}	1.27^{+8}
7.0	2.32^{+10}	8.39^{+6}	2.96^{+17}	2.13^{+8}
8.0	2.92^{+10}	7.77^{+6}	3.73^{+17}	2.13^{+8}
9.0	1.89^{+10}	3.96^{+6}	2.41^{+17}	1.17^{+8}
10.0	4.92^{+9}	8.52^{+5}	6.28^{+16}	2.68^{+7}

No reduction again occurred in the first step of the balancing algorithm. The scaling vectors converged in 3 iterations and were

$$r^T = [0, -2, -5, -8, -10, -13, -13],$$

$$c^T = [0, -2, -5, -8, -10, -13, -13].$$

The graded diagonal B_3 matrix contained elements ranging from 1.49^{-8} to 1.0^{+0}. The graded A_3 matrix remained a banded matrix of the same bandwidth. The comparison of results is presented in Tables 5 and 6.

TABLE 5
Computational results from $A_3x = \lambda B_3x$.

Original problem	Balanced problem
Eigenvalues	
-9.463474156469348^{+8}	-9.463474156469419^{+8}
-9.463469295288439^{+2}	-9.463469197097401^{+2}
9.998989999974069^{-1}	9.998990201929425^{-1}
1.046337224646799^{+3}	1.046337214788077^{+3}
1.009899030199693^{+6}	1.009899030199714^{+6}
1.046337712685939^{+9}	1.046337712685945^{+9}
1.010000009803941^{+12}	1.010000009803940^{+12}
α/β	
$-9.46^{+8}/1.00^{+0}$	$1.13^{+2}/1.19^{-7}$
$-9.46^{+2}/1.00^{+0}$	$-5.89^{+0}/6.22^{-3}$
$1.00^{+0}/1.00^{+0}$	$9.23^{-1}/9.23^{-1}$
$1.05^{+3}/1.00^{+0}$	$8.52^{+0}/8.14^{-3}$
$1.01^{+6}/1.00^{+0}$	$1.92^{+1}/1.90^{-5}$
$1.05^{+9}/1.00^{+0}$	$1.31^{+2}/1.25^{-7}$
$1.01^{+12}/1.00^{+0}$	$1.51^{+4}/1.49^{-8}$

TABLE 6
Eigenvalue sensitivities for $A_3x = \lambda B_3x$.

	$\lvert y^TBx\rvert^{-1}$		$\dfrac{\lVert A\rVert+\lvert\lambda\rvert\lVert B\rVert}{\lvert y^TBx\rvert}$	
Eigenvalues	Original	Balanced	Original	Balanced
-9.46^{+8}	1.00^{+0}	3.57^{+7}	1.10^{+12}	3.38^{+16}
-9.46^{+2}	1.00^{+0}	9.15^{+2}	1.10^{+12}	1.59^{+7}
1.00^{+0}	1.00^{+0}	1.17^{+0}	1.10^{+12}	1.93^{+4}
1.05^{+3}	1.00^{+0}	8.31^{+2}	1.10^{+12}	1.45^{+7}
1.01^{+6}	1.00^{+0}	7.16^{+4}	1.10^{+12}	7.35^{+10}
1.05^{+9}	1.00^{+0}	3.24^{+7}	1.10^{+12}	3.39^{+16}
1.01^{+12}	1.00^{+0}	6.71^{+7}	2.11^{+12}	6.78^{+19}

The *RG* code was also used to solve this eigenproblem. Since the A_3 matrix is symmetric, balancing in the *RG* code will have no effect upon the matrix. In the *QR*-computed eigenvalues there were about 15 accurate digits in the four of largest magnitude, about 11 in the next two largest, and only 4 accurate digits in the eigenvalue of smallest magnitude. Note that A_3 is graded in the wrong direction for the *QR* algorithm. This example illustrates that one cannot blindly use the *QR* algorithm with balancing and always expect results which are as accurate as possible in the presence of rounding errors.

The following observations can be drawn from the results of the many random, pathological, and pedagogical sets of matrices which have been tested.

1. Tremendous improvement in the accuracy of the eigenvalues is possible when the elements of A and B have widely varying magnitudes.

2. When significant improvement in eigenvalue accuracy is not obtained, the total number of QZ iterations required to converge to the eigenvalues is usually reduced. In fact, the savings in computer time executing the QZ-type algorithms may easily offset the cost of the balancing algorithm.

3. The bounds on the theoretical sensitivity of the larger eigenvalues may become larger when the matrices are balanced, but the actual accuracy is not seriously affected.

4. None of the eigenvalues corresponding to the balanced matrices was significantly less accurate than those of the original matrices.

5. Balancing helps the accuracy of the preliminary reduction phase of the QZ algorithm as well as the iteration phase.

6. Comparison of computed eigenvalues for $Ax = \lambda Bx$ and $Bx = (1/\lambda)Ax$ indicates that the variation between the computed eigenvalues is less when the balanced matrices are used than when the original matrices are used. The accuracy variation averaged 0.6 decimal digits per eigenvalue for the balanced matrices versus 1.0 for the original matrices over 24 test cases.

5. Acknowledgments. The author wishes to thank J. J. Moré, D. S. Scott and G. W. Stewart for their helpful discussions during the course of this study. The author also wishes to thank the referee whose suggestion lead to an improvement in the third step of the algorithm.

REFERENCES

[1] P. CONCUS, G. H. GOLUB AND D. P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–332.

[2] A. R. CURTIS AND J. K. REID, *On the automatic scaling of matrices for Gaussian elimination*, J. Inst. Maths. Applics., 10 (1972), pp. 118–124.

[3] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[4] 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.

[5] C. B. MOLER AND G. W. STEWART, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241–256.

[6] L. J. OSTERWEIL AND L. D. FOSDICK, DAVE—*A Validation Error Detector and Documentation System for Fortran Programs*, Software Practice and Experience, 6 (1976), pp. 473–486.

[7] B. N. PARLETT AND C. REINSCH, *Balancing a matrix for calculation of eigenvalues and eigenvectors*, Numer. Math., 13 (1969), pp. 293–304.

[8] B. G. RYDER, *The* PFORT *Verifier*, Software Practice and Experience, 4 (1974), pp. 359–377.

[9] B. T. SMITH, J. M. BOYLE, J. J. DONGARRA, B. S. GARBOW, Y. IKEBE, V. C. KLEMA AND C. B. MOLER, *Matrix Eigensystem Routines*—EISPACK *Guide*, Lecture Notes in Computer Science 6, Springer-Verlag, New York, 1976.

[10] R. C. WARD, *Error analysis of Householder transformations as applied to the standard and generalized eigenvalue problem*, NASA Technical Note D-7764, NASA Langley Research Center, Hampton, VA, 1974.

[11] R. C. WARD, *The combination shift QZ algorithm*, SIAM J. Numer. Anal., 12 (1975), pp. 835–853.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0004 $01.00/0

ON COMPUTING ROBUST SPLINES AND APPLICATIONS*

FLORENCIO I. UTRERAS†

Abstract. We present a new method for computing spline functions associated to a weight function. This algorithm is based on the Newton minimization algorithm for unrestricted problems. Then we use this method to compute, by a penalty function method, the spline functions with restricted values and show the convergence of this method. Finally, we give some numerical examples showing the nice smoothing properties of these spline functions.

Key words. splines, noisy data, penalty method

1. Introduction. Consider the problem of approximating an unknown function f belonging to $H^2[a, b]$ given its values known with error at n different points of the real interval $[a, b]$. Let $z_1, \cdots, z_n$ be those values, i.e.,

$$z_i = f(t_i) + \varepsilon_i, \qquad i = 1, 2, \cdots, n, \tag{1}$$

where

$$a < t_1 < t_2 < \cdots < t_n < b, \qquad |\varepsilon_i| \leqq \varepsilon, \qquad i = 1, 2, \cdots, n, \tag{2}$$

and assume that ε_i, $i = 1, \cdots, n$ are independent random variables with uniform distribution in $[-\varepsilon, \varepsilon]$.

To solve this problem Laurent [5] has proposed approximating f by the solution σ of the problem

$$\underset{\substack{g \in H^2[a,b] \\ |g(t_i) - z_i| \leqq \varepsilon,\ i=1,2,\cdots,n}}{\text{Minimize}} \int_a^b [g''(t)]^2\, dt. \tag{3}$$

In other words, among all functions g belonging to $H^2[a, b]$ whose values at the knots lie in the range of possible values for f, we seek the functions having the lowest "energy."

It is well known (see [5]) that for $n \geqq 2$ there exists one and only one solution to problem (3).

In [2], the authors analyze several algorithms to solve this minimization problem and conclude that the best is the one proposed by Laurent [7] based on a dual iteration. Unfortunately, even though the best, it is very expensive (cf. [2]).

In this paper, we use a penalty function method and prove, for an appropriate choice of the penalty functions, the convergence of this algorithm. To solve the partial problems, we use a Newton type method and give an efficient method allowing us to perform each iteration at a very low cost.

More precisely, it is clear that problem (3) is equivalent to:

$$\underset{u \in H^2[a,b]}{\text{Minimize}} \left\{ \int_a^b [u''(t)]^2\, dt + \sum_{i=1}^{n} \chi(u(t_i) - z_i) \right\}, \tag{4}$$

where χ is a function defined on $\mathbb{R}$, with values on $\bar{\mathbb{R}}$

$$\chi(\alpha) = \begin{cases} 0, & \alpha \in [-\varepsilon, \varepsilon], \\ +\infty, & \alpha \notin [-\varepsilon, \varepsilon]. \end{cases} \tag{5}$$

* Received by the editors January 30, 1980, and in revised form December 3, 1980.

† Departamento de Matematicas, Facultad de Ciencias Fisicas y Matematicas, Universidad de Chile, Casilla 5272 Correo 3, Santiago, Chile.

The penalized problem then becomes

$$(6)_k \qquad \underset{u\in H^2[a,b]}{\text{Minimize}} \left\{ \int_a^b [u''(t)]^2\, dt + \sum_{i=1}^{n} \Psi_k(u(t_i)-z_i) \right\},$$

where Ψ_k is a convex, twice differentiable even function such that

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

In § 4 we choose the sequence $\{\Psi_k\}$, $k \geqq 0$, in such a way that we have

$$\lim_{k\to\infty} \Psi_k(x) = \chi(x) \quad \text{for all } x. \tag{8}$$

If we call $\sigma^{(k)}$ the solution (if there is one) of problem $(6)_k$, we prove that

$$\lim_{k\to\infty} \sigma^{(k)} = \sigma, \tag{9}$$

the convergence being in the sense of $H^2[a, b]$.

To solve the partial problem $(6)_k$, we propose to use a Newton method with a specially designed method to perform each iteration. In this way, the cost of each iteration is linear in n.

2. Spline functions associated with a weight function Ψ. The results given in this section have been obtained by several authors in recent years (see [3], [4], [6], [8], [11]).

Let $\Psi: \mathbb{R} \to \bar{\mathbb{R}}$ be a lower semicontinuous convex function such that $\Psi(0)=0$. Consider the problem

$$(\mathrm{P}_\Psi) \qquad \underset{u\in H^2[a,b]}{\text{Minimize}} \left\{ \int_a^b [u''(t)]^2\, dt + \sum_{i=1}^{n} \Psi(u(t_i)-z_i) \right\}.$$

Then (P_Ψ) has a unique solution σ. Moreover, σ is a cubic spline; i.e.:

(i) σ is a polynomial of degree 1 in $[a, t_1]$, $[t_n, b]$.

(ii) σ is a cubic polynomial in $[t_i, t_{i+1}]$, $i = 1, 2, \cdots, n-1$.

(iii) $\sigma \in C^2[a, b]$.

In addition, σ satisfies

(iv) $$\sigma'''(t_i^+) - \sigma'''(t_i^-) \in -\partial\Psi(\sigma(t_i)-z_i), \qquad i = 1, 2, \cdots, n,$$

where $\partial\Psi(\bar{x})$ denotes the subdifferential set of Ψ at $\bar{x}$ (cf. [5], [10]).

A more precise idea of the behavior of such a spline can be obtained with more restrictive hypotheses:

(H1) Ψ is strictly convex on $\mathbb{R}$.

(H2) $\Psi \in C^2(\mathbb{R})$.

In this case, the problem (P_Ψ) has a unique solution, the subdifferential set reduces to the derivative of Ψ and condition (iv) becomes

(v) $$\sigma'''(t_i^+) - \sigma'''(t_i^-) = -\Psi'(\sigma(t_i)-z_i), \qquad i = 1, 2, \cdots, n.$$

We also know that the set of cubic splines with knots $\{t_1, t_2, \cdots, t_n\}$ is a linear n-dimensional space, and that $\{\sigma_i, i = 1, \cdots, n\}$ defined by

$$\sigma_i(t_j) = \begin{cases} 1, & j = i, \quad j = 1, \cdots, n, \\ 0, & j \neq i, \quad i = 1, \cdots, n \end{cases} \tag{10}$$

form a canonical basis. Moreover,

$$\sigma(t) = \sum_{i=1}^{n} \sigma(t_i)\sigma_i(t). \tag{11}$$

From this fact problem (P_Ψ) has the same solution as

$$(12)\ (P'_\Psi) \qquad \underset{x\in\mathbb{R}^n}{\text{Minimize}} \left\{ \sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij}x_i x_j + \sum_{i=1}^{n} \Psi(x_i - z_i) \right\},$$

where

$$(13) \qquad \omega_{ij} = \int_a^b \sigma_i''(t)\sigma_j''(t)\,dt.$$

To minimize (12), it is necessary and sufficient, (under hypotheses H1, H2), that the solution y satisfy

$$(14) \qquad 2\sum_{j=1}^{n} \omega_{ij}y_j + \Psi'(y_i - z_i) = 0, \qquad i = 1, 2, \cdots, n.$$

To find y, it is tempting to use a fixed-point method. Unfortunately, given the properties of Ω, the set of functions Ψ for which such a method converges is very small (see [2]).

In the following section we use a Newton method to solve (12) directly and see how our technique to perform iterations is also applicable to the IRLS algorithm (see [1]).

The problem of determining such a spline function has often arisen in the literature (see, e.g., [2], [3], [8], [4]). In particular, the use of spline functions for the smoothing of noisy data containing outliers has led Lenth [8], Huber [3] and others to use this type of splines; they call them robust splines because of their relation with robust statistics. In this paper, we also use this name.

3. Calculating the robust spline. As we have already said, to find y, the vector of values of σ at the knots, we use Newton's method to minimize

$$(15) \qquad J(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} \omega_{ij}x_i x_j + \sum_{i=1}^{n} \Psi(x_i - z_i).$$

J is convex because Ψ is convex and the matrix $\Omega = (\omega_{ij})$ is positive semidefinite (see [6]). Moreover, if Ψ is twice differentiable, we can apply Newton's method; that is, given $x^{(k)}$ we compute $x^{(k+1)}$ as the minimizer of the quadratic functional

$$(16) \qquad \begin{aligned} Q_k(x) = \langle x, \Omega x\rangle + \sum_{i=1}^{n} \Psi(x_i^{(k)} - z_i) + \sum_{i=1}^{n} \Psi'(x_i^{(k)} - z_i)(x_i - x_i^{(k)}) \\ + \tfrac{1}{2}\sum_{i=1}^{n} \Psi''(x_i^{(k)} - z_i)(x_i - x_i^{(k)})^2. \end{aligned}$$

In the case when Ψ is differentiable only once, we can use the IRLS algorithm, which amounts to replacing $\Psi''(x_i^{(k)} - z_i)$ by $\Psi'(x_i^{(k)} - z_i)/(x_i^{(k)} - z_i)$. This choice is convergent (see [1]), but the rate of convergence is only linear.

The vector $x^{(k+1)}$ will be uniquely determined if and only if Q_k is positive definite, that is, if $\Omega + D$ is positive definite, where D is a diagonal matrix with elements $d_{ii} = \frac{1}{2}\Psi''(x_i^{(k)} - z_i)$. In the following theorem we give necessary and sufficient conditions for $\Omega + D$ to be positive definite.

THEOREM 1. *A necessary and sufficient condition for $\Omega + D$ to be positive definite is that there are at least two elements of D different from zero.*

Proof. We have $d_{ii} \geqq 0$, $i = 1, 2, \cdots, n$. Suppose that there exists $w \in \mathbb{R}^n$ such that

$$(\Omega + D)w = 0;$$

then

$$\langle w, \Omega w\rangle + \langle w, Dw\rangle = 0,$$

and therefore

$$\langle w, \Omega w\rangle = 0, \tag{17}$$

$$\langle w, Dw\rangle = 0. \tag{18}$$

From (17) we see that w is an eigenvector of Ω corresponding to the eigenvalue 0. This implies (see [6]) the existence of $\alpha, \beta \in \mathbb{R}$ such that

$$w_i = \alpha + \beta t_i, \qquad i = 1, 2, \cdots, n. \tag{19}$$

Substituting (19) into (18) we get

$$\alpha^2 \sum_{i=1}^{n} d_{ii} + 2\alpha\beta \sum_{i=1}^{n} d_{ii}t_i + \beta^2 \sum_{i=1}^{n} d_{ii}t_i^2 = 0.$$

If $\alpha = 0$, we obviously have $\beta = 0$ and $w = 0$. Assume $\alpha \neq 0$. Then

$$\sum_{i=1}^{n} d_{ii} + 2\lambda \sum_{i=1}^{n} d_{ii}t_i + \lambda^2 \sum_{i=1}^{n} d_{ii}t_i^2 = 0, \tag{20}$$

where $\lambda = \beta/\alpha$.
A necessary and sufficient condition for the existence of a real number λ satisfying (20) is that

$$\left(\sum_{i=1}^{n} d_{ii}t_i\right)^2 - \left(\sum_{i=1}^{n} d_{ii}\right)\left(\sum_{i=1}^{n} d_{ii}t_i^2\right) \geqq 0,$$

or, equivalently,

$$\left(\sum_{i=1}^{n} d_{ii}\right)\left(\sum_{i=1}^{n} d_{ii}t_i^2\right) \leqq \left(\sum_{i=1}^{n} d_{ii}t_i\right)^2.$$

But the Schwarz inequality gives us

$$\left(\sum_{i=1}^{n} d_{ii}\right)\left(\sum_{i=1}^{n} d_{ii}t_i^2\right) \geqq \left(\sum_{i=1}^{n} d_{ii}t_i\right)^2.$$

Then there exists $\gamma \in \mathbb{R}$ such that

$$\sqrt{d_{ii}} = \gamma\sqrt{d_{ii}}\, t_i.$$

If $d_{kk} \neq 0$ and $d_{jj} \neq 0$ we have

$$1 = \gamma t_k, \qquad 1 = \gamma t_j,$$

which is a contradiction because $t_k \neq t_j$.

On the other hand, if only one element of D is different from zero, then there exist α, β and $\Omega + D$ is only positive semidefinite. This concludes the proof. □

If $\Omega + D$ is positive definite, $x^{(k+1)}$ is the only solution of the system

$$2\Omega x + \Psi'(x^{(k)} - z) + 2D(x - x^{(k)}) = 0,$$

which is equivalent to

$$(\Omega + D)x = Dx^{(k)} - \tfrac{1}{2}\Psi'(x^{(k)} - z). \tag{21}$$

Since Ω is a full symmetric matrix, we perform $O(n^3)$ arithmetical operations in computing the solution of (21). Moreover, this system is very ill-conditioned because of the distribution of the eigenvalues of Ω (see [12]). We then seek an equivalent system as follows:

Let s be the cubic spline taking values $x_1, x_2, \cdots, x_n$ at the knots; i.e.,

$$s(t_i) = x_i, \qquad i = 1, 2, \cdots, n. \tag{22}$$

Let

$$d = \Omega x. \tag{23}$$

It is well known (see [9]) that

$$d_i = s'''(t_i^+) - s'''(t_i^-), \qquad i = 1, 2, \cdots, n. \tag{24}$$

To find an equivalent system we proceed as in Paihua [9]. Let λ_i, $i = 1, 2, \cdots, n$ be the derivative of s at t_i. By a straightforward computation we get

$$\begin{aligned} d_1 &= 6\frac{\lambda_2 - 2m_1 + \lambda_1}{h_1^2}, \\ d_i &= 6\left[\frac{\lambda_{i+1} - 2m_i + \lambda_i}{h_i^2} - \frac{\lambda_i - 2m_{i-1} + \lambda_{i-1}}{h_{i-1}^2}\right], \qquad i = 2, \cdots, n-1, \\ d_n &= -6\frac{\lambda_n - 2m_{n-1} + \lambda_{n-1}}{h_{n-1}^2}, \end{aligned} \tag{25}$$

where

$$m_i = \frac{x_{i+1} - x_i}{h_i}, \qquad h_i = t_{i+1} - t_i, \qquad i = 2, \cdots, n-1.$$

In addition it is shown in [9] that $\lambda_1, \cdots, \lambda_n$ must satisfy

$$\begin{aligned} 2\lambda_1 + \lambda_2 &= 3m_1, \\ \frac{h_i}{h_{i-1}}\lambda_{i-1} + 2\left(1 + \frac{h_i}{h_{i-1}}\right)\lambda_i + \lambda_{i+1} &= 3\left(m_i + \frac{h_i}{h_{i-1}}m_{i-1}\right), \qquad i = 2, \cdots, n-1, \\ \lambda_{n-1} + 2\lambda_n &= 3m_{n-1}. \end{aligned} \tag{26}$$

Substituting (25) into (21) and using (26) we get the following block tridiagonal linear system:

$$\begin{bmatrix} A_1 & B_1 & & & & \\ C_2 & A_2 & B_2 & & & \\ & C_3 & A_3 & B_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & B_{n-1} \\ & & & & C_n & A_n \end{bmatrix} \begin{bmatrix} x_1 \\ \lambda_1 \\ x_2 \\ \lambda_2 \\ \vdots \\ x_n \\ \lambda_n \end{bmatrix} = \begin{bmatrix} D_1 \\ \cdot \\ \cdot \\ \cdot \\ \\ D_n \end{bmatrix},$$

where

$$A_1=\begin{bmatrix} \dfrac{3}{h_1} & 2 \\ \dfrac{2}{h_1}+\dfrac{\rho_1 h_1^2}{6} & 1 \end{bmatrix}, \qquad A_i=\begin{bmatrix} 3\left(\dfrac{1}{h_i}-\dfrac{h_i}{h_{i-1}^2}\right) & 2\left(\dfrac{h_i}{h_{i-1}}+1\right) \\ \dfrac{\rho_i h_i^2}{6}+\dfrac{2}{h_{i-1}}+\dfrac{2}{h_{i-1}}\left(\dfrac{h_i}{h_{i-1}}\right)^2 & 1-\left(\dfrac{h_i}{h_{i-1}}\right)^2 \end{bmatrix},$$

$$i=2,3,\cdots,n-1,$$

$$A_n=\begin{bmatrix} -\dfrac{3}{h_{n-1}} & 2 \\ -\left(\dfrac{3}{h_{n-1}}+\dfrac{\rho_n h_{n-1}^2}{6}\right) & 1 \end{bmatrix},$$

$$B_1=\begin{bmatrix} -\dfrac{3}{h_1} & 1 \\ -\dfrac{2}{h_1} & 1 \end{bmatrix}, \qquad B_i=\begin{bmatrix} 3\dfrac{h_i}{h_{i-1}^2} & 1 \\ -\dfrac{2}{h_i} & 1 \end{bmatrix}, \qquad i=2,\cdots,n-1$$

$$C_i=\begin{bmatrix} \dfrac{3h_i}{h_{i-1}^2} & 1 \\ -\dfrac{2}{h_{i-1}}\left(\dfrac{h_i}{h_{i-1}}\right)^2 & -\left(\dfrac{h_i}{h_{i-1}}\right)^2 \end{bmatrix}, \quad i=2,\cdots,n-1, \qquad C_n=\begin{bmatrix} \dfrac{3}{h_{n-1}} & 1 \\ \dfrac{2}{h_{n-1}} & 1 \end{bmatrix}$$

$$D_i=\begin{bmatrix} 0 \\ \dfrac{h_i^2}{6}w_i^{(k)} \end{bmatrix}, \qquad i=1,\cdots,n-1, \qquad D_n=\begin{bmatrix} 0 \\ -\dfrac{h_{n-1}^2}{6}w_i^{(k)} \end{bmatrix}$$

and

$$w_i^{(k)}=d_{ii}x_i^{(k)}-\tfrac{1}{2}\Psi'(x_i^{(k)}-z), \qquad i=1,2,\cdots,n.$$

This linear system can be solved by block Gauss elimination. For n greater than 3, the number of multiplications and divisions necessary to solve it is $36n-14$, instead of the $O(n^3)$ operations used by a Cholesky decomposition applied to the original system.

This reduction in the cost of a single iteration allows us to apply Newton's method successfully. Moreover, the stability of the calulations is no longer a problem.

In Table 1 we give typical run times on an IBM 370/145 computer. To stop iteration we use the criterion of maximum error less than 10^{-8}.

TABLE 1

n	Time (sec)
20	0.05
40	0.30
60	0.40
80	0.60
100	0.80

4. Convergence results. In this section, we choose an appropriate sequence of penalty functions Ψ_k and show that the penalty method converges in this case.

First, we recall a classical result (see [5]).

THEOREM 2. *If n is greater than 2, there exists one and only one solution $\bar{y}$ to the problem*

$$\text{Minimize}_{|y_i - z_i| \leqq \varepsilon} \langle y, \Omega y \rangle, \qquad i = 1, 2, \cdots, n. \tag{31}$$

Furthermore, if we set $\lambda = \Omega \bar{y}$, a necessary and sufficient condition for $\bar{y}$, λ to be a solution of (31) *is that*

(i) $$z_i - \varepsilon \leqq \bar{y}_i \leqq z_i + \varepsilon, \qquad i = 1, 2, \cdots, n,$$

(ii) $$\lambda_i \geqq 0 \quad \text{if } \bar{y}_i = z_i - \varepsilon,$$
$$\lambda_i \leqq 0 \quad \text{if } \bar{y}_i = z_i + \varepsilon,$$
$$\lambda_i = 0 \quad \text{if } z_i - \varepsilon < \bar{y}_i < z_i + \varepsilon.$$

Proof. See Laurent [5]. □

Let $\bar{y}^{(k)}$ be the solution of the problem

$$\text{Minimize}_{y \in \mathbb{R}^n} \left\{ \langle y, \Omega y \rangle + \sum_{i=1}^{n} \Psi_k(y_i - z_i) \right\}, \tag{32} (P_k)$$

where the functions Ψ_k are defined by

$$\Psi_k(u) = \frac{1}{2k} \left(\frac{u}{\varepsilon} \right)^{2k}. \tag{33}$$

It is clear that $\Psi_k(u)$ converges to $\chi(u)$ pointwise; i.e.,

$$\lim_{k \to \infty} \Psi_k(u) = \begin{cases} 0, & |u| \leqq \varepsilon, \\ +\infty, & |u| > \varepsilon. \end{cases} \tag{34}$$

We will now prove that $\bar{y}^{(k)}$ converges to $\bar{y}$ as k increases.

Define $\lambda^{(k)} = \Omega \bar{y}^{(k)}$. The following lemma gives us information about the behavior of $\bar{y}^{(k)}$ as k increases.

LEMMA 1. *The following inequalities are satisfied*:

(a) $$|y_i^{(k)} - z_i| \leqq \varepsilon + \frac{2\varepsilon^2}{2k-1} |\lambda_i^{(k)}| \quad \text{for } k \geqq 2.$$

(b) $$|y_i^{(k)} - z_i| \leqq 2(\|z\|_\infty + \varepsilon) \quad \text{for sufficiently large } k.$$

Proof.

(a) From § 2 we know that $\lambda^{(k)}$, $\bar{y}^{(k)}$ satisfy

$$2\lambda_i^{(k)} + \Psi_k'(\bar{y}^{(k)} - z_i) = 0, \qquad i = 1, 2, \cdots, n. \tag{35}$$

From (33) we have

$$\Psi_k'(u) = \frac{1}{\varepsilon} \left(\frac{u}{\varepsilon} \right)^{2k-1}$$

and

$$(\Psi_k')^{-1}(v) = \varepsilon (\varepsilon v)^{1/(2k-1)}. \tag{36}$$

Let ϕ be defined as

$$\phi(v) = (\Psi_k')^{-1}(v);$$

with this notation (35) becomes

$$\bar{y}_i^{(k)} - z_i = -\phi(2\lambda_i^{(k)}), \qquad i = 1, 2, \cdots, n.$$

Hence

$$|\bar{y}_i^{(k)} - z_i| = |\phi(2\lambda_i^{(k)})| = \phi(2|\lambda_i^{(k)}|). \tag{37}$$

Consider now two cases:

Case 1. $2|\lambda_i^{(k)}| \leqq 1/\varepsilon$. The function ϕ being nondecreasing, we have

$$\phi(2|\lambda_i^{(k)}|) \leqq \phi\left(\frac{1}{\varepsilon}\right) = \varepsilon \leqq \varepsilon + \frac{2\varepsilon^2}{2k-1}|\lambda_i^{(k)}|. \tag{38}$$

Case 2. $2|\lambda_i^{(k)}| \geqq 1/\varepsilon$. Using the fact that ϕ is concave on the positive real line, we have

$$\phi(\alpha) \leqq \phi\left(\frac{1}{\varepsilon}\right) + \phi'\left(\frac{1}{\varepsilon}\right)\left(\alpha - \frac{1}{\varepsilon}\right), \qquad \alpha > 0;$$

hence

$$\begin{aligned}
\phi(2|\lambda_i^{(k)}|) &\leqq \varepsilon + \frac{\varepsilon^2}{2k-1}\left(2|\lambda_i^{(k)}| - \frac{1}{\varepsilon}\right) \\
&\leqq \varepsilon\left(1 - \frac{1}{2k-1}\right) + \frac{2\varepsilon^2}{2k-1}|\lambda_i^{(k)}| \\
&\leqq \frac{2k-2}{2k-1}\varepsilon + \frac{2\varepsilon^2}{2k-1}|\lambda_i|^k \\
&\leqq \varepsilon + \frac{2\varepsilon^2}{2k-1}|\lambda_i|^k \quad \text{with } 2|\lambda_i^{(k)}| \geqq \frac{1}{\varepsilon},
\end{aligned}$$

which concludes the proof of part (a).

(b) We have $\lambda^{(k)} = \Omega\bar{y}^{(k)}$. Then

$$\|\lambda^{(k)}\|_\infty \leqq \|\Omega\| \, \|\bar{y}^{(k)}\|_\infty,$$

where

$$\|x\|_\infty = \max\{|x_1|, |x_2|, \cdots, |x_n|\}.$$

Replacing this inequality in the inequality obtained in (a), we obtain

$$|y_i^{(k)} - z_i| \leqq \varepsilon + \frac{2\varepsilon^2}{2k-1}\|\Omega\| \, \|\bar{y}^{(k)}\|_\infty$$

and

$$\|\bar{y}^{(k)}\|_\infty \leqq \|z\|_\infty + \varepsilon + \frac{2\varepsilon^2}{2k-1}\|\Omega\| \, \|\bar{y}^{(k)}\|_\infty.$$

We now choose K such that for $k \geqq K$

$$\frac{2\varepsilon^2\|\Omega\|}{2k-1} < \frac{1}{2},$$

and conclude that for $k \geqq K$:

$$\|\bar{y}^{(k)}\|_\infty \leqq \|z\|_\infty + \varepsilon + \tfrac{1}{2}\|\bar{y}^{(k)}\|_\infty$$

or

$$\|\bar{y}^{(k)}\|_\infty \leqq 2(\|z\|_\infty + \varepsilon);$$

the proof of part (b) and of Lemma 1 are complete. □

We are now ready to state the main result of this section.

THEOREM 3. *Let $\bar{y}$ be the solution of the problem* (P),

$$\text{(P)} \qquad \underset{|y_i - z_i| \leqq \varepsilon}{\text{Minimize}} \langle y, \Omega y \rangle, \qquad i = 1, 2, \cdots, n,$$

and $\bar{y}^k$ the solution of the problem (P_k)

$$(\mathrm{P}_k) \qquad \underset{y \in \mathbb{R}^n}{\text{Minimize}} \left\{ \langle y, \Omega y \rangle + \sum_{i=1}^{n} \Psi_k(y_i - z_i) \right\},$$

where Ψ_k is defined by

$$\Psi_k(u) = \frac{1}{2k}\left(\frac{1}{\varepsilon} u\right)^{2k}.$$

Then

$$\lim_{k \to \infty} \bar{y}^{(k)} = \bar{y}.$$

Proof. We first prove that any convergent subsequence of $\{\bar{y}^{(k)}\}$ converges to $\bar{y}$. Then we see how this implies that $\{\bar{y}^{(k)}\}$ converges to $\bar{y}$.

(a) Let $\{\tilde{y}^{(k)}\}$ be a convergent subsequence of $\{\bar{y}^{(k)}\}$ and let $\tilde{y}$ be its limit. Then

$$\tilde{\lambda} = \Omega \tilde{y} = \lim_{k \to \infty} \Omega \tilde{y}^{(k)} = \lim_{k \to \infty} \tilde{\lambda}^{(k)}.$$

(a.1) If $\tilde{y}_i = z_i - \varepsilon$, for $k \geqq K_1$ we have $\tilde{y}_i^{(k)} \leqq z_i$ and

$$\tilde{\lambda}_i^{(k)} = -\tfrac{1}{2}\Psi_k'(y_i^{(k)} - z_i) \geqq 0 \quad \text{for all } k \geqq K_1,$$

so that $\tilde{\lambda}_i \geqq 0$.

(a.2) If $\tilde{y}_i = z_i + \varepsilon$ for $k \geqq K_2$, we have $\tilde{y}_i^{(k)} \geqq z_i$, and

$$\lambda_i^{(k)} = -\tfrac{1}{2}\Psi_k'(y_i^{(k)} - z_i) \leqq 0 \quad \text{for all } k \geqq K_2,$$

so that $\tilde{\lambda}_i \leqq 0$.

(a.3) If $\tilde{y}_i \in]z_i - \varepsilon, z_i + \varepsilon[$ for $k \geqq K_3$, we have

$$z_i - \frac{\varepsilon}{2} \leqq y_i^{(k)} \leqq z_i + \frac{\varepsilon}{2},$$

and $\tilde{\lambda}_i^{(k)} = -\tfrac{1}{2}\Psi_k'(\tilde{y}_i^{(k)} - z_i)$ will converge to zero as k increases, because Ψ_k' converges uniformly to zero on any compact subset of $]-\varepsilon, \varepsilon[$. Hence $\tilde{\lambda}_i = 0$.

(a.4) Following Lemma 1(a) we have

$$|\tilde{y}_i^{(k)} - z_i| \leqq \varepsilon + \frac{2\varepsilon^2}{2k-1}|\tilde{\lambda}_i^{(k)}|;$$

since this is bounded, we obtain

$$|\tilde{y}_i - z_i| \leqq \varepsilon, \qquad i = 1, 2, \cdots, n.$$

We have thus shown that $(\tilde{y}, \tilde{\lambda})$ satisfy the necessary and sufficient conditions of Theorem 2, and so $\bar{y} = \tilde{y}$.

(b) From Lemma 1, we know that $\bar{y}^{(k)}$ is bounded and this implies the existence of at least one convergent subsequence. From (a) we know that its limit is $\bar{y}$; moreover, all accumulation points of $\bar{y}^{(k)}$ coincide with $\bar{y}$, so $\{\bar{y}^{(k)}\}$ converges to $\bar{y}$. □

COROLLARY. *If σ is the solution of* (3) *and $\sigma^{(k)}$ the solution of* $(6)_k$, *we have*

$$\lim_{k\to\infty} \sigma^{(k)} = \sigma,$$

in the sense of the norm of $H^q[a, b]$.

Proof. We know (cf. [5]) that the linear operator $\mathcal{S}$ which associates σ to $\bar{y}$ is continuous. Hence

$$\lim_{k\to\infty} \sigma^{(k)} = \lim_{k\to\infty} \mathcal{S}(\bar{y}^{(k)}) = \mathcal{S}(\bar{y}) = \sigma.$$

Numerical results. We have performed computations on artificial data obtained by rounding the values of a known function f at n different points of the real interval $[-1, 1]$. In Table 2 we tabulate run times obtained on an IBM 370/145 computer and compare them with the DUAL algorithm as programmed by C. Di Crescenzo [2]. All computer times of the dual algorithm are taken from [2]; Di Crescenzo used an IBM 370/67 computer, so we have divided those times by 3 in order to obtain the run times in our computer.

We can observe the great difference existing between run times. We can also see that run times for the penalty function method are approximately linear in n.

TABLE 2
Run times (secs)

n	Penalty	Dual
20	1.35	3.5
40	4.54	18.6
60	8.12	52.0
80	11.39	83.6
100	15.19	129.0

To illustrate the behavior of the spline restricted to smooth data containing roundoff errors, we ran the program for 100 points in the interval $[-1, 1]$. The test function is

$$\cos\frac{\pi}{2}x - 0.5\cos\frac{3\pi}{2}x$$

and the rounding error ε is 0.25.

In Fig. 1 we plot the data (marked by an x); the test function (marked by an F) and the resulting restricted spline, calculated by the penalty function method (marked by an S). We can see the remarkable approximating properties of these spline functions.

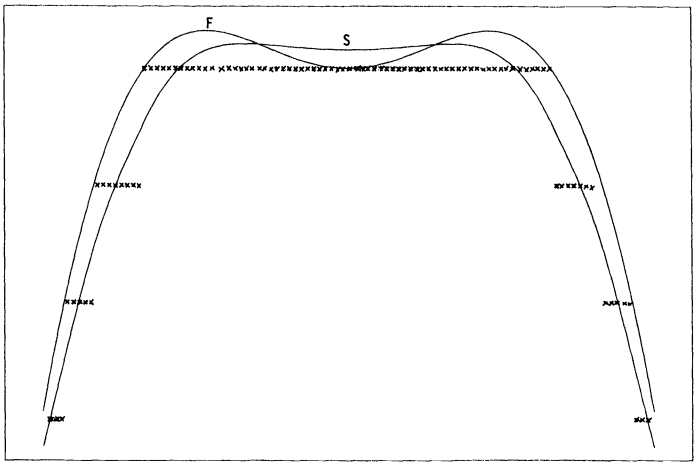


FIG. 1

Acknowledgments. The author wishes to express his appreciation to Prof. P. J. Laurent from the University of Grenoble with whom he began the study of robust splines.

REFERENCES

[1] R. H. BYRD AND D. A. PYNE, *Convergence of the Iteratively Reweighted Least Squares Algorithm for Robust Regression*, Johns Hopkins University, Technical Report 313, June, 1979.

[2] CLAIRE DI CRESCENSO AND P. J. LAURENT, *Etude du développement des fonctions spline de lissage pour des données de télémesure, Vol.* III, *Résultats numériques*, IMAG, Grenoble, 1976.

[3] P. HUBER, *Robust smoothing*, in Robustness in Statistics, Proceedings of the ARO Workshop, R. Launder and G. Wilkinson, eds., Academic Press, New York, 1979.

[4] B. KRUMM AND TH. GASSER, *Robust spline smoothing*, Communication at the Workshop held at Heidelberg on Smoothing Techniques for Curve Estimation, April, 1979.

[5] P. J. LAURENT, *Approximation et Optimisation*, Hermann, Paris, 1972.

[6] ———, *Etude du développement des fonctions spline de lissage pour des données de télémesure, Vol.* I, IMAG, Grenoble, 1976.

[7] ———, *Etude du développement des fonctions spline de lissage pour des données de télémesure, Vol.* II, *Méthodes numériques*, IMAG, Grenoble, 1976.

[8] R. V. LENTH, *Robust splines*, Comm. Statist. A6 (1977), pp. 847–854.

[9] L. PAIHUA, *Quelques méthodes numériques pour le calcul de fonctions spline à une et plusieurs variables*, Docteur de Spécialité, Thesis, Grenoble, 1978.

[10] R. T. ROCKAFELLAR, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[11] F. UTRERAS, *Funciones spline asociadas a una función ϕ*, SIGMA 4 (1978), pp. 1–7.

[12] ———, *Sur la choix du paramètre d'ajustement dans le lissage par fonctions spline*, Numer. Math., 34 (1980), pp. 15–28.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0005 $01.00/0

NUMERICAL SOLUTION OF A QUADRATIC MATRIX EQUATION*

GEORGE J. DAVIS†

Abstract. This paper is concerned with the efficient numerical solution of the matrix equation $AX^2+BX+C=0$, where A, B, C and X are all square matrices. Such a matrix X is called a *solvent*. This equation is very closely related to the problem of finding scalars λ and nonzero vectors x such that $(\lambda^2A+\lambda B+C)x=0$. The latter equation represents a *quadratic eigenvalue problem*, with each λ and x called an eigenvalue and eigenvector, respectively. Such equations have many important physical applications.

By presenting an algorithm to calculate solvents, we shall show how the eigenvalue problem can be solved as a byproduct. Some comparisons are made between our algorithm and other methods currently available for solving both the solvent and eigenvalue problems. We also study the effects of rounding errors on the presented algorithm, and give some numerical examples.

Key words. matrix equations, eigenvalues, solvents, Newton's method

1. Overview. The quadratic eigenvalue problem has been studied for some time. Lancaster [11] provides a good introduction to the subject, and explores the applications to vibrating systems. Far less is known about general matrix equations. Dennis, Traub and Weber [4], [5] consider general matrix polynomials, and present two algorithms for the calculation of solvents. The first is a generalization of a scalar algorithm of Traub, and the second is a generalization of Bernoulli's method. These algorithms apply to polynomials of arbitrary degree, and produce only a "dominant" solvent. Solvent S_1 is said to *dominate* solvents $S_2, \cdots, S_k$ if all the eigenvalues of S_1 are greater in modulus than all the eigenvalues of the other S_i.

This work is concerned only with quadratic polynomials, and the dominance of solvents is not required. The algorithm itself is Newton's method applied to the matrix equation $F(X)=AX^2+BX+C=0$. Critical to the algorithm is the method of calculating the correction $(F'(X_i))^{-1}F(X_i)$. Much of the work is done by a complex version of the QZ algorithm [8], [12], [15]. Once a solvent S is found, it is clear that $F(\lambda I)=\lambda^2A+\lambda B+C$ has the following factorization:

$$F(\lambda I)=(-B-AS-\lambda A)(S-\lambda I). \tag{1}$$

The eigenvalues of the original quadratic, then, are the union of the set of eigenvalues of S and the set of eigenvalues of

$$(-B-AS)x=\lambda Ax. \tag{2}$$

Equation (2) represents a *generalized eigenvalue problem*, and it can also be solved by the QZ. The factorization in (1) is a reason why S is sometimes called a *right-solvent* of $F(X)$. Corresponding theory exists for *left-solvents*, i.e., left factors in (1). We will consider only right-solvents, and call them, briefly, *solvents*.

Discussion of the QZ algorithm immediately brings to mind an algorithm for solving $AX^2+BX+C=0$. Consider the λ-matrix problem $(\lambda^2+\lambda B+C)x=0$ and use the substitution $y=\lambda x$ to get

$$\lambda\begin{pmatrix}0 & A\\ I & 0\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}+\begin{pmatrix}C & B\\ 0 & -I\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}. \tag{3}$$

* Received by the editors December 27, 1979.
† Department of Mathematics, Georgia State University, Atlanta, Georgia 30303.

Use the QZ algorithm to produce the eigenvalues λ and eigenvectors $(x, y)^T$ of length $2n$. Choose an independent set of n vectors from the set of x's just computed, and assemble them as columns into a matrix W. If Λ is the diagonal matrix of corresponding eigenvalues, we have

$$AW\Lambda^2+BW\Lambda+CW=0, \tag{4}$$

or

$$A(W\Lambda W^{-1})^2+B(W\Lambda W^{-1})+C=0. \tag{5}$$

Knowing the eigenvalues and having an independent set of eigenvectors, makes it possible to construct a solvent. After one solvent is found, the factorization (2) can be used to deflate the problem.

Although a plausible algorithm on some problems, this method fails when it is impossible to choose an independent set of vectors to form W. Consider the problem

$$X^2+X+\begin{pmatrix}-6 & -5\\ 0 & -6\end{pmatrix}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix} \tag{6}$$

and the associated λ-matrix problem

$$\begin{pmatrix}\lambda^2+\lambda-6 & -5\\ 0 & \lambda^2+\lambda-6\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}. \tag{7}$$

There are two solvents $\left(\begin{smallmatrix}2 & 1\\ 0 & 2\end{smallmatrix}\right)$ and $\left(\begin{smallmatrix}-3 & -1\\ 0 & -3\end{smallmatrix}\right)$, each with a Jordan block of size two. There is only one eigenvector associated with (7), namely $(1, 0)^T$. The QZ algorithm on the system (3) produces vectors of the form $(1, 0, \lambda, 0)^T$ for $\lambda=2$ or -3. Thus it is impossible to construct a solvent of the form $W\Lambda W^{-1}$.

2. Perturbation theory. Before we can begin any analysis of a computer algorithm to find solvents of $F(X)$, we must first understand the nature of the problem itself. When the matrices A, B and C are read into a computer, certain inaccuracies result due to the finite precision of the machine. It is the intent of this section to assess what effects these errors can have on a computed solution. No mention will be made of our algorithm, as these topics concern the problem and not the method used to solve it.

In what follows, the derivative of $F(X)$ is needed. F is a function from $R^{n\times n}\to R^{n\times n}$, and its Fréchet derivative is an operator. This operator is described by what it does to a typical element $H\in R^{n\times n}$. We employ the notation $F'(X)H$ to mean "the operator $F'(X)$ applied to H". It is easy to show that

$$F'(X)H=(AX+B)H+AHX. \tag{8}$$

It is clear that the derivative, and hence its inverse, is a linear operator. Indeed, the derivative can be expressed as an $n^2\times n^2$ matrix. In fact,

$$F'(X)H=[(AX+B)\otimes I+A\otimes X^T]\,h, \tag{9}$$

where $Y\otimes Z$ is a block matrix with its (i, j) block equal to $Y_{ij}Z$, and h is the vector $[h_{11}, h_{12}, \cdots, h_{1n}, h_{21}, \cdots, h_{nn}]^T$.

We now examine the effect of small perturbations in the original data on the solvents of $F(X)$. Such perturbations can arise from many sources, including errors in measurement and inaccuracy involved by generating matrices in a computer. Throughout this discussion, ε is assumed to be some small positive number.

Let X be an exact solvent of $F(X)$, and let $\hat{X} = X + \Delta X$ be a solvent of the perturbed equation $\hat{A}X^2 + \hat{B}X + \hat{C} = 0$. Here, $\hat{A} = A + E$, $\hat{B} = B + F$, $\hat{C} = C + G$ with $\|E\| \leqq \varepsilon \|A\|$, $\|F\| \leqq \varepsilon \|B\|$ and $\|G\| \leqq \varepsilon \|C\|$.

If

$$F(X) = AX^2 + BX + C$$

and

$$\hat{F}(X) = (A+E)X^2 + (B+F)X + (C+G),$$

then define

$$\Delta F(X) \equiv \hat{F}(X) - F(X) = EX^2 + FX + G.$$

Similarly, if

$$F'(X)H = (AX + B)H + AHX$$

and

$$\hat{F}'(X)H = [(A+E)X + (B+F)]H + (A+E)HX,$$

then define

$$\Delta F'(X)H \equiv \hat{F}'(X)H - F'(X)H = (EX + F)H + EHX.$$

Under the assumption

$$\|F'(X)^{-1}\| \cdot \|\Delta F'(X)\| \leqq k < 1, \tag{10}$$

we get the following bound on the error in X.

THEOREM. *For sufficiently small* ε, $\|\Delta X\| \leqq \gamma\varepsilon$, *where*

$$\alpha = \frac{2}{1-k}\|F'(X)^{-1}\|\,[\|A\| \cdot \|X\|^2 + \|B\| \cdot \|X\|],$$

$$\beta = \frac{1+\varepsilon}{1-k}\|F'(X)^{-1}\| \cdot \|A\|,$$

$$\gamma = \frac{2\alpha}{1 + \sqrt{1 - 4\alpha\beta\varepsilon}} = \alpha + O(\varepsilon).$$

Proof. First note that $\hat{F}(\hat{X}) = 0$, or

$$(A+E)(X+\Delta X)^2 + (B+F)(X+\Delta X) + (C+G) = 0. \tag{11}$$

Expanding (6) gives

$$F'(X)\Delta X + \Delta F'(X)\Delta X + \Delta F(X) + (A+E)(\Delta X)^2 = 0. \tag{12}$$

Premultiplying by $F'(X)^{-1}$ and using assumption (10), it is easy to verify

$$\|\Delta X\| \leqq \frac{\|F'(X)^{-1}\|}{1-k}\,[\|\Delta F(X) + (A+E)(\Delta X)^2\|]. \tag{13}$$

Further, recalling the definitions of $\Delta F(X)$, E, F and G, use the fact that

$$\|C\| \leqq \|A\| \cdot \|X\|^2 + \|B\| \cdot \|X\| \tag{14}$$

to get

$$\|\Delta X\| \leqq \frac{\|F'(X)^{-1}\|}{1-k}[2\varepsilon(\|A\| \cdot \|X\|^2 + \|B\| \cdot \|X\|) + (1+\varepsilon)\|A\| \cdot \|\Delta X\|^2], \tag{15}$$

or

$$\|\Delta X\| \leqq \alpha\varepsilon + \beta\|\Delta X\|^2. \tag{16}$$

As this is a quadratic inequality in $\|\Delta X\|$, it can readily be solved to yield the desired result.

This analysis breaks down if $F'(X)$ is singular. A singular $F'(X)$ represents an extreme case of sensitivity in that small perturbations in the data can produce unbounded changes in the solvents. There is no question here of subroutine accuracy or of errors in computation; these are fundamental properties of the problem itself.

As an example, consider the matrix equation

$$\begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix} X^2 + \begin{pmatrix} -8 & -4 \\ -8 & -4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{17}$$

This equation has an infinite number of solvents generated by the equations

$$X^2 = \begin{pmatrix} y_1 & y_2 \\ y_3 & y_4 \end{pmatrix}, \qquad \begin{aligned} 2y_1 + y_3 &= 8, \\ 2y_2 + y_4 &= 4. \end{aligned}$$

The derivative operator may be expressed as the 4×4 matrix

$$F'(X)H = \begin{pmatrix} 4x_{11}+x_{21} & 2x_{21} & 2x_{12}+x_{22}+x_{11} & x_{21} \\ 2x_{12} & 2x_{11}+x_{21}+2x_{22} & x_{12} & 2x_{12}+2x_{22} \\ 4x_{11}+x_{21} & 2x_{21} & 2x_{12}+x_{22}+x_{11} & x_{21} \\ 2x_{12} & 2x_{11}+x_{21}+2x_{22} & x_{12} & 2x_{12}+2x_{22} \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{21} \\ h_{22} \end{pmatrix},$$

which is singular.

If we perturb (17) into

$$\begin{pmatrix} 2 & 1 \\ 2 & 1 \end{pmatrix} X^2 + \begin{pmatrix} -8-\varepsilon & -4 \\ -8 & -4 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \tag{18}$$

setting up the equations for X shows that *no* solvents exist.

Two things can be concluded from this study of perturbations. First, *any* algorithm can expect difficulty on problems with a singular or nearly singular derivative. Second, any well-designed algorithm should be able to detect such a singular derivative, and not produce meaningless answers. After the algorithm is described, its performance on such examples will be analyzed.

3. Subroutine SQUINT. SQUINT stands for Solving the Quadratic by Iterating Newton Triangularizations. It is Newton's method applied to the matrix function $F(X)$. After an initial guess X_0 is chosen, successive iterates are generated by the formula

$$X_{i+1} = X_i - T_i, \tag{19}$$

where

$$T_i = F'(X_i)^{-1}F(X_i), \qquad i = 0, 1, 2, \cdots. \tag{20}$$

The correction T_i is given as the solution of the system

$$(AX_i + B)T_i + AT_iX_i = F(X_i), \qquad i = 0, 1, 2, \cdots. \tag{21}$$

The following algorithm for the computation of T_i is proposed. Using unitary transformations, simultaneously reduce (AX_i+B) and A to upper triangular form, then reduce X_i to lower triangular form. Solve the transformed triangular system and translate back.

The idea of triangularizing such a system first appeared in a paper by Bartels and Stewart [1] in connection with the equation $AX+XB=C$. Golub, Nash and Van Loan [9] describe some modifications to the algorithm in [1] and mention problems of the form $AXB+CXD=F$. We have put the triangularization idea to work for the system (21).

More specifically, the algorithm is:

(1) Find a unitary Q and Z such that both $Q(AX_i+B)Z=U_1$ and $Q(A)Z=U_2$ are upper triangular.

(2) Find a unitary R such that $R^H(X_i)R=L$ is lower triangular. Note that at this point the system (21) can be written

$$(Q^HU_1Z^H)T_i+(Q^HU_2Z^H)T_i(RLR^H)=F(X_i).$$

(3) Solve for $Y_i=Z^HT_iR$ in the system $U_1Y_i+U_2Y_iL+QF(X_i)R$.

(4) Translate back $T_i=ZY_iR^H$.

To accomplish step 1 a complex version of the QZ algorithm [8], [12], [15] is used. The matrix Z must be saved for the translation step 4. Matrix Q need not be saved, but in addition to applying it to (AX_i+B) and A, it must also be applied to $F(X_i)$. The QZ program was modified to accept an auxiliary matrix F and to form QF.

The details of step 2 are slightly more complicated. To reduce the number of subroutines, the reduction of X_i is accomplished with another call to the QZ subroutine with X_i and an identity matrix. This produces unitary matrices R and S such that

$$SX_i^HR=U_3, \qquad SIR=\tilde{I}.$$

The matrix $\tilde{I}$ is the identity with ± 1 on the diagonal. Note from the second equation above that $S=\tilde{I}R^H$. Thus

$$X_i^H=(\tilde{I}R^H)^HU_3R^H, \qquad X_i=R(U_3^H\tilde{I})R^H.$$

We apply $\tilde{I}$ to the columns of U_3^H producing another lower triangular matrix. The transformation is complete when $QF(X_i)$ is multiplied by R. The back substitution and translation in steps 3 and 4 are straightforward. The new iterate is then generated.

The operation count for one iteration is overestimated by $22n^3+9n^3\sigma+O(n^2)$. Here the number σ is the average number of iterations for the QZ algorithm to reduce a subdiagonal element to zero. These are complex operation counts and details can be found in [3]. The algorithm requires $11n^2+n$ storage locations, and at the expense of some readability this total can be cut to $9n^2+3n$.

As an attempt to reduce the total of operations, we might consider using a *modified Newton's method*:

$$X_{i+1}=X_i-F'(X_0)^{-1}F(X_i), \qquad i=0,1,2,\cdots.$$

The subscript on $F'(X_0)^{-1}$ never changes, indicating that the decompositions of (AX_0+B), A and X_0 are required only once. As in the scalar case, the quadratic convergence of the method is lost with this modification. More seriously, on several problems where Newton's method converged the modified method did not. The difficulties encountered with the modification have far outweighed any savings in work.

The solution of the triangular system warrants further investigation. The subroutine in which the solution is done has an error return which indicates an inconsistent linear system. Such inconsistency is associated with a singular or nearly singular derivative. Upon triangularization, we must backsolve the system

$$U_1 Y_i + U_2 Y_i L = \hat{F}, \tag{22}$$

where again U_1 and U_2 are upper triangular, L is lower triangular, and $\hat{F} = QF(X_i)R$. A singular derivative occurs when

$$(U_1)_{ii} + (U_2)_{ii}(L)_{jj} = 0$$

for some i and j. Just having a singular derivative is not necessarily cause for an error return, however. Recall that in the solution of the linear system $Ax = b$, a singular A may or may not be a problem. If b is in the column space of A, a solution will exist, but it need not be unique. What we detect is not singularity but inconsistency of the linear system (22).

The explicit formula for y_{ij} is given by

$$\begin{aligned}&[(U_1)_{ii} + (U_2)_{ii}(L)_{jj}]\, y_{ij} \\ &= \hat{f}_{ij} - \sum_{k=i+1}^{n} (U_1)_{ik} y_{kj} - \Big(\sum_{k=i+1}^{n} (U_2)_{ik} y_{kj} \Big) l_{jj} - \sum_{p=j+1}^{n} \sum_{k=i}^{n} (U_2)_{ik} y_{kp} l_{pj}.\end{aligned} \tag{23}$$

Inconsistency is declared when the $(U_1)_{ii} + (U_2)_{ii}(L)_{jj}$ is small *relative* to the right-hand side of (23), that is, when

$$|(U_1)_{ii} + (U_2)_{ii}(L)_{jj}| \leqq \varepsilon\, |\text{right-hand side}|.$$

Here ε is the floating point precision. If singularity but not inconsistency is present, y_{ij} is set to zero.

SQUINT is written entirely in complex arithmetic. This is done since one has little idea whether the solvents of a general problem are real or complex. Matrix equations related to certain physical systems can, however, be shown to possess all real solvents. For example, the lambda-matrix equation for an overdamped vibrating system has A, B and C real and symmetric, A and B positive definite, C nonnegative definite, and $(x^T Bx)^2 - 4(x^T Ax)(x^T Cx) > 0$ for all real vectors x. All the eigenvalues and eigenvectors in this case are real and thus so are the solvents.

Garbow [7] has written a complex version of the QZ as modified by Ward [15], which performs all computations in real arithmetic. Such a program is valuable on any system with real solvents. SQUINT is a general purpose algorithm and thus no advantage is gained by remaining in real arithmetic.

The original QZ algorithm uses Householder transformations to perform the reductions. A Householder matrix is of the form $I + vu^T$, where $v^T u = -2$, v is a multiple of u and only selected elements of u are nonzero. Such matrices are symmetric and orthogonal. The complex analogue of a Householder matrix is of the form $I + vu^H$, where $v^H u = -2$. These matrices are Hermitian and unitary. The QZ code was rewritten employing these. The end product of the QZ reduction on two matrices A_1 and A_2 is an upper triangular A_2 and "quasi-triangular" A_1; that is, A_1 is upper triangular with 2×2 blocks on the diagonal representing pairs of complex eigenvalues. The further extraction of the quotients $(A_1)_{ii}/(A_2)_{ii}$ is quite complicated. In the complex QZ case, A_1 and A_2 are always upper triangular, and the 2×2 block problem does not arise. This makes the complex code significantly shorter.

There are two fundamental questions which must be addressed, namely where to start and when to stop. A general matrix equation usually gives no hints about even the existence of a solvent. We therefore use the following initial guess, which is designed merely to provide a rough estimate for the magnitude of a possible solution:

$$X_0 = \left[\frac{\|B\| + \sqrt{\|B\|^2 + 4\|A\| \cdot \|C\|}}{2\|A\|}\right] I. \tag{24}$$

The ∞-norm is used above because of its easy computability. Note the plus signs in (24). The first plus sign is used to avoid the possibility of $X_0 = 0$. If B were also zero, such a guess produces a singular derivative and an error return. The second plus sign avoids the possibility of a complex X_0 on a real problem. It is easy to construct examples with real solvents and yet where the square root in (24) produces a complex guess.

Implemented also is a re-initialization strategy. If, in 30 iterations, the initial X_0 has failed to produce a convergent sequence of iterates, we restart with $X_{31} = (\sqrt{-1})X_0$ to introduce complex iterates. If 30 more iterations also fail, we start again with $X_{61} = C$. If all three guesses fail, we indicate so and stop. There is strong numerical evidence justifying these strategies, and some examples are presented later.

Convergence is declared when the norm of the residual $F(X_n)$ is less than the roundoff error in computing it. In the section on error analysis it is shown that the roundoff error in computing $F(X)$, where X is a solvent, is bounded by

$$\tfrac{1}{2}\varepsilon[(4\sqrt{2}\,n+2)\|A\| \cdot \|X\|^2 + (2\sqrt{2}\,n+2)\|B\| \cdot \|X\| + \|C\|] + O(\varepsilon^2).$$

The ε is again the floating-point precision.

As usual, this bound is pessimistic in the sense that it assumes the worst possible roundoff at each step.

The main theorem on the convergence of Newton's method is due to Kantorovich [10], [13].

THEOREM (Kantorovich). *If* $\|F''(X)\| \leqq K$ *in some closed ball* $\bar{U}(X_0, r)$ *and* $h_0 = (B_0)(\eta_0)(K) \leqq \frac{1}{2}$ *with* $\|F'(X_0)^{-1}\| \leqq B_0$ *and* $\|X_1 - X_0\| \leqq \eta_0$, *then the Newton sequence starting from* X_0 *will converge to a solution* X_* *of* $F(X)$ *which exists in* $\bar{U}(X_0, r)$, *provided that*

$$r \geqq r_0 = \frac{1 - \sqrt{1 - 2h_0}}{h_0} \cdot \eta_0.$$

Proof. [See 13, p. 135.] Although the theorem is a powerful theoretical result, as is usually the case in applications, it turns out to be quite restrictive for our purposes. Ideally, we would hope to use this theorem to predict the success of the algorithm starting from a given X_0. We have produced many examples where the conditions of the theorem are not satisfied, yet the algorithm converges. In fact it is rare that all the above conditions are met. Our experiments indicate that the proposed algorithm converges under much less stringent conditions.

4. Error analysis. We denote by $\mathrm{fl}(x)$ the floating point number which the computer assigns to the real number x. For rounding computers, $\mathrm{fl}(x)$ is the floating point number nearest x, and for chopping computers $\mathrm{fl}(x)$ is the first floating point number with absolute value less than or equal to x. The *unit roundoff* u is defined as the largest number m such that

$$\mathrm{fl}(1+m) = 1.$$

The rounding u is $\frac{1}{2}\varepsilon$ and the chopping u is ε. This *machine* ε is the floating point resolution of the machine, or the distance from 1 to the next largest floating point number.

Some standard assumptions are made on real floating point arithmetic. In particular, let fl (x * y) denote floating point addition, subtraction, multiplication or division. We assume that

$$\text{fl}\,(x * y) = (x * y)(1+\delta) \quad \text{with } |\delta| \leqq u.$$

Further details on real floating point arithmetic can be found in [6] and [17].

There are similar assumptions for complex arithmetic. These first appeared in Varah [14], and for convenience they are listed here. Let z_1 and z_2 be complex numbers. Then

$$\begin{aligned}
&\text{fl}\,(z_1 \pm z_2) = (z_1 \pm z_2)(1+\delta), && |\delta| \leqq u,\\
&\text{fl}\,(z_1 z_2) = (z_1 z_2)(1+\delta), && |\delta| \leqq 2\sqrt{2}u + O(u^2),\\
&\text{fl}\,(z_1 \div z_2) = (z_1 \div z_2)(1+\delta), && |\delta| \leqq 5\sqrt{2}u + O(u^2).
\end{aligned}$$

Wilkinson [17] presents an error analysis for basic matrix addition and multiplication. Under the assumptions on complex arithmetic, it is easy to verify the following.

LEMMA. *Let A and B be complex square matrices of order n. Then*

$$\begin{aligned}
&\text{fl}\,(A+B) = A+B+E_1 \quad \textit{with } \|E_1\|_F \leqq u\|A+B\|_F,\\
&\text{fl}\,(AB) = AB+E_2 \quad \textit{with } \|E_2\|_F \leqq 2\sqrt{2}nu\|A\|_F\|B\|_F + O(u^2).
\end{aligned}$$

Having this lemma, we may now establish the roundoff error in computing $F(X)$. We compute fl $(F(X))$ according to $([(AX)X]+(BX))+C$ and get the following bound.

THEOREM. *Let $F(X) = AX^2 + BX + C$. Then*

$$\begin{aligned}
\text{fl}\,(F(X)) &= F(X)+E \quad \textit{with } \|E\|_F\\
&\leqq u[(4\sqrt{2}n+2)\|A\|_F\|X\|_F^2 + (2\sqrt{2}+2)\|B\|_F\|X\| + \|C\|_F] + O(u^2).
\end{aligned}$$

In this section on perturbation theory, it was shown that roundoff errors in the data can be magnified by $\|F'(X)^{-1}\|_F$. We now prove a similar assertion about the solution of $F'(X_i)^{-1}F(X_i)$. These results are applications of results in Wilkinson [16] and their extensions in Burris [2]. Throughout, we denote computed quantities with the "hat" notation, all norms used are Frobenius norms and u is the unit roundoff. The expression $a \lesssim b$ is used to mean that a and b are on the order of u and $a \leqq b + O(u^2)$. The actual development parallels one in [9] for the problem $AX + XB = C$. Omitted details can be found in [3]. The system we wish to solve is

$$(AX_i + B)T_i + AT_iX_i = F(X_i).$$

Step 1. Use the QZ algorithm to simultaneously triangularize (AX_i+B) and A. We calculate a $\hat{U}_1$ and $\hat{U}_2$, both upper triangular, such that

$$\begin{aligned}
Q(AX_i + B + E_1)Z &= \hat{U}_1,\\
Q(A+E_2)Z &= \hat{U}_2
\end{aligned} \tag{25}$$

and

$$\begin{aligned}
\|E_1\| &\lesssim g_1(n)u\|AX_i + B\|,\\
\|E_2\| &\lesssim g_2(n)u\|A\|.
\end{aligned} \tag{26}$$

In other words, the exact triangularization is computed for a nearby pair of matrices. Also, $g_i(n)$ is a function of n whose exact form in unimportant. Details can be found in [2].

Step 2. Use the QZ or QR algorithm to triangularize X_i. A lower triangular matrix $\hat{L}$ is calculated which satisfies

$$R^H(X_i+E_3)R=\hat{L} \tag{27}$$

with

$$\|E_3\|\lesssim g_3(n)u\|X_i\|. \tag{28}$$

Step 3. Solve the triangular system. First we take a detailed look at setting up the right-hand side of the triangular system

$$\hat{U}_1(Z^HT_iR)+\hat{U}_2(Z^HT_iR)\hat{L}=\widehat{QFR}. \tag{29}$$

Before any of the triangularizations begin, we must first calculate $F=AX_i^2+BX_i+C$. From the previous analysis it is clear that $F_1=F+E_4$ is produced with $\|E_4\|\lesssim g_4(n)\,u\|F\|$. In the upper triangularization step we do not explicitly form the matrix $\hat{Q}$, but apply it to F_1 at each stage. Burris guarantees that

$$\widehat{QF}=Q(F_1+E_5)=Q(F+E_4+E_5), \tag{30}$$

with

$$\|E_5\|\lesssim g_5(n)u\|F+E_4\|. \tag{31}$$

In the lower reduction step, we actually construct the matrix of transformations $\hat{R}$, that is,

$$\hat{R}=R+E_6 \quad \text{with } \|E_6\|\lesssim g_6(n)u. \tag{32}$$

We emphasize that, although the Q and R matrices are accumulations of unitary transformations, they are handled in two completely different ways. There is never a matrix $\hat{Q}$, only the result $\widehat{QF}$. $\hat{R}$ is applied to $\widehat{QF}$ by matrix multiplication. Thus,

$$\widehat{QF}\hat{R}=Q(F+E_7)R \quad \text{with } \|E_7\|\lesssim g_7(n)u\|F\|. \tag{33}$$

Now consider the solution of the triangular system. Define $Y_i=Z^HT_iR$ and $\hat{F}=\text{fl}\,(\widehat{QF}\hat{R})$, and solve

$$\hat{U}_1Y_i+\hat{U}_2Y_i\hat{L}=\hat{F}. \tag{34}$$

Although it is not actually carried out this way, the solution of (34) is the same as the solution of an $n^2\times n^2$ system with

$$\hat{M}=(\hat{U}_1\otimes I)+(\hat{U}_2\otimes\hat{L}^T),$$
$$y=(y_{11}, y_{12}, \cdots, y_{1n}, y_{21}, \cdots, y_{nn})^T,$$
$$\hat{f}=(\hat{f}_{11}, \hat{f}_{12}, \cdots, \hat{f}_{1n}, \hat{f}_{21}, \cdots, \hat{f}_{nn})^T.$$

We have now

$$\hat{M}y=\hat{f}. \tag{35}$$

Solving (35) produces a $\hat{y}$ such that

$$(\hat{M}+E_8)\hat{y}=\hat{f} \quad \text{with } \|E_8\|\lesssim g_8(n)u\|\hat{M}\|. \tag{36}$$

Step 4. Translate back. The solution $\hat{T}_i$ is obtained by computing $\hat{Z}\hat{Y}_i\hat{R}^H$, where, as before,

$$\hat{R}^H = R^H + E_6^H, \qquad \|E_6\| \lesssim g_6(n)u,$$
$$\hat{Z} = Z + E_9, \qquad \|E_9\| \lesssim g_9(n)u.$$

Thus we compute the solution $\hat{T}_i$, where

$$\hat{T}_i = Z(Y_i + E_{10})R^H \tag{37}$$

and

$$\|E_{10}\| \lesssim g_{10}(n)u\|\hat{Y}_i\|. \tag{38}$$

We seek a bound for the relative error in T_i. Note that, since $T_i = ZY_iR^H$, the unitary invariance of the Frobenius norm guarantees that $\|T_i\| = \|Y_i\|$. Manipulation with (37) and (38) shows that

$$\frac{\|T_i - \hat{T}_i\|}{\|T_i\|} \lesssim \frac{\|Y_i - \hat{Y}_i\|}{\|Y_i\|} + g_{10}(n)u\frac{\|\hat{Y}_i\|}{\|Y_i\|}. \tag{39}$$

Thus it is critical to assess the error in solving the triangular system for Y_i. Examining some well-known analyses of the solution of triangular systems, we can again make an assumption that $\|F'(X)^{-1}\|\|\Delta F(X)\| \leqq k < 1$, to get the bound

$$\frac{\|T_i - \hat{T}_i\|}{\|T_i\|} \lesssim \frac{1}{1-k}\|F'(X_i)^{-1}\|g_{12}(n)u[5\|A\|\,\|X_i\| + 3\|B\|] + \frac{g_{10}(n)u}{1-k}. \tag{40}$$

Details on this phase of the analysis can be found in [3].

The presence of the $\|F'(X)^{-1}\|$ term throughout this paper is of fundamental importance to the study of quadratic matrix equations. In the bound above it is seen that this term has a great effect on the accuracy of the Newton correction T_i. These corrections, in turn, affect the eventual speed of convergence. Indeed, if the T_i are large enough the method may fail to converge at all.

In perturbation theory it was seen that the same $\|F'(X)^{-1}\|$ greatly affects the accuracy of a computed solvent of $F(X)$. Even if the correction T_i is computed exactly, the next iterate X_{i+1} may be large due to the size of $\|F'(X_i)^{-1}\|$. If the derivative is nearly singular, no algorithm (including ours) can expect success. It is important to note, however, that the present algorithm is stable. It does not introduce any errors which are not inherent to the problem itself.

5. Examples. To illustrate the behavior of the algorithm on various problems, we present a few examples.

Consider the example mentioned in § 1 on which the *QZ* approach fails:

$$X^2 + X + \begin{pmatrix} -6 & -5 \\ 0 & -6 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}. \tag{6}$$

There are two solvents,

$$S_1 = \begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix} \quad \text{and} \quad S_2 = \begin{pmatrix} -3 & -1 \\ 0 & -3 \end{pmatrix},$$

but only one eigenvector, $(1, 0)^T$. SQUINT converges to S_1 in 6 iterations.

In Dennis, Traub and Weber [5] are two algorithms for the calculation of dominant solvents. Solvent S_1 is said to *dominate* solvents $S_2, \cdots, S_k$ if all the eigenvalues of S_1 are greater in modulus than all the eigenvalues of the other S_i. If no solvent dominates, the algorithm in [5] may fail without further analysis of the problem.

An example is

$$X^2+\begin{pmatrix}-1 & -6\\ 2 & -9\end{pmatrix}X+\begin{pmatrix}0 & 12\\ -2 & 14\end{pmatrix}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}. \tag{41}$$

The solvents have eigenvalue pairs of 1 and 3, 1 and 2, 1 and 4, and 2 and 3, so no solvent dominates. Although [5, algorithm 1] fails, SQUINT produces

$$S=\begin{pmatrix}4 & 0\\ 2 & 2\end{pmatrix}$$

in 9 iterations. We emphasize that the work in [4] and [5] involves polynomials of arbitrary degree, whereas SQUINT was designed only for quadratics. The general degree case is much more complicated.

Finally, problems of the form

$$X^2+X+C=0 \tag{42}$$

were generated in such a way that the complex $n \times n$ Hilbert matrix is a solvent; that is, $S_{ij}=(1/(i+j-1))+\sqrt{-1}\,(1/(i+j-1))$. Table 1 gives the order of the matrix and the number of iterations required by SQUINT to converge. All calculations were due in COMPLEX*16 arithmetic on a Univac 90/80 computer.

TABLE 1

Order of the matrix	no. of iterations
1	6
2, 3, 4, 5	7
6, 7, 8, 9, 10	8

REFERENCES

[1] R. H. BARTELS AND G. W. STEWART, *A solution of the matrix equation $AX+XB=C$*, Comm. ACM, 15, pp. 820–826.

[2] C. H. BURRIS, JR., *The Numerical Solution of the Generalized Eigenvalue Problem for Rectangular Matrices*, University of New Mexico Technical Report 294, 1974.

[3] G. J. DAVIS, *Numerical Solutions of a Quadratic Matrix Equation*, Ph.D. Thesis, University of New Mexico, 1979.

[4] J. E. DENNIS, JR., J. F. TRAUB AND R. P. WEBER, *The algebraic theory of matrix polynomials*, SIAM J. Numer. Anal., 13 (1976), pp. 831–845.

[5] ———, *Algorithms for solvents of matrix polynomials*, SIAM J. Numer. Anal., 15 (1978), pp. 523–533.

[6] G. FORSYTHE AND C. B. MOLER, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[7] B. S. GARBOW, *Algorithm 535—the QZ algorithm to solve the generalized eigenvalue problem for complex matrices*, ACM TOMS, 4 (1978), pp. 404–410.

[8] B. S. GARBOW et al., *Matrix Eigensystems Routines*—EISPACK *Guide Extension*, Springer-Verlag, Berlin, 1977.

[9] G. H. GOLUB, S. NASH AND C. VAN LOAN, *A Hessenberg–Schur method of the problem $AX+XB=C$*, IEEE Trans. Automat. Control, AC24 (1979), pp. 909–913.

[10] L. V. KANTOROVICH AND G. P. AKILOV, *Functional Analysis in Normed Spaces*, Pergamon Press, New York, 1964.
[11] P. LANCASTER, *Lambda-Matrices and Vibrating Systems*, Pergamon Press, Oxford, 1966.
[12] C. B. MOLER AND G. W. STEWART, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241–256.
[13] L. B. RALL, *Computational Solutions of Nonlinear Operator Equations*, John Wiley, New York, 1969.
[14] J. M. VARAH, *Rigorous machine bounds for the eigensystem of a general complex matrix*, Math. Comp., 22 (1968), pp. 793–801.
[15] R. C. WARD, *The combination shift QZ algorithm*, SIAM J. Numer. Anal., 12 (1975), pp. 835–853.
[16] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Oxford, 1965.
[17] ———, *Rounding Errors in Algebraic Processes*, Prentice-Hall, Englewood Cliffs, NJ, 1963.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0006 $01.00/0

AN ANALYSIS OF AN INVERSE PROBLEM IN ORDINARY DIFFERENTIAL EQUATIONS

RICHARD C. ALLEN† AND STEVEN A. PRUESS†

Abstract. A method is proposed for recovering the matrix X in $u' = Xu$ when some of the components of u are known. Both continuous and discrete cases are considered, with the latter of most interest for applications. An algorithm is described and numerical examples are given.

Key words. ordinary differential equation, inverse problem, numerical methods

1. Introduction. In this paper we are concerned with the problem of recovering the coefficient matrix X in the linear system of differential equations

$$\frac{d}{dt}u(t) = Xu(t), \qquad 0 \leqq t \leqq 1, \tag{1.1}$$

where $k \leqq n$ of the $u_i(t)$ are available as tabular functions. A closely related problem is that of fitting exponential functions to data since the solution $u(t)$ to (1.1) clearly has this form. Such exponential fitting is discussed in many places in the literature [2], [6]–[9], [12]–[19] but, as is mentioned in several of these references (e.g., [8, pp. 272–280], [18]), the solution to this problem can be very sensitive to noise. In many applications knowledge of the coefficient matrix X is sought, so we have addressed this problem.

Since the authors are unaware that the problem of determining X has been analyzed before, we begin in § 2 by deriving conditions under which the problem of determining X is well defined. Algorithms are constructed for computing X both when $k = n$ and when $k < n$. These algorithms recover X up to a similarity transformation, uniquely if $k = n$. The more realistic case of discrete data is treated in § 3; the arguments use known results on perturbations of linear systems, but application to the inverse problem (1.1) is new. The final section offers some numerical results which illustrate the utility of the algorithms. Only interpolating spline approximations are used since they permit a straightforward verification of the theory of § 3. When $k < n$, the matrix X can always be recovered uniquely by imposing a suitable number of auxiliary conditions. An example of this is also included in § 4.

A major motivation and application of our work comes from problems in linear compartmental analysis which arise in such diverse areas as biology, medicine and engineering. An excellent discussion of the compartmental problem may be found in [14, especially lectures 3 and 4], where the author surveys a class of biological problems leading to systems of the form $\dot{u} = Xu$ where u is known and X is desired. There it is assumed that the components of u have somehow been represented satisfactorily by explicit exponential sums

$$\sigma_1 e^{\alpha_1 t} + \cdots + \sigma_n e^{\alpha_n t}$$

with distinct α_i. The Jordan form is diagonal under this assumption and X is obtained by a similarity transformation. In contrast to this approach, our method does not require that the eigenvalues of X be distinct, nor does it require that the data be fit explicitly by exponential sums.

* Received by the editors March 31, 1980, and in revised form December 3, 1980.

† Department of Mathematics and Statistics, University of New Mexico, Albuquerque, New Mexico 87131.

2. Mathematical preliminaries. In this section u_i refers to the ith component of the vector u; Greek letters refer to scalars or scalar-valued functions.

From $u' = Xu$, it follows that $u(t) = e^{Xt}u(0)$. For a more useful alternate representation, let $\{\lambda_i\}_{i=1}^{d}$ be the distinct eigenvalues of X with algebraic multiplicities $M_1, M_2, \cdots, M_d$; then u can be written as

$$u(t) = \sum_{j=1}^{n} f_j \phi_j(t) \tag{2.1}$$

for some vectors $\{f_j\}$ with $\phi_1(t) = e^{\lambda_1 t}, \cdots, \phi_{M_1}(t) = t^{M_1-1}e^{\lambda_1 t}$, etc. Since $\phi_1' = \lambda_1\phi_1$, $\phi_2' = \lambda_1\phi_2 + \phi_1$, etc., and $\{\phi_i\}$ is linearly independent over $[0, 1]$, $u' = Xu$ is equivalent to the system

$$XF = FV. \tag{2.2}$$

Here, F is the $n \times n$ matrix with f_j in column j; V has diagonal $(\lambda_1, \cdots, \lambda_d)$, subdiagonal $(\mu_2, \cdots, \mu_n)$ and all other entries zero. Each $\mu_i \in \{0, 1, \cdots, n-1\}$ and $\mu_j = 0$ if and only if $j \in \{M_1, M_1 + M_2, \cdots, n\}$. Clearly, X is determined uniquely if and only if

$$F \text{ is invertible.} \tag{2.3}$$

While much use is made of (2.3) below, the condition suffers from being dependent on a particular representation of u (in terms of $\{\phi_j\}$). A more satisfactory condition can be found by noting that $\{\phi_j\}$ linearly independent implies F is invertible if and only if $\{u_i(t)\}_{i=1}^{n}$ is linearly independent. Since any change of basis from $\{\phi_j\}$ to another representation is via an invertible transformation, it follows that X is determined uniquely if and only if

$$\{u_i(t)\}_{i=1}^{n} \text{ is linearly independent over } [0, 1]. \tag{2.4}$$

Note that if $X = SJS^{-1}$ with J in Jordan form, then $u(t) = Se^{Jt}S^{-1}u(0)$. Set $v = S^{-1}u$ so that $v(t) = e^{Jt}v(0) = \sum g_j\phi_j(t)$ with $g_j = S^{-1}f_j$. It follows that F is invertible if and only if $G := [g_1, \cdots, g_n]$ is invertible. Now, if some λ, say λ_1, appears in more than one Jordan block of J, then $t^{M_1-1}e^{\lambda_1 t}$ cannot appear in $v(t)$ (recall that M_1 is the algebraic multiplicity of λ_1). But then $g_{M_1} = 0$, so G, and hence F, is not invertible. Thus, the problem is not well defined when X is derogatory.

Next consider the case when only k components of u are known, but still for all t. For simplicity, the known components are assumed to be the first k. If not all rows of F are known, it is clear from the characterizing equation $XF = FV$ that X cannot be determined. In this situation, we attempt to determine X up to a certain type of similarity transformation, so that at least the eigenvalues can be found. Since a similarity transformation on X is equivalent to a change of basis for u, it seems reasonable to restrict attention to those transformations which leave the known components of u unchanged. In particular, with I_k the k by k identity, set

$$\mathcal{S} = \left\{ Pn \times n, \text{ invertible}/P = \begin{pmatrix} I_k & 0 \\ P_1 & P_2 \end{pmatrix} \right\}. \tag{2.5}$$

If $\bar{X}$ is the exact (but unknown) answer, we try to find (or estimate) $X = P\bar{X}P^{-1}$ for some $P \in \mathcal{S}$.

With this goal in mind, we note that if the problem were well defined when $k = n$, i.e., the associated $\bar{F}$ is invertible, then the $n - k$ unknown rows of $\bar{F}$ are arbitrary as long as full rank is maintained. To show this, let $[F_{11}F_{12}]$ be the k known rows of $\bar{F}$ with

F_{11} $k \times k$. Choose C_1 $n-k \times k$, C_2 $n-k \times n-k$ so that $F = \binom{F_{11}\ F_{12}}{C_1\ C_2}$ has full rank (any row dependencies in $[F_{11}F_{12}]$ would produce a singular $\bar{F}$, so we must have rank $[F_{11}F_{12}] = k$). If $X = FVF^{-1}$, $\bar{X} = \bar{F}V\bar{F}^{-1}$, then a simple calculation shows that $F = P\bar{F}$ with $P = \binom{I_k\ 0}{(C_1\ C_2)\bar{F}^{-1}}$ and hence $X = P\bar{X}P^{-1}$ for $P \in \mathcal{S}$, as desired.

Besides the full rank condition on $[F_{11}F_{12}]$, the above analysis assumes that V, which depends on the eigenvalues of $\bar{X}$ and their algebraic multiplicities, is known. For this to be so, the functions ϕ_{M_1}, $\phi_{M_1+M_2}$, $\cdots$, ϕ_n must appear at least once in the formulas for $u_1, \cdots, u_k$. For example, if $n=3$ and $k=2$ with $u_1(t) = 3e^{-t} - 2e^{-2t}$, $u_2(t) = te^{-t}$, then the eigenvalues are -1, -1, -2. However, if u_1 alone is given, then only two of the three eigenvalues are apparent. The precise statement for determination of a solution is: $\bar{X}$ is determined to within a similarity transformation in $\mathcal{S}$ if and only if

$$\text{rank } [F_{11}F_{12}] = k \tag{2.6a}$$

and

$$\text{for } 1 \leqq i \leqq k \text{ not all } f_{ij} = 0,\ j = M_1, M_1+M_2, \cdots, n. \tag{2.6b}$$

While (2.6b) looks rather odd, its significance is that it characterizes n. It also implies that the first k equations of the system $u' = Xu$ do not uncouple from the remaining $n-k$.

It appears difficult to restate condition (2.6b) in an equivalent manner which is independent of the representation of u in terms of $\{\phi_j\}$. What is needed is a condition which characterizes n and says that information about all n eigenvalues is present in the known data. There is an analogue of (2.6a), viz., $\{u_i(t)\}_{i=1}^k$ is linearly independent. This equivalence follows immediately from the linear independence of $\{\phi_j\}$.

Since $\{\phi_j\}$ is not easily recoverable from discrete data, we now introduce an alternate characterization of X from which discretizations can be made. From $u' = Xu$ it follows that

$$X \cdot \int_0^t u(s)\, ds = u(t) - u(0).$$

Choose $\{s_i\}_{i=1}^n$ distinct in $(0, 1]$; then

$$XA = B, \tag{2.7}$$

where

$$A = \left[\int_0^{s_1} u(s)\, ds, \cdots, \int_0^{s_n} u(s)\, ds \right], \qquad B = [u(s_1) - u(0), \cdots, u(s_n) - u(0)].$$

When $k < n$, only the first k rows of A and B can be computed. We now show that if the determination of X is well defined, the known k rows of A are independent.

THEOREM 1. *Assume $\{\int_0^t \phi_j(t)\, dt\}_{j=1}^n$ is linearly independent over $\{s_i\}_{i=1}^n$. If $k = n$ and (2.3) holds, then A is invertible. If $k < n$ and (2.6) holds, then the first k rows of A are linearly independent.*

Proof. If some linear combination of the rows of A vanishes, then from (2.1) for $1 \leqq l \leqq n$

$$0 = \sum_{i=1}^k \gamma_i \int_0^{s_l} u_i(t)\, dt = \sum_{j=1}^n \left(\sum_{i=1}^k \gamma_i f_{ij} \right) \psi_j(s_l), \tag{2.8}$$

with $\psi_j(t) := \int_0^t \phi_j(s)ds$. By assumption these functions are linearly independent over $\{s_l\}$, so $\sum_{i=1}^{k} \gamma_i f_{ij} = 0, 1 \leqq j \leqq n$. Finally, (f_{ij}) of full rank implies $\gamma_i = 0, 1 \leqq i \leqq k$, as desired. □

Note that $\{\psi_j\}_{j=1}^{n}$ linearly independent over $\{s_i\}$ follows if X has real eigenvalues (see, e.g., Meinardus [10, § 6.5]). This is the usual situation for the applications mentioned in the introduction. In any case our numerical algorithm discussed in § 3 can monitor this independence so we do not feel this is a critical restriction.

In order to use (2.7) to solve for X when $k < n$, some way is needed to compute the remaining rows of A and B. Equivalently, some means must be found for computing $u_{k+1}, \cdots, u_n$. This section is concluded with a procedure which uses the known data along with the differential equation to fill in the remaining data. The algorithm requires $n-k$ stages: at stage $s (k \leqq s < n)$ assume (1) $u_1, \cdots, u_s$ are known; and (2) condition (2.6) holds with $k = s$. To generate u_{s+1}, first factor $A^T = [A_{11} A_{12}]^T$, A_{11} $s \times s$, A_{12} $s \times n-s$, into $Q[R^T 0]^T$ with R $s \times s$ and upper triangular, Q $n \times n$ and invertible. This can be done by Gauss elimination (with partial pivoting) or a QR factorization (with Q orthogonal). Set $C = Q^{-1}B^T$ and partition C into $\binom{C_1}{C_2}$ with C_1 $s \times s$, C_2 $n-s \times s$. Solve $RX_{11}^T = C_1$ for $X_{11} = (x_{ij})$, $s \times s$. Choose J so that $\|C_2 e_J\|_2 = \max_j \|C_2 e_j\|_2 (\|y\|_2^2 := \sum y_i^2$, e_j is column j of the identity matrix of appropriate size). Finally define $u_{s+1} = u_J' - \sum_{j=1}^{s} x_{Jj} u_j$.

THEOREM 2. *If the underlying problem is well defined, i.e.,* (2.6) *holds, then the above algorithm is well defined.*

Proof. The argument is by induction on s. For $s = k$ the assumptions follow from (2.6). Assuming that the algorithm works up to stage s, we must show the above steps are valid and that the resulting $u_1, \cdots, u_{s+1}$ satisfy (2.6). From (2.7),

$$\begin{pmatrix} A_{11}^T & A_{21}^T \\ A_{12}^T & A_{22}^T \end{pmatrix} \begin{pmatrix} X_{11}^T \\ X_{12}^T \end{pmatrix} = \begin{pmatrix} B_1^T \\ B_2^T \end{pmatrix},$$

with X partitioned as is A. Since X is characterized only up to a change of variables for $u_{k+1}, \cdots, u_n$, the last $n-k$ rows of A are arbitrary as long as full rank is maintained. In particular, at stage s the blocks A_{21}^T and A_{22}^T are arbitrary, and after the factorization,

$$\begin{pmatrix} R & \tilde{A}_{21}^T \\ 0 & \tilde{A}_{22}^T \end{pmatrix} \begin{pmatrix} X_{11}^T \\ X_{12}^T \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}.$$

The arbitrariness of A_{21}, A_{22} is equivalent (through the invertible matrix Q) to arbitrariness of $\tilde{A}_{21}$, $\tilde{A}_{22}$. Set $\tilde{A}_{21} = 0$ so that $X_{11}^T = R^{-1}C_1$. Then rank $[A_{11}A_{12}] = s$ (from the induction hypotheses and Theorem 1) implies R is invertible, so X_{11} is uniquely determined. Choose J as described. Not all the $C_2 e_j = 0$, since otherwise $X_{12} = 0$, which implies the differential equation $u' = Xu$ uncouples into an $s \times s$ system in violation of (2.6b). Thus, we have $C_2 e_J \neq 0$. Let $\tilde{A}_{22}^T$ have first column $C_2 e_J$, with the remaining columns chosen so that $\tilde{A}_{22}$, and thus A, is invertible. Then, $\tilde{A}_{22}^T X_{12}^T = C_2$ implies $X_{12}^T e_J = e_1$; i.e., row J of X_{12} is e_1. From row J of $u' = Xu$ it then follows that $u_{s+1} = u_J' - \sum_{j=1}^{s} x_{Jj} u_j$. To complete the proof, it is necessary to show that the F matrix corresponding to the new data $u_1, \cdots, u_{s+1}$ satisfies (2.6). The first s components have been unchanged by the above process, only the new variable u_{s+1} has been added. Equivalently, only a new row has been added to F; the first s rows remain unchanged from the previous stage. Condition (2.6b) follows trivially. For (2.6a), if rank $[F_{11}F_{12}] < s+1$, then the newly added row must be dependent on the others. Equivalently,

$u_{s+1} = \sum_{i=1}^{s} \gamma_i u_i$ with at least one γ_i nonzero. Since some of the X entries change in proceeding from stage s to stage $s+1$, we use a circumflex to indicate the values at the beginning of stage s. Similarly, $\hat{u}_{s+1}, \cdots, \hat{u}_n$ are the variables (known only implicitly) at the start of stage s. Then $J \leqq s$ implies

$$u'_J = \sum_{j=1}^{n} \hat{x}_{Jj}\hat{u}_j = \sum_{j=1}^{s} x_{Jj}u_j + \sum_{j=s+1}^{n} \hat{x}_{Jj}\hat{u}_j.$$

But at the end of stage s

$$u'_J = u_{s+1} + \sum_{j=1}^{s} x_{Jj}u_j = \sum_{j=1}^{s} (\gamma_j + x_{Jj})u_j,$$

which implies

$$\sum_{j=s+1}^{n} \hat{x}_{Jj}\hat{u}_j = \sum_{j=1}^{s} \gamma_j u_j = \sum_{j=1}^{s} \gamma_j \hat{u}_j, \quad \text{not all } \gamma_j \text{ zero},$$

in contradiction to the assumed linear independence of variables at the start of this stage. □

3. Discrete data. In applications u can only be measured at discrete data points, say $\{t_i\}$, $1 \leqq i \leqq m$. In order to apply the method suggested by (2.7) to estimate X, some closed form approximation $\hat{u}$ to u is needed. Thus, the first step of our algorithm is to construct approximations $\hat{u}_i$ to u_i, $1 \leqq i \leqq k$. The choice for the type of approximation used is irrelevant to the theory of this section as it is only necessary for $\hat{u}$ to be sufficiently close to u.

When $k = n$, i.e., data are known for all components, a straightforward analogue of (2.7) is used to compute an approximation $\hat{X}$ to X.

Algorithm I.

(1) Compute $\hat{u}_i(t)$, an approximation to $u_i(t)$ $(1 \leqq i \leqq n)$, based on the given data $\{u(t_j)\}$.

(2) Set $s_i = i/n$, $1 \leqq i \leqq n$, and form the matrices

$$\hat{A} = \left[\int_0^{s_1} \hat{u}(s)\, ds, \cdots, \int_0^{s_n} \hat{u}(s)\, ds \right], \qquad \hat{B} = [\hat{u}(s_1) - \hat{u}(0), \cdots, \hat{u}(s_n) - \hat{u}(0)].$$

(3) Solve $\hat{X}\hat{A} = \hat{B}$ (by solving $\hat{A}^T\hat{X}^T = \hat{B}^T$ for the n columns of $\hat{X}^T$).

For the following error analysis the vector norm $\|\cdot\|_2$ defined earlier, with associated map norm $\|\cdot\|_2$ for matrices, and Frobenius norm $\|X\|_F := \text{sqrt}\,(\sum\sum x_{ij}^2)$ are used. In addition, we need $\|u_i\|_t := \max_t |u_i(t)|$ and $\|u\|_{2,t} := \text{sqrt}\,(\sum \|u_i\|_t^2)$.

Bounds on the error $X - \hat{X}$ follow from standard perturbation results for linear systems [5], [11]. We assume, henceforth, that the underlying continuous problem is well defined so that the matrix A in (2.7) is invertible. It is first necessary to bound the perturbations in A and B; the proof of the following lemma is just a simple calculation.

LEMMA.

$$\|(A^T - \hat{A}^T)e_i\|_2^2 \leqq n\|u_i - \hat{u}_i\|_t^2, \qquad 1 \leqq i \leqq n,$$

$$\|A - \hat{A}\|_F \leqq \sqrt{n}\,\|u - \hat{u}\|_{2,t},$$

$$\|(B^T - \hat{B}^T)e_i\|_2^2 \leqq 4n\|u_i - \hat{u}_i\|_t^2, \qquad 1 \leqq i \leqq n,$$

$$\|B - \hat{B}\|_F \leqq 2\sqrt{n}\,\|u - \hat{u}\|_{2,t}.$$

THEOREM 3. *If* $\|u-\hat{u}\|_{2,t} < 1/(\sqrt{n}\,\|A^{-1}\|_2)$, *then*
(1) $\hat{A}$ *is invertible*;
(2) *for* $i = 1, 2, \cdots, n$,

$$\frac{\|X^T e_i - \hat{X}^T e_i\|_2}{\|X^T e_i\|_2} \leqq M\left\{\frac{2}{\|B^T e_i\|_2} + \frac{1}{\|A^T e_i\|_2}\right\}\|u-\hat{u}\|_{2,t}$$

where

$$M = \frac{\sqrt{n}\|A\|_2\|A^{-1}\|_2}{1-\sqrt{n}\|A^{-1}\|_2 \cdot \|u-\hat{u}\|_{2,t}}.$$

The bound in this theorem assumes $B^T e_i \neq 0$ or equivalently $X^T e_i \neq 0$. If $X^T e_i = 0$, then an analogous bound can be established on the absolute error $\|\hat{X}^T e_i\|_2$.

For $k < n$, new approximate data $\hat{u}_{k+1}, \cdots, \hat{u}_n$ are generated by a procedure analogous to that described at the close of § 2, with $\hat{u}$, $\hat{A}$, $\hat{B}$ replacing u, A, B respectively. The first two steps of the algorithm proceed as in the $k = n$ case, except that only the first k rows of $\hat{A}$, $\hat{B}$ and the first k components of $\hat{u}$ are computed. Then $n-k$ stages of Algorithm II are performed, viz., at stage s $(k \leqq s < n)$:

Algorithm II.
(1) Factor $\hat{A}$ into $\hat{Q}[\hat{R}^T 0]^T$ with $\hat{R}$ s by s, upper triangular.
(2) Partition $\hat{C} := \hat{Q}^{-1}\hat{B}^T$ into $\binom{C_1}{C_2}$ with $\hat{C}_1$ $s \times s$.
(3) Solve $\hat{R}\hat{X}_{11} = \hat{C}_1$ for $\hat{X}_{11} = (\hat{x}_{ij})$ $s \times s$.
(4) Choose J so that $\|\hat{C}_2 e_J\|_2 = \max_j \|\hat{C}_2 e_j\|_2$ and define

$$\hat{u}_{s+1} = \hat{u}'_J - \sum_{j=1}^{s} \hat{x}_{Jj}\hat{u}_j.$$

Once data are known for all components, Algorithm I can be used to construct $\hat{X}$. For most classes of approximating functions $\hat{u}$, accuracy can be lost at each stage of Algorithm II due to the $\hat{u}'_j$ term in the new data $\hat{u}_{s+1}$.

If $\|u-\hat{u}\|_{2,t}$ is sufficiently small, then standard perturbation results from linear algebra can be used to demonstrate that Algorithm II is well defined by following the argument in Theorem 2. This assumes the new data generated are also sufficiently accurate, which depends on the type of approximation being used. When $k = n$, Theorem 3 holds for any kind of approximation $\hat{u}$ to u.

4. Numerical examples. To use the algorithms of the preceding section, some choice must be made for the form of $\hat{u}$. In order to demonstrate rates of convergence which verify the theory, interpolating spline approximations have been used. In practice the data will not be sufficiently accurate for this, so some kind of smoothing is advised. Polynomial splines for each $\hat{u}_i$ are easy to calculate and manipulate, so we have used these. For algorithms and analysis of interpolating polynomial splines see de Boor [3, Chapts. 3–6, 13]. This reference also discusses smoothing and least squares splines which may be used when the data is noisy.

We begin this section with specific error bounds for general k when cubic interpolating splines are used. The bound (4.3) is well known; however, its application (Theorem 4) is new. If end conditions like those incorporated into the code in [4] are used, then it is known [16] that for u sufficiently smooth (here $u(t) = e^{Xt}u(0)$ is smooth) there is a constant K, independent of the data spacing, such that for each component i

$$\|D^p(u_i - \hat{u}_i)\|_t \leqq Kh^{4-p}, \qquad 0 \leqq p \leqq 4. \tag{4.1}$$

Here $h := \max(t_{j+1} - t_j)$, D^p is the pth derivative operator and it is assumed the mesh is quasi-uniform; i.e., $\max(t_{j+1} - t_j)/\min(t_{j+1} - t_j)$ can be bounded by one constant for all $\{t_j\}$ considered. It then follows that as $h \to 0$

$$\|u_i - \hat{u}_i\|_t = O(h^4), \qquad 1 \leqq i \leqq k. \tag{4.2}$$

To construct data for the remaining $n-k$ components, use Algorithm II. Since at each stage the new component $\hat{u}_{s+1}$ depends on $\hat{u}'_j$, it follows from (4.1) that one power of h in the error bound is lost for each stage, i.e., as $h \to 0$,

$$\|u_i - \hat{u}_i\|_t = O(h^{4+k-i}), \qquad k < i \leqq n. \tag{4.3}$$

If $n - k \geqq 4$, $\hat{u}$ will not converge to u when cubic splines are used. Fortunately, in most applications n is small.

The bound on $X - \hat{X}$ when $k = n$ follows directly from (4.2) and Theorem 3; when $k < n$, observe that at stage s of Algorithm II the $s \times s$ principal minor of $\hat{X}$ is unchanged. At the initial stage $s = k$, the perturbations in the matrices used to characterize this minor are $O(h^4)$ from (4.2). At the next stage the perturbations can be $O(h^3)$ since $\hat{u}_{k+1}$ has error $O(h^3)$. Proceeding inductively one can establish

THEOREM 4. *If $k = n$ and $\hat{X}$ is the output of Algorithm* I, *then as $h \to 0$*

$$\|X - \hat{X}\|_F = O(h^4). \tag{4.4}$$

If $k < n$ and $\hat{X}$ is the output of Algorithm II *and* I, *then there is a matrix $X = P\bar{X}P^{-1}$, $P \in \mathcal{S}$, $\bar{X}$ the exact (but unknown) answer, such that as $h \to 0$*

$$|x_{ij} - \hat{x}_{ij}| = O(h^4), \qquad 1 \leqq i, j \leqq k.$$

For $s = k+1, \cdots, n$,

$$|x_{ij} - \hat{x}_{ij}| = O(h^{4+k-s}), \qquad \begin{cases} i = s, & j \leqq s, \\ j = s, & i \leqq s. \end{cases}$$

One modification of Algorithm II produces more accurate approximations (asymptotically), at least for $k > 1$. Rather than produce new data in merely one component at each stage, generate data in l components, where l is the maximum number of linearly independent columns of $\hat{C}_2$. Then set

$$u_{s+i} = u'_{I(i)} - \sum_{j=1}^{s} x_{I(i),j} u_j, \qquad 1 \leqq i \leqq l,$$

where $I(i)$ is the index of the ith independent column of $\hat{C}_2$. This approach has the advantage of higher accuracy (since fewer derivatives are needed to compute $u_{k+1}, \cdots, u_n$) but has the obvious disadvantage that it requires a means of determining linear independence. Since it is possible to have examples where $l = 1$ at each stage, we have used Algorithm II as given above.

For a completely different approach for generating new data, Allen-Wright [1] have proposed a variant of Prony's method. For equally spaced data with spacing h, $u(t_{i+1}) = e^{Xh} u(t_i)$ implies

$$u(t_{i+n-1})C_1 + \cdots + u(t_i)C_n = -u(t_{i+n}), \tag{4.5}$$

where $t^n + C_1 t^{n-1} + \cdots + C_n$ is the characteristic polynomial for e^{Xh}. If sufficient data exist, viz., $(m-n)k \geqq n$, an overdetermined system results for $\{C_i\}$. Choose $\{u_i(t_j)\}$,

$k < i \leqq n$, $1 \leqq j \leqq n$ so that $(u_i(t_j))$, $1 \leqq i \leqq n$, $1 \leqq j \leqq n$ has full rank. Then (4.5) is used to compute $u_i(t_j)$ when $i > k$ and $j > n$. For discrete data one uses $\hat{u}$ rather than u. While this idea can be extended to the case of non-equally spaced data, it suffers from the fact that the coefficient matrix associated with (4.5) can be ill-conditioned. It appears that the condition number is $O(1/h^{n-k})$ as $h \to 0$. However, if the data are quite accurate and equally spaced, then this method produces good results.

We note that the condition that F be of full rank and $\{\int_0^t \phi_j(t)\,dt\}$ be linearly independent over $\{s_i\}$ can be monitored by our algorithm. The violation of either assumption will produce a rank deficient A. Since the linear system solver in [4] provides condition number estimates, dependence can be tested. In fact, the condition number estimate provides a measure of the sensitivity of the answers to uncertainties in the data, which is useful to know.

This section is concluded with two numerical examples to illustrate our algorithms for recovering X. All calculations were done on an IBM 3032 in REAL*8 arithmetic.

Let X_k denote the $k \times k$ principal minor of X and $\delta X_k = X_k - \hat{X}_k$. As a measure of the size of δX_k, we use $\|\delta X_k\|_F$. As our first example we consider

$$u = \begin{pmatrix} 3 - e^{-t} - e^{t} \\ 3e^{-t} + e^{t} \\ 1 - e^{t} \end{pmatrix}$$

for $0 \leqq t \leqq 1$, which has the solution

$$X = \begin{pmatrix} -0.6 & 0.2 & 1.8 \\ 0.8 & -0.6 & -2.4 \\ -0.4 & -0.2 & 1.2 \end{pmatrix}.$$

In Table 1, we tabulate the error $\|\delta X_k\|_F$ for various values m. In this case $h = 1/m$. The "exact" answer X used for $k < n$ is the computed $\hat{X}$ corresponding to $m = 128$. The largest condition number was $O(10^3)$.

TABLE 1

m	$k=3$	$k=2$		$k=1$		
	$\|\delta X_3\|_F$	$\|\delta X_2\|_F$	$\|\delta X_3\|_F$	$\|\delta X_1\|_F$	$\|\delta X_2\|_F$	$\|\delta X_3\|_F$
4	$.13\times10^{-1}$	$.13\times10^{-2}$	$.42\times10^{0}$	$.48\times10^{-3}$	$.49\times10^{-1}$	$.97\times10^{0}$
8	$.10\times10^{-3}$	$.88\times10^{-4}$	$.11\times10^{1}$	$.29\times10^{-5}$	$.45\times10^{-2}$	$.81\times10^{-1}$
16	$.15\times10^{-5}$	$.77\times10^{-5}$	$.15\times10^{-2}$	$.17\times10^{-6}$	$.47\times10^{-3}$	$.17\times10^{-1}$
32	$.59\times10^{-7}$	$.56\times10^{-6}$	$.17\times10^{-3}$	$.22\times10^{-7}$	$.54\times10^{-4}$	$.40\times10^{-2}$

As our second example, we use

$$u = \begin{pmatrix} e^{-t} + e^{-2t} \\ e^{-t} - e^{-2t} \end{pmatrix},$$

which has the solution

$$\bar{X} = \begin{pmatrix} -1.5 & 0.5 \\ 0.5 & -1.5 \end{pmatrix}.$$

The results are given in Table 2.

TABLE 2

m	$k=2$	$k=1$	
	$\|\delta X_2\|_F$	$\|\delta X_1\|_F$	$\|\delta X_2\|_F$
4	$.49\times10^{-1}$	$.52\times10^{-3}$	$.22\times10^{0}$
8	$.17\times10^{-3}$	$.53\times10^{-4}$	$.48\times10^{-1}$
16	$.47\times10^{-5}$	$.43\times10^{-5}$	$.71\times10^{-2}$
32	$.27\times10^{-6}$	$.29\times10^{-6}$	$.95\times10^{-3}$

The original matrix of coefficients $\bar{X}$ can be recovered from our computed

$$\hat{X}=\begin{pmatrix}-1.4182431 & 1.0000000\\ 0.24331553 & -1.5817550\end{pmatrix},\qquad k=1,$$

by imposing some auxiliary conditions. Recalling that $X=P\times P^{-1}$ where P is of the form

$$P=\begin{pmatrix}1 & 0\\ P_1 & P_2\end{pmatrix},$$

we see that

$$\bar{x}_{11}=x_{11}-\frac{p_1x_{12}}{p_2},$$

$$\bar{x}_{12}=\frac{x_{12}}{p_2},$$

$$\bar{x}_{21}=p_1x_{11}+p_2x_{21}-\frac{p_1^2x_{12}}{p_2}-p_1x_{22},$$

$$\bar{x}_{22}=\frac{p_1x_{12}}{p_2}+x_{22}.$$

If we require that $\bar{x}_{12}=\bar{x}_{21}$, $u_2(0)=0$ and $\bar{x}_{12}>0$ uniquely determines

$$p_1=0.1635,\qquad p_2=2.0000,$$

and hence, correct to four digits, we have the following approximation to $\bar{X}$:

$$\begin{pmatrix}-1.4997 & 0.4996\\ 0.4995 & -1.4993\end{pmatrix}.$$

REFERENCES

[1] R. ALLEN AND R. WRIGHT, *A spline method for recovering the matrix A in $q'=Aq$*, TR 310, Department of Mathematics, University of New Mexico, Albuquerque, 1975.

[2] R. CORNELL, *A method for fitting linear combinations of exponentials*, Biometrics, 18 (1962), pp. 104–113.

[3] C. DE BOOR, *A Practical Guide to Splines*, Springer, New York, 1978.

[4] G. FORSYTHE, M. MALCOLM AND C. MOLER, *Computer Methods for Mathematical Computations*, Prentice Hall, Englewood Cliffs, NJ, 1977.

[5] G. FORSYTHE AND C. MOLER, *Computer Solution of Linear Algebraic Systems*, Prentice Hall, Englewood Cliffs, NJ, 1967.
[6] H. GLASS AND A. DE GANETA, *Quantitative analysis of exponential curve fitting for biological applications*, Phys. Med. Biol., 12 (1967), pp. 379–388.
[7] D. KAMMLER AND R. MCGINN, *A bibliography for approximation with exponential sums*, J. Comput. Appl. Math., 4 (1978), pp. 167–173.
[8] C. LANCZOS, *Applied Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1964.
[9] P. MANCINI AND A. PILO, *A computer program for multiexponential fitting by the peeling method*, Comp. and Biomed. Res., 3 (1970), pp. 1–14.
[10] G. MEINARDUS, *Approximation of Functions*, Springer, New York, 1967.
[11] B. NOBLE AND J. DANIEL, *Applied Linear Algebra*, Prentice Hall, Englewood Cliffs, NJ, 1977.
[12] M. OSBORNE, *Some special nonlinear least squares problems*, SIAM J. Numer. Anal., 12 (1975), pp. 571–592.
[13] W. PERL, *A method for curve fitting by exponential functions*, Int. J. Appl. Radiation and Isotopes, 8 (1960), pp. 211–222.
[14] S. RUBINOW, *Mathematical Problems in the Biological Sciences*, CBMS Regional Conference Series in Applied Mathematics, 10, Society for Industrial and Applied Mathematics, Philadelphia, 1973.
[15] H. SALZER, *Some extensions of Prony approximation*, Z. Angew. Math. Mech., 57 (1977), pp. 269–271.
[16] M. SCHULTZ, *Spline Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1973.
[17] W. SIMON, *A method of exponential separation applicable to small computers*, Phys. Med. Biol., 15 (1970), pp. 355–360.
[18] M. SIMPSON-MORGAN, *A note on fitting multiexponential functions of time to experimental data*, Math. Biosci., 5 (1969), pp. 195–199.
[19] W. WISCOMBE AND J. EVANS, *Exponential-sum fitting of radiative transmission functions*, J. Comp. Phys., 24 (1977), pp. 416–444.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0007 $01.00/0

COMPUTING OPTIMAL LOCALLY CONSTRAINED STEPS*

DAVID M. GAY†

Abstract. In seeking to solve an unconstrained minimization problem, one often computes steps based on a quadratic approximation q to the objective function. A reasonable way to choose such steps is by minimizing q constrained to a neighborhood of the current iterate. This paper considers ellipsoidal neighborhoods and presents a new way to handle certain computational details when the Hessian of q is indefinite, paying particular attention to a special case which may then arise. The proposed step computing algorithm provides an attractive way to deal with negative curvature. Implementations of this algorithm have proved very satisfactory in the nonlinear least-squares solver NL2SOL.

Key words. unconstrained optimization, negative curvature

1. Introduction. Many unconstrained minimization algorithms employ a sequence of quadratic approximations $\varphi_i : \mathbb{R}^n \to \mathbb{R}$, $i = 0, 1, 2, \cdots$ to the objective function $\varphi^* : \mathbb{R}^n \to \mathbb{R}$. Using them, they determine a sequence $x_0, x_1, x_2, \cdots$ of points which (usually) are ever better approximations to a local minimizer x^* of φ^*. Given the current iterate $x = x_i$ and quadratic model $\varphi = \varphi_i$, these algorithms usually compute the next iterate $x_+ = x_{i+1}$ in one of two ways. Either they determine a Newton step s such that $\varphi(x+s)$ is minimized, then set $x_+ = x + \lambda s$, where $\lambda > 0$ is chosen so that $\varphi^*(x+\lambda s) < \varphi^*(x)$, or they choose a neighborhood $N = N_i$ of 0 and a point $s^* \in N$ which minimizes $\varphi(x+s)$ over $s \in N$, and they set $x_+ = x + s^*$, having taken care in choosing N that $\varphi^*(x+s^*) < \varphi^*(x)$. Algorithms of the latter sort have an intuitive appeal: φ often approximates φ^* well only in a neighborhood of x, and these algorithms attempt to achieve the maximum function reduction possible on an educated guess at such a neighborhood. This paper concerns itself with choosing s^* in this sort of algorithm, given x and an N of the reasonable form described below. Since we can expect the quadratic approximation φ to φ^* only to be accurate in a neighborhood $x+N$ of the current iterate x, we shall refer to a point s^* that minimizes $\varphi(x+s)$ subject to $s \in N$ as an optimal locally constrained (OLC) step.

Suppose now that we have $g \in \mathbb{R}^n$ and a symmetric $n \times n$ matrix $H \in \mathbb{R}^{n \times n}$ such that $\varphi(x+s) = \varphi^*(x) + g^T s + \frac{1}{2} s^T H s$. (We regard vectors as column vectors and use superscript T for "transpose". Superscript $-T$ means "inverse transpose".) Of course, we have in mind that $g \doteq \nabla \varphi^*(x)$ and $H \doteq \nabla^2 \varphi^*(x)$ in a suitable sense. We lose no generality in assuming $x = 0$ and $\varphi^*(x) = 0$, so that

$$\varphi(s) = g^T s + \tfrac{1}{2} s^T H s. \tag{1.1}$$

For the rest of this paper we assume that N has the form

$$N = \{y \in \mathbb{R}^n : \|Dy\| \leqq \delta\}, \tag{1.2}$$

where $D \in \mathbb{R}^{n \times n}$ is nonsingular, $\|\cdot\|$ denotes the Euclidean norm $\|\cdot\|_2$ (i.e., $\|y\| = (y^T y)^{1/2}$), and $\delta > 0$. Some of the algebra below is simpler if D is the identity matrix I. It

* Received by the editors July 24, 1980. This paper was presented at the Tenth International Symposium on Mathematical Programming, Montreal, Canada, 27–31 August 1979.

† Center for Computational Research in Economics and Management Science, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139. This work was sponsored by the United States Army under contract DAAG-75-C-0024. This material is based upon work supported by the National Science Foundation under grants MCS78-09525, MCS76-00324, and MCS79-06671.

is possible, in effect, to arrange this by a change of variables. Let

$$\tilde{s} = Ds, \tag{1.3a}$$

$$\tilde{g} = D^{-T}g, \tag{1.3b}$$

$$\tilde{H} = D^{-T}HD^{-1}, \tag{1.3c}$$

$$\tilde{N} = \{y \in \mathbb{R}^n : \|y\| \leqq \delta\}, \tag{1.3d}$$

and

$$\tilde{\varphi}(\tilde{s}) = \tilde{g}^T\tilde{s} + \tfrac{1}{2}\tilde{s}^T\tilde{H}\tilde{s}. \tag{1.3e}$$

Then $\varphi(s) = \tilde{\varphi}(\tilde{s})$ and $s \in N$ if and only if $\tilde{s} \in \tilde{N}$, so in proofs we may just as well deal with $\tilde{\varphi}$ and $\tilde{N}$, for which $\tilde{D} = I$, as with φ and N. Therefore we may assume without loss of generality in the proofs below that $D = I$. It is often useful in practice to choose $D \neq I$, e.g., to reflect scale in the components of s, so we leave D in the statements of some results.

Some of the proofs below are simpler if H is a diagonal matrix. We may assume this without losing generality, because we can also arrange for it, in effect, by changing variables. Specifically, since $\tilde{H}$ is symmetric, there exists an orthogonal matrix $U \in \mathbb{R}^{n\times n}$ such that $\hat{H} = U\tilde{H}U^T$ is diagonal with nonincreasing diagonal elements. Thus if $\hat{\tilde{g}} = U\tilde{g}$, $\hat{\tilde{\varphi}}(\hat{\tilde{s}}) = \hat{\tilde{g}}^T\hat{\tilde{s}} + \frac{1}{2}\hat{\tilde{s}}^T\hat{H}\hat{\tilde{s}}$, and $\hat{\tilde{s}} = U\tilde{s}$, then $\hat{\tilde{\varphi}}(\hat{\tilde{s}}) = \tilde{\varphi}(\tilde{s})$ and $\hat{\tilde{s}} \in \tilde{N}$ if and only if $\tilde{s} \in \tilde{N}$.

In many applications H is positive definite. In others, however, H may have one or more nonpositive eigenvalues. This may happen, for instance, when H comes from the "augmented model" in the NL2SOL algorithm [DenGW80]. We therefore consider the general case where H may be indefinite. The scheme we are discussing thus provides an appealing way to deal with negative curvature.

In the next section we show that an OLC step s^* satisfies $(H + \alpha D^TD)s = -g$ for some $\alpha \geqq 0$ such that $H + \alpha D^TD$ is positive semidefinite. Our treatment differs from that of Goldfield, Quandt and Trotter [GolQT66] in that we consider the special case where $H + \alpha D^TD$ is (nearly) singular. This case requires special handling, which we discuss in § 3. In § 4 we give a complete algorithm for computing s^*, and we discuss numerical experience with it in § 5.

The present work builds on that of many others. Among the first papers to consider computing an OLC step was that of Marquardt [Mar63], in which φ^* was a nonlinear sum of squares. (The paper of Levenberg [Lev44] that is often mentioned in the same breath actually considers a somewhat different step, one that minimizes $\varphi(x+s) + \|Ds\|^2$.) Goldfeld, Quandt and Trotter considered a general φ^* but restricted themselves to the case where $H + \alpha D^TD$ is positive definite. Hebden [Heb73] gave an interesting algorithm that computes a good approximation to s^* when $H + \alpha D^TD$ is sufficiently positive definite and that otherwise computes what often would be a reasonable step. Moré [Mor78] refined Hebden's scheme and specialized it to a good algorithm for the case where φ^* is a sum of squares. The algorithm we give in § 4 incorporates Moré's refinements of Hebden's scheme along with a new, often more reasonable way to handle the case of a (nearly) singular $H + \alpha D^TD$. When specialized to least-squares problems, the new algorithm computes the same step as does Moré's, but may expend less work in the special case.

2. Characterizing s^*. The following characterization of an OLC step s^* lies at the heart of the algorithm in § 4 for approximating s^*.

THEOREM 2.1. *If φ and N are given by* (1.1) *and* (1.2), *i.e.*, $\varphi(s) := g^Ts + \frac{1}{2}s^THs$ *and* $N := \{y \in \mathbb{R}^n : \|Dy\| \leqq \delta\}$ *with* $\delta > 0$, *then* $s^* \in N$ *minimizes* $\varphi(s)$ *over* $s \in N$ *if and only if*

there exists $\alpha^* \geqq 0$ *such that* $H + \alpha^* D^T D$ *is positive semidefinite,*

$$(H + \alpha^* D^T D)s^* = -g, \tag{2.1a}$$

and

$$\|Ds^*\| = \delta \quad \text{if } \alpha^* > 0. \tag{2.1b}$$

This α^* *is unique.*

Proof. Without loss of generality, $D = I$, $H = \operatorname{diag}(h_1, \cdots, h_n)$ with $h_1 \geqq h_2 \geqq \cdots \geqq h_n$, and (2.1) has the form

$$(H + \alpha^* I)s^* = -g \tag{2.2a}$$

and

$$\|s^*\| = \delta \text{ if } \alpha^* > 0. \tag{2.2b}$$

(*Only if*). Suppose s^* minimizes φ over N. By the second-order necessary conditions in Theorem 4 of [McC76], there exists $\alpha^* \geqq 0$ such that (2.2a) holds and either $\alpha^* = 0$, $\|s^*\| \leqq \delta$, and H is positive semidefinite, or else $\alpha^* > 0$, $\|s^*\| = \delta$, and

$$y^T(H + \alpha^* I)y \geqq 0 \quad \text{for all } y \in \mathbb{R}^n \text{ with } y^T s^* = 0. \tag{2.3}$$

The "only if" assertion thus holds if $\alpha^* = 0$, so assume $\alpha^* > 0$. In this case (2.3) implies that $H + \alpha^* I$ has at most one negative eigenvalue, so $h_i \geqq -\alpha^*$ for $i < n$ and we must show that $h_n \geqq -\alpha^*$.

If $g_n = 0$, then (2.2a) implies that $h_n + \alpha^* = 0$ or $s_n^* = 0$. If $s_n^* = 0$, then (2.3) implies $h_n + \alpha^* \geqq 0$. The "only if" assertion thus holds whenever $g_n = 0$.

Suppose $g_n \neq 0$. There clearly exist $\alpha^\# \geqq \alpha^* > 0$ and $s^\# \in \mathbb{R}^n$ such that $h_n + \alpha^\# \geqq 0$, $(H + \alpha^\# I)s^\# = -g$, and $\|s^\#\| = \|s^*\| = \delta$. It suffices to show that $\alpha^\# = \alpha^*$. Now (2.2a) implies $(s^*)^T H s^* = -g^T s^* - \alpha^* \delta^2$, so $\varphi(s^*) = (g^T s^* - \alpha^* \delta^2)/2$. Similarly, $\varphi(s^\#) = (g^T s^\# - \alpha^\# \delta^2)/2$. Since $(H + \alpha^* I)s^* = (H + \alpha^\# I)s^\# = -g$, we have $g^T s^* = -(s^\#)^T(H + \alpha^\# I)s^*$ and $g^T s^\# = -(s^\#)^T(H + \alpha^* I)s^*$, so

$$\varphi(s^\#) - \varphi(s^*) = (\alpha^* - \alpha^\#)[\delta^2 - (s^\#)^T s^*]/2. \tag{2.4}$$

If $\alpha^* < \alpha^\#$, then $(h_n + \alpha^*)s_n^\# = (h_n + \alpha^\#)s_n^\# = -g_n \neq 0$ implies $s^\# \neq s^*$, and the Cauchy–Schwarz inequality then implies $\delta^2 - (s^\#)^T s^* > 0$, so $\varphi(s^\#) < \varphi(s^*)$. This contradicts the choice of s^*, so we must have $\alpha^* = \alpha^\#$, and the proof of the "only if" assertion is complete. (The key equation (2.4) also follows from Theorem 7.1 of [Gan78].)

(*If, uniqueness*). Suppose $s^* \in N$ and $\alpha^* \geqq 0$ satisfy (2.2) and that $H + \alpha^* I$ is positive semidefinite. Since N is compact, there exists an $s^\# \in N$ that minimizes φ, and by the "only if" part proved above there exists $\alpha^\# \geqq 0$ such that (2.2) holds with α^* replaced by $\alpha^\#$ and $s^\#$. We must show that $\alpha^* = \alpha^\#$ and $\varphi(s^*) = \varphi(s^\#)$.

First consider the special case $g = 0$. If $h_n \geqq 0$, then (2.2) implies $\alpha^* = \alpha^\# = 0$ and $s_i^* = s_i^\# = 0$ unless $h_i = 0$, whence $\varphi(s^*) = \varphi(s^\#) = 0$. If $h_n < 0$, then the positive semidefiniteness of $H + \alpha^* I$ implies $\alpha^* \geqq -h_n > 0$, and (2.2) implies $\alpha^* = -h_n$. The same holds for $\alpha^\#$, so (2.2a) implies $s_i^* = s_i^\# = 0$ for $h_i \neq h_n$, and (2.2b) implies $\sum_{h_i = h_n}(s_i^*)^2 = \sum_{h_i = h_n}(s_i^\#)^2 = \delta^2$, whence $\varphi(s^*) = \varphi(s^\#) = h_n \delta^2/2$. Thus the "if" assertion holds and α^* is unique when $g = 0$.

Now suppose $g \neq 0$ and let $\eta(\alpha) := \|(H + \alpha I)^{-1} g\|$ for $\alpha > -h_n$. Extend the domain of η to $-h_n$ by defining $\eta(-h_n) := \lim_{\alpha \to -h_n^+} \eta(\alpha)$. If $g_i \neq 0$ for at least one i with $h_i = h_n$, then $\eta(-h_n) = +\infty$; otherwise $\eta(-h_n)$ is finite. In either case $\eta(\alpha)$ is a strictly decreasing function of α. If $h_n \geqq 0$ and $\eta(0) \leqq \delta$, then (2.2) cannot hold with $\alpha^* > 0$, so $\alpha^* = \alpha^\# = 0$. In this case (2.2a) implies that $g_i = 0$ whenever $h_i = h_n = 0$ and that $s_i^* = s_i^\# = -g_i/h_i$

otherwise, so $\varphi(s^*) = \varphi(s^{\#})$. If $h_n < 0$ or $\eta(0) > \delta$, then α^* and $\alpha^{\#}$ must be positive, and (2.2b) and the strict monotonicity of η imply that $\alpha^* = \alpha^{\#}$ is uniquely determined. From (2.2a) we deduce as above that $\varphi(s^*) = \varphi(s^{\#})$. In all cases we thus find that s^* minimizes φ over N and α^* is unique. □

The next section discusses approximating s^* when $H + \alpha^* D^T D$ is singular or nearly singular, so it is interesting to consider the case of exact singularity. From (2.1) it follows that $H + \alpha^* D^T D$ can only be singular if $-\alpha^*$ is the least eigenvalue of $D^{-T} H D^{-1}$ and the projection of $D^{-1} g$ onto the eigenspace of $D^{-T} H D^{-1}$ corresponding to $-\alpha^*$ is zero. In the notation of the above proof, $H + \alpha^* I$ is positive definite whenever $g_i \neq 0$ for at least one i with $h_i = h_n$. When $H + \alpha^* D^T D$ is singular, s^* has the form $s^* = \lim_{\alpha \to \alpha^*} (H + \alpha D^T D)^{-1} g + D^{-1} v$, where v is an eigenvector of $D^{-T} H D^{-1}$ corresponding to the eigenvalue $-\alpha^*$.

3. (Near) Singularity in $H + \alpha^* D^T D$. When $g \neq 0$ (as we henceforth assume), it is usually possible to compute an approximate OLC step s as

$$s = s(\alpha) = -(H + \alpha D^T D)^{-1} g, \tag{3.1}$$

where $\alpha \geqq 0$ is chosen so that $H + \alpha D^T D$ is positive definite and $\|Ds(\alpha)\|$ is near δ. Such a step s exactly minimizes φ (given by (1.1)) on an approximation to the "trust region" N given by (1.2). If $H + \alpha^* D^T D$ is singular in (2.1), however, then it may be impossible to compute a suitable s in this way, because it can happen that $\|Ds(\alpha)\|$ is considerably less than δ for all α that make $H + \alpha D^T D$ positive definite. And if $H + \alpha^* D^T D$ is nearly singular, then computing a sufficiently large $s(\alpha)$ may be impractical or at least unduly costly. Of course, we could simply accept a "short" $s(\alpha)$. But if H has a significantly negative eigenvalue and is a good approximation to the current true Hessian $\nabla^2 \varphi^*(x)$, then it may prove well worthwhile to compute a step $\hat{s}$ such that $D\hat{s}$ has a significant component in the direction of an eigenvector of $D^{-T} H D^{-1}$ corresponding to the smallest eigenvalue. In what follows we describe a simple way to compute such a step $\hat{s}$. This step approximately minimizes φ (to within a prescribed tolerance) on the exact trust region N of (1.2).

We assume for the rest of this section that $D^{-T} H D^{-1}$ has eigenvalues $h_1, h_2, \cdots, h_n$ with $h_1 \geqq h_2 \geqq \cdots \geqq h_n$. We also assume that H is not positive definite and hence that $h_n \leqq 0$.

When $h_n \leqq 0$, the algorithm of § 4 maintains a lower bound η on acceptable values of α such that $\eta \leqq -h_n$. If $\alpha > -h_n$ yields $\|Ds(\alpha)\| < \delta$, then

$$\eta \leqq -h_n \leqq \alpha^* \leqq \alpha. \tag{3.2a}$$

It will be convenient to let

$$\theta = \alpha - \eta. \tag{3.2b}$$

We may regard $H + \alpha^* D^T D$ as nearly singular if we encounter an $\alpha > -h_n$ for which θ is sufficiently small (in the sense made precise in Theorem 3.2 below) and for which $\|Ds(\alpha)\|$ is unacceptably small, say

$$\|Ds(\alpha)\| < \beta\delta, \tag{3.3}$$

for some prescribed $\beta \in (0, 1)$, e.g. $\beta = 0.75$ or $\beta = 0.9$.

Suppose now that (3.3) holds, and let $\tilde{s} = Ds(\alpha)$, $\tilde{s}^* = Ds^*$, $\tilde{g} = D^{-T} g$, and $\tilde{H} = D^{-T} H D^{-1}$. Further, let U be an orthogonal matrix as in § 1 such that $U\tilde{H}U^T = \operatorname{diag}(h_1, \cdots, h_n)$, and let $\tilde{\tilde{s}} = U\tilde{s}$, $\tilde{\tilde{g}} = U\tilde{g}$, and $\tilde{\tilde{s}}^* = U\tilde{s}^*$. For $h_i > h_n$, we may expect that $\tilde{\tilde{s}}_i \doteq \tilde{\tilde{s}}_i^*$, and for $h_i \doteq h_n$, we may expect $\tilde{\tilde{g}}_i \doteq 0$. Thus it seems reasonable to consider

computing an approximate OLC step $\hat{s}$ by finding an approximate eigenvector v of $\tilde{H}$ corresponding to h_n and adding a suitable multiple of $D^{-1}v$ to s. Because of (3.2), $\tilde{H}+\alpha I$ has an eigenvalue $h_n+\alpha$ with $0 \leqq h_n+\alpha \leqq \alpha-\eta =: \theta$ (and eigenvectors corresponding to this eigenvalue are also eigenvectors of $\tilde{H}$ corresponding to h_n). We probably have a factorization of $D^{-T}HD^{-1}+\alpha I$, so for an arbitrary $\kappa > 1$, e.g. $\kappa = 2$, it should be easy to compute $v \in \mathbb{R}^n$ by the inverse power method such that

$$\|v\| = 1, \tag{3.4a}$$

$$\|(D^{-T}HD^{-1}+\alpha I)v\| \leqq \kappa\theta. \tag{3.4b}$$

Given such a v, we may compute

$$\hat{s} = s + \sigma D^{-1}v, \tag{3.5a}$$

with σ chosen so that $\|D\hat{s}\| = \delta$ and $\varphi(\hat{s})$ is minimized. Specifically:

LEMMA 3.1. *If* $\psi(\tau) = \varphi(s+\tau D^{-1}v)$, *and if*

$$\sigma = (\delta^2 - \|Ds\|^2)/[v^TDs + \text{sign}\,(v^TDs)\{(v^TDs)^2 + \delta^2 - \|Ds\|^2\}^{1/2}], \tag{3.5b}$$

(*with* $\text{sign}\,(v^TDs) = 1$ *or* -1 *if* $v^TDs = 0$), *then* $\tau = \sigma$ *minimizes* $\psi(\tau)$ *subject to the constraint* $\|Ds + \tau v\| = \delta$.

Proof. There are two values of τ for which $\|\tilde{s}+\tau v\| = \delta$, namely

$$\tau_+ = -\tilde{s}^Tv + \sqrt{(\tilde{s}^Tv)^2 + \delta^2 - \|\tilde{s}\|^2}, \tag{3.6a}$$

$$\tau_- = -\tilde{s}^Tv - \sqrt{(\tilde{s}^Tv)^2 + \delta^2 - \|\tilde{s}\|^2}. \tag{3.6b}$$

Because of (1.1) and (3.1),

$$\begin{aligned}\psi(\tau) &= \varphi(s) + \tau\tilde{g}^Tv + \tau\tilde{s}^T\tilde{H}v + \tfrac{1}{2}\tau^2v^T\tilde{H}v\\ &= \varphi(s) + \tau[\tilde{g} + (\tilde{H}+\alpha I)\tilde{s}]^Tv - \tau\alpha\tilde{s}^Tv + \tfrac{1}{2}\tau^2v^T\tilde{H}v\\ &= \varphi(s) - \tau\alpha\tilde{s}^Tv + \tfrac{1}{2}\tau^2v^T\tilde{H}v.\end{aligned}$$

Using (3.6) and (3.4a), we find

$$\begin{aligned}\psi(\tau_+) - \psi(\tau_-) &= -(\tau_+ - \tau_-)\alpha\tilde{s}^Tv + \tfrac{1}{2}(\tau_+ - \tau_-)(\tau_+ + \tau_-)v^T\tilde{H}v\\ &= -(\tau_+ - \tau_-)(\tilde{s}^Tv)(\alpha + v^T\tilde{H}v)\\ &= -2\sqrt{(\tilde{s}^Tv)^2 + \delta^2 - \|\tilde{s}\|^2}\,(\tilde{s}^Tv)v^T(\tilde{H}+\alpha I)v.\end{aligned}$$

But $\tilde{H}+\alpha I$ is positive definite, so $v^T(\tilde{H}+\alpha I)v > 0$ and $\psi(\tau_+) > \psi(\tau_-)$ if and only if $\tilde{s}^Tv < 0$. Thus the choice $\tau = -\tilde{s}^Tv + \text{sign}\,(\tilde{s}^Tv)\sqrt{(\tilde{s}^Tv)^2 + \delta^2 - \|\tilde{s}\|^2}$ minimizes ψ subject to $\|\tilde{s}+\tau v\| = \delta$, which is equivalent to $\tau = \sigma$ with σ given by (3.5b). □

To simplify the notation, we assume for most of the rest of this section that $D = I$ and $H = \text{diag}\,(h_1, h_2, \cdots, h_n)$, as in the proof of Theorem 2.1.

We now consider how to tell whether the θ of (3.2) is small enough that the relative difference between $\varphi(s^*)$ and $\varphi(\hat{s})$ is small, i.e., that

$$\varepsilon\varphi(\hat{s}) \leqq \varphi(s^*) - \varphi(\hat{s}) \leqq 0 \tag{3.7}$$

for some prescribed $\varepsilon \in (0, 1)$, e.g. $\varepsilon = 0.1$. To this end, it is convenient to define $\tilde{s}:[\alpha^*, \infty) \to \mathbb{R}^n$ by

$$\tilde{s}_i(\tau) = \begin{cases} s_i(\tau) = \dfrac{-g_i}{h_i+\tau} & \text{if } i < n,\\ -\text{sign}\,(g_n)\sqrt{\delta^2 - \sum_{j<n} s_j(\tau)^2} & \text{if } i = n.\end{cases} \tag{3.8}$$

In the event that $h_i = h_n$ for some $i < n$, we assume without losing generality that $g_i = s_i^* = 0$ and interpret (3.8) as specifying $\tilde{s}_i(\alpha^*) = 0$. If $g_n = 0$, then we assume that sign $(g_n) = -$sign (s_n^*) in (3.8). Thus, in all cases $\tilde{s}$ is a smooth function with $\tilde{s}(\alpha^*) = s^*$ and $\|\tilde{s}\| \equiv \delta$.

To bound $\varphi(\hat{s}) - \varphi(s^*)$, we shall first bound $\varphi(\tilde{s}(\alpha)) - \varphi(s^*)$, then bound $\varphi(\hat{s}) - \varphi(\tilde{s}(\alpha))$. To bound $\varphi(\tilde{s}(\alpha)) - \varphi(s^*)$, it is convenient to define $\psi : [\alpha^*, \infty) \to \mathbb{R}$ by

$$\begin{aligned} \psi(\tau) &:= \varphi(\tilde{s}(\tau)) - g_n \tilde{s}_n(\tau) \\ &= \sum_{i<n} \left(\frac{-g_i^2}{h_i + \tau} + \frac{h_i g_i^2}{2(h_i+\tau)^2} \right) + \frac{1}{2} h_n \left(\delta^2 - \sum_{i<n} \frac{g_i^2}{(h_i+\tau)^2} \right) \\ &= \frac{1}{2} h_n \delta^2 - \frac{1}{2} \sum_{i<n} g_i^2 (2\tau + h_i + h_n)/(h_i + \tau)^2. \end{aligned} \tag{3.9}$$

Note that $\psi'(\tau) = \sum_{i<n} g_i^2 (h_n + \tau)/(h_i + \tau)^3 \leqq \sum_{i<n} s_i^{*2} (h_n + \tau)/(h_i + \tau) \leqq \sum_{i<n} s_i^{*2} \leqq \delta^2$. Since $\alpha - \alpha^* \leqq \theta$ by (3.2), we thus find

$$\psi(\alpha) - \psi(\alpha^*) \leqq \theta \delta^2. \tag{3.10}$$

Now (3.2) and (3.3) imply

$$|g_n| \leqq (h_n + \alpha)\|s\| \leqq \theta\beta\delta, \tag{3.11}$$

while (3.8) implies $|\tilde{s}_n(\alpha^*) - \tilde{s}_n(\alpha)| \leqq \delta$. Together with (3.9) and (3.10), these yield

$$\begin{aligned} \varphi(\tilde{s}(\alpha)) - \varphi(s^*) &= \psi(\alpha) - \psi(\alpha^*) - g_n[\tilde{s}_n(\alpha^*) - \tilde{s}_n(\alpha)] \\ &\leqq (1+\beta)\theta\delta^2. \end{aligned} \tag{3.12}$$

Now we deduce a bound on $\varphi(\hat{s}) - \varphi(\tilde{s}(\alpha))$. For brevity, denote $\tilde{s}(\alpha)$ by $\tilde{s}$. Then (1.1) gives

$$\varphi(\hat{s}) - \varphi(\tilde{s}(\alpha)) = g^T(\hat{s} - \tilde{s}) + \tfrac{1}{2}\hat{s}^T H \hat{s} - \tfrac{1}{2}\tilde{s}^T H \tilde{s}. \tag{3.13}$$

To derive a convenient expression for $\hat{s}^T H \hat{s}$, it is useful to let

$$f = (H + \alpha I)v, \tag{3.14}$$

so that $Hv = f - \alpha v$. Note from (3.4b) that

$$\|f\| \leqq \kappa\theta. \tag{3.15}$$

Since $\hat{s} = s + \sigma v$ and $\|\hat{s}\| = \delta$, we have $\sigma^2 + 2\sigma s^T v = \delta^2 - \|s\|^2$ and

$$\begin{aligned} \hat{s}H\hat{s} &= s^T H s + \sigma^2 v^T H v + 2\sigma s^T H v = s^T H s + \sigma^2(v^T f - \alpha) + 2\sigma(s^T f - \alpha s^T v) \\ &= s^T H s - \alpha[\delta^2 - \|s\|^2] + \sigma[\sigma v + 2s]^T f. \end{aligned}$$

Similarly, we have $\tilde{s} = s + \tilde{\sigma} e_n$, where $e_n = (0, \cdots, 0, 1)^T$ is the nth standard unit vector of $\mathbb{R}^n$ and $\tilde{\sigma}$ is chosen so that $\|\tilde{s}\| = \delta$ (with sign $(\tilde{s}_n) =$ sign (s_n)), and we find

$$\tilde{s}^T H \tilde{s} = s^T H s + h_n[\delta^2 - \|s\|^2].$$

Together with (3.13), these equations imply

$$\begin{aligned} \varphi(\hat{s}) - \varphi(\tilde{s}(\alpha)) &= g^T(\hat{s} - \tilde{s}) - \tfrac{1}{2}(\alpha + h_n)(\delta^2 - \|s\|^2) + \tfrac{1}{2}\sigma(\sigma v + 2s)^T f \\ &\leqq g^T(\sigma v - \tilde{\sigma} e_n) + \tfrac{1}{2}\sigma(\sigma v + 2s)^T f. \end{aligned}$$

Now (3.1) and (3.14) give $g^T v = -s^T(H + \alpha I)v = -s^T f$, so

$$\varphi(\hat{s}) - \varphi(\tilde{s}(\alpha)) \leqq \tfrac{1}{2}\sigma^2 v^T f - \tilde{\sigma} g_n.$$

The definitions of σ and $\tilde{\sigma}$ imply $|\sigma| \leqq \delta$ and $|\tilde{\sigma}| \leqq \delta$, so (3.3), (3.4a), (3.11), and (3.15) combine with the above inequality to give

$$\varphi(\hat{s}) - \varphi(\tilde{s}(\alpha)) \leqq \tfrac{1}{2}\delta^2\kappa\theta + \delta^2\theta\beta = (\beta + \tfrac{1}{2}\kappa)\theta\delta^2.$$

Combining this with (3.12) and Theorem 2.1, we finally obtain

$$0 \leqq \varphi(\hat{s}) - \varphi(s^*) \leqq (2\beta + \tfrac{1}{2}\kappa + 1)\theta\delta^2. \tag{3.16}$$

This leads to

THEOREM 3.2. *Let $\beta \in (0, 1)$, $\varepsilon \in (0, 1)$, and $\kappa \in (1, \infty)$ be given. Suppose $g \neq 0$ and that $\alpha > 0$ renders $H + \alpha D^T D$ positive definite; also suppose that $s := -(H + \alpha D^T D)^{-1} g$ satisfies $\|Ds\| < \beta\delta$. If η is a lower bound on the largest eigenvalue of $-D^{-T} H D^{-1}$ and $\theta := \alpha - \eta$ satisfies*

$$\theta \leqq \varepsilon\delta^{-2}(\alpha\|Ds\|^2 - g^T s)/(4\beta + \kappa + 2), \tag{3.17}$$

then for any $v \in \mathbb{R}^n$ with $\|v\| = 1$ and $\|(D^{-T} H D^{-1} + \alpha I)v\| \leqq \kappa\theta$, it follows that

$$\hat{s} := s + \left(\frac{\delta^2 - \|Ds\|^2}{v^T Ds + \operatorname{sign}(v^T Ds)\{(v^T Ds)^2 + \delta^2 - \|Ds\|^2\}^{1/2}} \right) D^{-1} v$$

satisfies (3.7), *i.e.*, $\varepsilon\varphi(\hat{s}) \leqq \varphi(s^*) - \varphi(\hat{s}) \leqq 0$.

Proof. In view of (1.3), we shall assume $D = I$. From (1.1) we have

$$\begin{aligned} \varphi(s) &= g^T s + \tfrac{1}{2} s^T (H + \alpha I) s - \tfrac{1}{2}\alpha\|s\|^2 \\ &= g^T s - \tfrac{1}{2} g^T s - \tfrac{1}{2}\alpha\|s\|^2 \\ &= \tfrac{1}{2}(g^T s - \alpha\|s\|^2). \end{aligned}$$

Together with (3.16), (3.17), and the fact that $\varphi(s) < 0$, this implies $\varepsilon\varphi(s) \leqq \varphi(s^*) - \varphi(\hat{s})$. But Lemma 3.1 implies $\varphi(\hat{s}) \leqq \varphi(s)$, so (3.7) follows. □

4. Choosing α. If either H is indefinite or the Newton step $s^{(N)} = -H^{-1} g$ is too large, then computing an OLC step s^* requires finding a solution α^* to the scalar nonlinear equation $\|D(H + \alpha D^T D)^{-1} g\| = \delta$. There are many iterations for approximating such an α^* (see Gander's excellent discussion in [Gan 78, § 6]). In view of the fast convergence reported by Moré [Mor78], we prefer to use an iteration proposed by Reinsch [Rei71] and independently by Hebden [Heb73], together with Moré's (modification of Hebden's) safeguarding scheme. Let

$$\psi(\alpha) := \|D(H + \alpha D^T D)^{-1} g\|^{-1} - \delta^{-1}. \tag{4.1}$$

The basic iteration is Newton's method applied to ψ. Thus if iterate α_k renders $H + \alpha_k D^T D$ positive definite but yields an unacceptable step, then we compute a tentative value $\bar{\alpha}_{k+1} = \alpha_k - \psi(\alpha_k)/\psi'(\alpha_k)$ for α_{k+1}, i.e.

$$\bar{\alpha}_{k+1} = \alpha_k + \|Ds\|^2 \frac{\|Ds\| - \delta}{[\delta s^T D^T D(H + \alpha D^T D)^{-1} D^T Ds]}. \tag{4.2a}$$

where

$$s = s(\alpha_k) = -(H + \alpha_k D^T D)^{-1} g. \tag{4.2b}$$

We also maintain lower and upper bounds l_k and u_k on α^* and, if H is not positive definite, a value $\eta_k \geqq 0$ such that $-\eta_k$ is an upper bound on the smallest eigenvalue of $D^{-T} H D^{-1}$. (For convenience, we set $\eta_k = -1$ if H is positive definite.) We discuss below how these quantities are updated. Once l_{k+1} and u_{k+1} have been determined, we obtain

a safeguarded α_{k+1} from the rule

$$\alpha_{k+1} = \begin{cases} \bar{\alpha}_{k+1} & \text{if } l_{k+1} \leqq \bar{\alpha}_{k+1} < u_{k+1} \quad \text{and} \quad \bar{\alpha}_{k+1} > \eta_{k+1}, \\ \max\{10^{-3} \cdot u_{k+1}, (l_{k+1} u_{k+1})^{1/2}\} & \text{otherwise.} \end{cases} \tag{4.3}$$

Again assume variables have been changed so that $D = I$. To solve linear systems involving $H + \alpha_k I$, e.g., to compute $s(\alpha_k)$ in (3.1), we recommend attempting to compute the Cholesky (or LDL^T) decomposition of $H + \alpha_k I$ (see e.g. [Ste73, § 3.3]). If this works, then we may regard $H + \alpha_k I$ as numerically positive definite (provided the Cholesky factor has no zeros on the diagonal). Otherwise for some l between 1 and n we may express the leading principal $l \times l$ submatrix of $H + \alpha_k I$ as LML^T, where L is a lower triangular matrix with nonzero diagonal and M is the diagonal matrix $\operatorname{diag}(1, 1, \cdots, 1, \mu)$. Both L and μ are readily available as a byproduct of the attempted factorization, and $\mu \leqq 0$. We compute $z = L^{-T} e_l$ (i.e., solve $L_z^T = e_l$), where $e_l = (0, 0, \cdots, 0, 1)^T \in \mathbb{R}^l$. Then $\binom{z}{0}^T (H + \alpha_k I) \binom{z}{0} = z^T L M L^T z = e_l^T M e_l = \mu$, so $\mu / \|z\|^2$ is a Raleigh quotient for $H + \alpha_k I$, and $(\mu / \|z\|^2) - \alpha_k \leqq 0$ is an upper bound on the smallest eigenvalue of H. Hence we set

$$\eta_{k+1} = l_{k+1} = \alpha_k - \frac{\mu}{\|z\|^2}, \tag{4.4a}$$

$$u_{k+1} = u_k \tag{4.4b}$$

and choose $\bar{\alpha}_{k+1} = \alpha_k$ (which will force a safeguarded choice of α_{k+1} in (4.3)).

If $H + \alpha_k I$ is positive definite then the concavity of ψ [Rei71] implies that the Newton iterate $\bar{\alpha}_{k+1}$ given by (4.2) satisfies $\bar{\alpha}_{k+1} \leqq \alpha^*$. In this case we recommend using the following variation on Moré's update prescription for l_k and u_k: if $\psi(\alpha_k) < 0$, then choose

$$l_{k+1} = \bar{\alpha}_{k+1}, \tag{4.5a}$$

$$u_{k+1} = u_k. \tag{4.5b}$$

Otherwise choose

$$l_{k+1} = \max\{l_k, \bar{\alpha}_{k+1}\}, \tag{4.6a}$$

$$u_{k+1} = \alpha_k. \tag{4.6b}$$

In both cases, let $\eta_{k+1} = \eta_k$.

To obtain the initial bounds l_1 and u_1, we obtain lower and upper bounds EMIN and EMAX on the eigenvalues of H from the Gerschgorin circle theorem (optimized by the diagonal scaling technique described below), and we exploit the following observation: the α^* of Theorem 2.1 is such that $\delta = \|g\|/(\lambda + \alpha^*)$ for some λ between the smallest and largest eigenvalues of H. Thus

$$(\|g\|/\delta) - \text{EMAX} \leqq \alpha^* \leqq \frac{\|g\|}{\delta} - \text{EMIN},$$

and we let

$$l_1 = \max\{l_0, \frac{\|g\|}{\delta} - \text{EMAX}\}, \tag{4.7a}$$

$$u_1 = \frac{\|g\|}{\delta} - \text{EMIN}, \tag{4.7b}$$

where l_0 is given by whichever of (4.4a) or (4.5a) applies, with $\alpha_{-1} = l_{-1} = 0$.

To compute EMIN and EMAX, we use a special case of the scaling by diagonal matrices considered in [Var65]. Specifically, we find a diagonal matrix $\mathcal{D}$ having $n-1$ diagonal entries of unity and one other positive diagonal entry such that the Gerschgorin lower bound on the spectrum of $\mathcal{D}H\mathcal{D}^{-1}$ is as large as possible, and we use this bound as EMIN. This is quickly done (in $O(n^2)$ operations) as follows: Compute the off-diagonal row sums

$$\sigma_j = \sum_{i \neq j} |H_{ij}| \tag{4.8a}$$

and find k such that row k gives the minimum Gerschgorin lower bound:

$$H_{kk} - \sigma_k = \min \{H_{jj} - \sigma_j | 1 \leqq j \leqq n\}. \tag{4.8b}$$

For $j \neq k$, let θ_j denote the auxiliary quantity

$$\theta_j = \frac{H_{kk} - H_{jj} + \sigma_j - |H_{jk}|}{2} \tag{4.8c}$$

and compute

$$\text{EMIN} = H_{kk} - \max \{\theta_j + (\theta_j^2 + \sigma_k |H_{jk}|)^{1/2} | j \neq k\}. \tag{4.8d}$$

EMAX is computed similarly: with σ_j as in (4.8a), find k such that

$$H_{kk} + \sigma_k = \max \{H_{jj} + \sigma_j | 1 \leqq j \leqq n\}. \tag{4.9b}$$

For $j \neq k$, let

$$\theta_j = \frac{H_{jj} - H_{kk} + \sigma_j - |H_{jk}|}{2} \tag{4.9c}$$

and compute

$$\text{EMAX} = H_{kk} + \max \{\theta_j + (\theta_j^2 + \sigma_k |H_{jk}|)^{1/2} | j \neq k\}. \tag{4.9d}$$

We begin the quest for α^* by trying $\alpha = 0$. If this proves unsatisfactory, then we compute l_1 and u_1 by (4.7–9). If acceptable values $\alpha^{(\text{prev})}$ and $\delta^{(\text{prev})}$ of α and δ from a previously computed OLC step are available, then we obtain $\bar{\alpha}_1$ from a rule which J. J. Moré [private communication] has found helpful:

$$\bar{\alpha}_1 = \frac{\delta^{(\text{prev})} \alpha^{(\text{prev})}}{\delta}. \tag{4.10}$$

If $\alpha^{(\text{prev})}$ and $\delta^{(\text{prev})}$ are unavailable, then we simply set $\bar{\alpha}_1 = 0$.

To prevent excessive iterations, we deem the step $s = s^k$ computed from α_k acceptable if $\beta\delta \leqq \|s^k\| \leqq \gamma\delta$ for some specified $\beta \in (0, 1)$ and $\gamma \in (1, \infty)$. (Hebden [Heb73] and Moré [Mor78] choose $\beta = 0.9$ and $\gamma = 1.1$. In connection with an algorithm like that of NL2SOL [DenGW80], where $\delta/\|s^{(\text{prev})}\|$ or $\delta/\delta^{(\text{prev})}$ either equals unity or two or lies in $[0.1, 0.5]$, Dennis and Schnabel suggest $\beta = 0.75$ and $\gamma = 1.5$ [DenS79]. Our computational experience with NL2SOL slightly favors the former choice.)

In practice, D is usually a diagonal matrix, so the explicit change of variables (1.3) is easily performed, and we recommend actually performing it when H is given explicitly. When H has the form J^TJ, g has the form J^Tr, and J and r are given explicitly, on the other hand, we prefer the technique advocated by Moré [Mor78], i.e., using a QR factorization of $\begin{bmatrix} J \\ \alpha D^{1/2} \end{bmatrix}$ to compute s.

When H is given explicitly and $g \neq 0$, the method described above for computing a reasonable approximation s to s^* may be summarized as follows:

ALGORITHM 4.1.
Compute $H := D^{-T}HD^{-1}$ and $g := D^{-T}g$.
If H is positive definite, then:
  Compute the Newton step $s^{(N)} = -H^{-1}g$.
  If $\|s^{(N)}\| \leqq \gamma\delta$, then halt and return $s := D^{-1}s^{(N)}$.
  Set $\eta_1 = -1$ and determine l_0 from (4.5) with $\alpha_{-1} = l_{-1} = 0$.
Else [H not positive definite] compute η_0 and l_0 from (4.4) with $\alpha_{-1} = l_{-1} = 0$ and set $\eta_1 = \eta_0$.
Compute EMIN and EMAX by (4.8) and (4.9), and compute l_1 and u_1 from (4.7).
If $\alpha^{(\text{prev})}$ and $\delta^{(\text{prev})}$ are available, compute $\bar{\alpha}_1$ from (4.10); otherwise let $\bar{\alpha}_1 = 0$.
Compute α_1 from (4.3).
For $k = 1, 2, \cdots$
  If $H + \alpha_k I$ is positive definite, then:
    Set $\eta_{k+1} = \eta_k$ and compute $s^{(k)} = -(H + \alpha_k I)^{-1}g$.
    If $\beta\delta \leqq \|s^{(k)}\| \leqq \gamma\delta$, then halt and return $s := D^{-1}s^{(k)}$.
    Compute $\bar{\alpha}_{k+1}$ from (4.2) with $D := I$.
    If $\|s^{(k)}\| < \beta\delta$, then
      If $\eta_k \geqq 0$ and $\theta := \alpha_k - \eta_k$ satisfies
        (3.17) with $s = s^{(k)}$, then
        Compute v satisfying (3.4).
        Compute $\hat{s}$ from (3.5).
        Halt and return $s := D^{-1}\hat{s}$.
      Compute l_{k+1} and u_{k+1} from (4.6).
    Else [$\|s^{(k)}\| > \gamma\delta$] compute l_{k+1} and u_{k+1} from (4.5).
  Else [$H + \alpha_k I$ not positive definite] set $\bar{\alpha}_{k+1} = \alpha_k$ and compute η_{k+1}, l_{k+1}, u_{k+1} from (4.4).
  Determine α_{k+1} from (4.3).

After an OLC step has been computed, it is often necessary to compute another OLC step from the same g and H with a new value of δ. To handle this situation, it is worthwhile to modify the initial part of Algorithm 4.1 to take advantage of the extra information that is available. For example, if δ has been increased, then u_1 can be set to the minimum of the value given by (4.7b) and the last value $u^{(\text{old})}$ of u_k, while if δ has been decreased, l_1 can be set to the maximum of the value given by (4.7a) and the last value $l^{(\text{old})}$ of l_k. In either case, η_1 can be set to $\eta^{(\text{old})}$ and $\bar{\alpha}_1$ can be computed from (4.2) with $\alpha_0 = \alpha^{(\text{old})}$.

5. Numerical experience. For use in NL2SOL [DenGW80], we have implemented two versions of Algorithm 4.1: GQTSTP is designed for use with a general H, while LMSTEP deals explicitly with J (or its QR decomposition) when H has the form J^TJ, g has the form J^Tr, and J is a rectangular matrix with at least as many rows as columns. Both codes make special provision for the case in which an OLC step is to be recomputed with a new δ but the same g and H. To handle D, which both codes assume to be a diagonal matrix, GQTSTP explicitly changes variables, whereas LMSTEP follows the procedure recommended by Moré [Mor78]. (Both codes use (3.17) with 4β replaced by $4\beta(\kappa + 1)$. A referee pointed out a cancellation that let us change the old $4\beta(\kappa + 1)$ in (3.17) to the present 4β.)

The way these codes deal with (near) singularity in $H + \alpha^* D^T D$ deserves further discussion. Both detect this case by test (3.17) [with $\beta := \beta(\kappa + 1)$]. In the case of

LMSTEP, $H = J^TJ$ never has a negative eigenvalue, so in the event of true singularity in $H + \alpha^* D^T D$, we would have $\alpha^* = 0$ and $D^{-T}HD^{-1}$ positive semidefinite, and any v in the null space of $D^{-T}HD^{-1}$ would be orthogonal to $D^{-T}g$, whence $\varphi(s + D^{-1}v) = \varphi(s)$. LMSTEP therefore returns without modification a step s for which (3.3) and (3.17) hold. GQTSTP similarly avoids replacing s by $\hat{s} = s + \sigma D^{-1}v$ in cases where $\varphi(s)$ and $\varphi(\hat{s})$ would not differ significantly. Specifically, if $\varphi(s) - \varphi(\hat{s}) < -(\varepsilon/3)\varphi(s)$, then GQTSTP returns s rather than $\hat{s}$. To assure that (3.7) holds in this case, GQTSTP uses (3.17) with ε replaced by $2\varepsilon/3$. Both codes used $\varepsilon = 0.1$ and $\kappa = 2$ in the tests reported below.

We have also used GQTSTP in HUMSOL [Gay80], a code for solving general unconstrained minimization problems in cases where the Hessian of the objective function is explicitly available.

Table I gives some statistics on the performance of LMSTEP and GQTSTP in NL2SOL and HUMSOL. The column headed "Test" tells which test problems and

TABLE I
Statistics from test problems.

				All (Non-Newton) OLC Steps				Special Case		
Test	Module	β	γ	No. of Steps	% of Total	k mean	k max	% of OLC	k mean	k max
A	LMSTEP	0.9	1.1	1228	80.6	1.46	5	5.5	1.49	3
A	LMSTEP	0.75	1.5	1207	79.2	1.30	4	3.8	1.54	3
A	GQTSTP	0.9	1.1	254	57.7	1.85	9	2.8	1.86	4
A	GQTSTP	0.75	1.5	351	55.1	1.51	7	2.6	1.44	3
B	GQTSTP	0.9	1.1	949	55.0	2.00	10	0	—	—
C	GQTSTP	0.9	1.1	831	98.2	3.59	19	8.1	10.24	18

optimizer were used in obtaining the results in the corresponding row. Test A is NL2SOL running on the same problem set (and under the same conditions) used to generate [DenGW80, Table II]. Tests B and C involve HUMSOL running on some of the unconstrained minimization test problems described in [MorGH81] (with $D = I$): test set B consists of all the problems and starting guesses suggested in [MorGH81] with the exceptions of Biggs EXP6, Penalty function II with $n = 4$, 10, and Chebyquad with $n = 8, 9, 10$. Test set C consists of Biggs EXP6 and Chebyquad with $n = 8, 9, 10$ (all with the standard starting guess). The column headed "No. of Steps" gives the total number of non-Newton steps computed (those for which α is positive in (3.1)), and the column labeled "% of Total" tells what percentage these were of all the steps computed by the module in question. The average value of k when Algorithm 4.1 halted (averaged over the non-Newton steps) appears in the column labeled "k mean", and the maximum such value appears under "k max". In the columns headed "Special Case" are the percentage of non-Newton steps in which (near) singularity in $H + \alpha^* D^T D$ was detected and the mean and maximum final values of k for these steps. (All computations were performed on the IBM 370/168 at the Massachusetts Institute of Technology using double precision versions of NL2SOL and HUMSOL compiled by FORTHX with OPT = 2.)

The problems in test set C deserve special comment. On Biggs EXP6, the starting guess given to HUMSOL and all subsequent iterates that it generated were points at which H was indefinite. In the 378 steps that it computed for this problem (before our function evaluation limit was reached), GQTSTP detected the special case 59 times at

widely scattered iterates. For the Chebyquad problems, H was also indefinite at the starting guess and the special case was detected on the first step computed. The relatively poor behavior of Algorithm 4.1 on these problems (in terms of the number of iterations it took) raises the question of whether one can significantly improve upon this algorithm, but the poor behavior is sufficiently rare that we believe Algorithm 4.1 to be useful as it stands.

Had the mean values of k been much greater than two in the case of LMSTEP or four in the case of GQTSTP, then it would have been possible to save some time in these modules by preprocessing the input matrices to a sparser form: by reducing J to bidiagonal form in LMSTEP or reducing $D^{-1}HD^{-1}$ to tridiagonal form in GQTSTP (see [Ste73, §§ 7.5 and 7.1 and the references cited therein]).

After all the effort we spent in studying the special case, it is somewhat disappointing to see it detected so rarely by GQTSTP. On the other hand, checking for it in subroutines like LMSTEP and GQTSTP appears well worthwhile, since it is often detected at much smaller values of k than the limit that one otherwise might impose.

Acknowledgments. I heartily thank Jorge J. Moré for some helpful discussions, Burton S. Garbow for providing subroutines for the test problems used in testing HUMSOL, and the referees for their constructive comments.

REFERENCES

[DenGW80] J. E. DENNIS, D. M. GAY AND R. E. WELSCH. *An adaptive nonlinear least-squares algorithm*, Tech. Rep. TR–20, Center for Computational Research in Economics and Management Science, Mass. Institute of Technology, Cambridge, 1980, ACM Trans. Math. Software, to appear.

[DenS79] J. E. DENNIS AND R. B. SCHNABEL, *Quasi-Newton Methods for Unconstrained Nonlinear Problems*, draft of forthcoming book.

[Gan78] W. GANDER, *On the linear least squares problem with a quadratic constraint*, Tech. Rep. STAN-CS-78-697, Computer Science Dept., Stanford Univ., Stanford, CA, 1978.

[Gay80] D. M. GAY, *Subroutines for unconstrained minimization using a model/trust-region approach*, Tech. Rep. TR-18, Center for Computational Research in Economics and Management Science, Mass. Institute of Technology, Cambridge, 1980.

[GolQT66] S. M. GOLDFELD, R. E. QUANDT AND H. F. TROTTER, *Maximization by quadratic hill-climbing*, Econometrica, 34 (1966), pp. 541–551.

[Heb73] M. D. HEBDEN, *An algorithm for minimization using exact second derivatives*, Rep. TP 515, A.E.R.E. Harwell, England, 1973.

[Lev44] K. LEVENBERG, *A method for the solution of certain nonlinear problems in least squares*, Quart. Appl. Math., 2 (1944), pp. 164–168.

[Mar63] D. W. MARQUARDT, *An algorithm for least squares estimation of nonlinear parameters*, SIAM J. Appl. Math., 11 (1963), pp. 431–441.

[McC76] G. P. MCCORMICK, *Optimality criteria in nonlinear programming*, in *Nonlinear Programming*, SIAM–AMS Proc. vol. IX, R. W. Cottle and C. E. Lemke, eds., American Mathematical Society, Providence, RI, 1976, pp. 27–38.

[MorGH81] J. J. MORÉ, B. S. GARBOW AND K. E. HILLSTROM, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.

[Rei71] C. H. REINSCH, *Smoothing by spline functions*. II, Numer. Math., 16 (1971), pp. 451–454.

[Ste73] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[Var65] R. S. VARGA, *Minimal Gerschgorin sets*, Pacific J. Math., 15 (1965), pp. 719–729.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0008 $01.00/0

FAST NUMERICAL SOLUTION OF TIME-PERIODIC PARABOLIC PROBLEMS BY A MULTIGRID METHOD*

WOLFGANG HACKBUSCH†

Abstract. The discrete solutions of parabolic problems subject to the condition $y(\cdot, T) = y(\cdot, 0)$ of time periodicity are solutions of large sparse systems. In this paper we propose a multigrid algorithm. It is a very fast iterative method. The algorithm can easily be generalized to nonlinear problems and to conditions of the type $y(\cdot, 0) = A(y(\cdot, T))$ (A is a nonlinear mapping). The computational work for solving the periodic problem is of the same order as the work for solving an initial value problem ($y(\cdot, 0)$ given). Numerical results are reported for a linear and a nonlinear example.

Key words. multigrid method, numerical solution, parabolic differential equations, periodic time condition, time-periodic parabolic problem

1. Introduction.

We consider the periodic parabolic equation

$$\frac{\partial y}{\partial t} + Ly = g_1, \qquad (x, t) \in Q, \tag{1.1a}$$

$$By = g_2, \qquad (x, t) \in \Sigma, \tag{1.1b}$$

$$y(\cdot, T) = y(\cdot 0), \qquad x \in \Omega, \tag{1.1c}$$

where

$$\Omega \subset \mathbb{R}^n, \quad \Gamma = \partial\Omega, \quad Q = \Omega \times (0, T), \quad \Sigma = \Gamma \times (0, T).$$

$L = L(x, t)$ is an elliptic differential operator, e.g.,

$$L(x, t) = \sum_{i,j=1}^{n} \frac{\partial}{\partial x_j} a_{ij}(x, t) \frac{\partial}{\partial x_i} + \sum_{i=1}^{n} a_i(x, t) \frac{\partial}{\partial x_i} + a(x, t), \tag{1.2}$$

whereas $B = B(x, t)$ is a boundary operator, for instance, $Bu = u$ or $Bu = \partial u/\partial n$. (1.1c) is the requirement of T-periodicity[1].

We will also treat the nonlinear problem (1.3a, b), (1.1c):

$$\frac{\partial y}{\partial t} + L(y) = 0, \qquad (Q), \tag{1.3a}$$

$$B(y) = 0, \qquad (\Sigma). \tag{1.3b}$$

For a discretization of (1.1a–c) we have to introduce step sizes Δt and Δx. Assume $\Delta x = h$, $\Delta t = \lambda h^2$. Let Ω_h be a grid of width Δx. We define

$$I_h = \{\nu \cdot \Delta t \in (0, T]: \nu \text{ integer}\}, \qquad Q_h = \Omega_h \times I_h.$$

Then a suitable finite difference (or finite element or least square) discretization is of the form

$$M_h y_h = g_h, \qquad (Q_h), \tag{1.4a}$$

$$y_h(\cdot, T) = y_h(\cdot, 0), \qquad (\Omega_h); \tag{1.4b}$$

* Received by the editors October 14, 1980.

† Mathematisches Institut, Ruhr-Universität Bochum, Postfach 102148, D-4630 Bochum 1, Germany.

[1] This condition may be generalized to $u(\cdot, 0) = Au(\cdot, T) + a$ for a suitable operator A. Also a nonlinear equation $u(\cdot, 0) = A(u(\cdot, T))$ is permitted.

e.g., $(M_h y_h)(x, t) = [y_h(x, t) - y_h(x, t - \Delta t)]/\Delta t + L_h(x, t) y_h(x)$, where L_h is a discrete counterpart of L.

In the case of the nonlinear problem (1.3a, b), (1.1c) the equation (1.4a) becomes

$$M_h(y_h) = 0 \qquad (Q_h). \tag{1.5a}$$

Numerical methods for solving the system (1.4a, b) of difference equations are proposed, e.g., by Tee [12] and Osborne [7], [8]. Another iterative method due to the author is explained in [2]. Nonlinear periodic problems are studied in the recent paper [10] of Steuerwalt.

Here we propose a fast method for solving the linear problem (1.4) as well as the nonlinear problem (1.5). The algorithm is iterative, but usually one step of the iteration yields sufficient accuracy. The method is based on the representation of the periodic problem (1.1a–c) and of its discretization (1.4a, b) by the equations (2.2) and (2.4), respectively, defined in § 2. Sections 3 and 5 contain the description of the linear and nonlinear algorithm. Numerical examples are reported in §§ 4 and 6. Further applications of this multigrid iteration are described in [3], [6], and in [5, refs. 7, 8, 10].

2. Reformulation of the problem. We introduce a certain linear mapping K such that the original problem is equivalent to (2.2) given below. It must be emphasized that this mapping K is only of theoretical interest. We need *no* explicit representation of K.

Consider the usual initial-boundary value problem (1.1a, b) with

$$y(x, 0) = y_0(x), \qquad x \in \Omega. \tag{2.1}$$

Denote the result of (1.1a, b) and (2.1) at $t = T$ by $\mathfrak{R}(y_0)$:

$$\mathfrak{R}(y_0) := y(\cdot, T).$$

Obviously, $\mathfrak{R}(\cdot)$ is an affine mapping. Hence, it can be rewritten as

$$\mathfrak{R}(y_0) = K y_0 + f$$

where $f(x) := y(x, T)$ is the result of (1.1a, b) and (2.1) at $t = T$ in the case of $y_0(x) = 0$. Ky_0 is $y(\cdot, T)$, where y is the solution of (1.1a, b) (with $g_1 = 0$, $g_2 = 0$) and (2.1).

With this notation the condition (1.1c) becomes

$$u = Ku + f, \tag{2.2}$$

where $u = y(\cdot, 0) = y(\cdot, T)$ is a function defined on Ω. Equation (2.2) is equivalent to the original problem in the following sense.

STATEMENT 2.1. *If y is a solution of* (1.1a–c), *then $u = y(\cdot, T)$ satisfies* (2.2). *If u fulfils* (2.2), *then the solution y of the initial-boundary value problem* (1.1a, b) *and* (2.1) *with $y_0 = u$ is a solution of* (1.1a–c).

Obviously, (2.2) is the formulation of the periodic parabolic equation as an integral equation. K is the integral operator $(Ky_0)(x) = \int_\Omega k(x, T; \xi) y_0(\xi)\, d\xi$, where the kernel $k(x, t; \xi)$ satisfies (1.1a, b) with $g_1 = 0$, $g_2 = 0$ and $k(x, 0; \xi) =$ Dirac's delta function at $x = \xi$.

The discrete initial-boundary value problem consists of (1.4a) and

$$y_h(x, 0) = y_{0h}(x), \qquad x \in \Omega_h. \tag{2.3}$$

By similar considerations we obtain the equivalent discrete equation

$$u_h = K_h u_h + f_h, \tag{2.4}$$

where $u_h = y_h(\cdot, 0) = y_h(\cdot, T)$. $K_h y_{0h} + f_h$ is the solution of (1.4a) and (2.3) restricted to

$t = T$. If, for instance, the discretization (1.4a) for the homogeneous problem ($g_1 = 0$, $g_2 = 0$) is the implicit scheme

$$\frac{y_h(x,t) - y_h(x,t-\Delta t)}{\Delta t} + L_h(x) y_h(x,t) = 0, \qquad x \in \Omega_h, \qquad t \in I_h$$

with L_h independent of t, then K_h is the matrix

$$K_h = [I + \Delta t L_h]^{-T/\Delta t} \qquad (I = \text{identity matrix}).$$

Hence, the computation of $K_h y_{0h}$ requires $T/\Delta t$-times the solution of a linear equation $[I + \Delta t L_h] y_h = g_h$ with different g_h's.

For the following it is important to study the "smoothing property" of K (cf. Wahlbin [13]).

STATEMENT 2.2. *Under usual assumptions on the differential operator L (of second order, cf. (1.2)) and B, the operator K is a bounded mapping from $L^2(\Omega)$ into the Sobolev space $H^1(\Omega)$.*

Proof. For the case of $L = L(x)$ independent of t compare Bramble and Thomée [1]. The general case is treated by Tanabe [11, (5.136)].

COROLLARY 2.1. *If $n = 1$ [i.e., $\Omega = (x_0, x_1) \subset \mathbb{R}$] or if $\partial\Omega$ is a sufficiently smooth boundary, Statement 2.2 holds with $H^1(\Omega)$ replaced by $H^2(\Omega)$.*

Let $L_h^2(\Omega_h)$ and $H_h^s(\Omega_h)$ be the discrete counterparts of $L^2(\Omega)$ and $H^s(\Omega)$, respectively. The discrete analogue of Statement 2.2 is

STATEMENT 2.3. *If (1.4a) is the implicit difference scheme (or implicit Galerkin method), the matrix K_h of (2.4) is a mapping from $L_h^2(\Omega_h)$ into $H_h^1(\Omega_h)$ [or $H_h^2(\Omega_h)$ (cf. Corollary 2.1) or at least $H_h^{1+\theta}(\Omega_h)$ ($\theta < \frac{1}{2}$, cf. [4])] uniformly bounded with respect to h:*

$$\|K_h\|_{L_h^2(\Omega_h) \to H_h^s(\Omega_h)} \leqq C, \qquad s = 1, 1+\theta, \text{ or } 2. \tag{2.5}$$

Here and in the following C denotes a generic constant not depending on the discretization parameter h.

An easy proof of (2.5) can be given in the case of the model problem

$$\begin{aligned} &u_t - u_{xx} = g_1(x,t), && 0 < x < \pi, \quad 0 < t < T, \\ &u(x,t) = g_2(x,t) \quad \text{at } x = 0, \pi, && 0 < t < T, \\ &u(x,0) = y_0(x), && 0 < x < \pi \end{aligned}$$

discretized by the *implicit scheme*

$$\Delta t^{-1}[u(x,t) - u(x,t-\Delta t)] - h^{-2}[u(x-h,t) - 2u(x,t) + u(x+h,t)] = g_1(x,t).$$

Setting $g_1 = 0$, $g_2 = 0$, and

$$y_0(x) = \sum_{\nu=1}^{N-1} c_\nu \sin \nu x, \qquad N = \frac{\pi}{h},$$

gives that the amplification factor of each $\sin \nu x$-term is

$$\left(1 + 4\Delta t h^{-2} \sin^2 \frac{\nu h}{2}\right)^{-T/\Delta t}, \qquad 0 < \nu < N$$

(cf. Richtmyer and Morton [9]). Inequality (2.5) holds if and only if the amplification factor is $\leqq C(1+\nu^2)^{-s/2}$. Obviously, this is true with $s \leqq 2$ for the implicit scheme without any limitation to Δt.

In the case of the *Crank–Nicolson difference scheme* (cf. [9]) the amplification factor becomes

$$\left[\left(1-2\Delta t h^{-2}\sin^2\frac{\nu h}{2}\right)\Big/\left(1+2\Delta t h^{-2}\sin^2\frac{\nu h}{2}\right)\right]^{-T/\Delta t}, \qquad 0<\nu<N.$$

This expression is $\leqq C(1+\nu^2)^{-1}$ if the time step Δt is restricted by

$$\Delta t \leqq \frac{\text{const}\cdot h}{\log(h^{-1})}.$$

For $\Delta t = \text{const}\cdot h$ we obtain inequality (2.5) only with $s=0$.

The *explicit difference scheme*

$$\Delta t^{-1}[u(x,t)-u(x,t-\Delta t)]-h^{-2}[u(x-h,t-\Delta t)-2u(x,t-\Delta t)+u(x+h,t-\Delta t)] = g_1(x,t)$$

is stable only if $\Delta t \leqq h^2/2$. For the strengthened condition

$$\Delta t \leqq \lambda h^2, \qquad \lambda < \tfrac{1}{2}$$

one can prove that the amplification factor $|1-4\Delta t h^{-2}\sin^2(\nu h/2)|^{T/\Delta t}$ is bounded by $C(1+\nu^2)^{-1}$. Hence, (2.5) holds for $s \leqq 2$.

3. Linear algorithm (multigrid iteration of the second kind). Let $h_0 > h_1 > \cdots > h_l > \cdots$ be a sequence of decreasing step sizes, e.g.,

$$h_l = \Delta x_l = \frac{h_0}{2^l}, \qquad \Delta t_l = \lambda h_l^2 = \frac{\Delta t_0}{4^l} \qquad (l = \text{level number}). \tag{3.1}$$

We replace the subscript h_l by l:

$$L_l^2 = L_{h_l}^2(\Omega_{h_l}), \qquad H_l^s = H_{h_l}^s(\Omega_{h_l}),$$
$$K_l = K_{h_l}, \quad u_l = u_{h_l}, \quad f_l = f_{h_l}, \quad y_l = y_{h_l}, \quad \Omega_l = \Omega_{h_l}.$$

Define the prolongation

$$p_{l,l-1}: L_{l-1}^2 \to L_l^2$$

by piecewise linear interpolation, and let

$$r_{l-1,l}: L_l^2 \to L_{l-1}^2$$

be the restriction to the coarser grid Ω_{l-1}. Then

$$\|I - p_{l,l-1}r_{l-1,l}\|_{H_l^s \to L_l^2} \leqq Ch_l^s, \qquad 0 \leqq s \leqq 2 \tag{3.2}$$

is valid. For the implicit discretization with $\Delta t = \lambda \Delta x^2$ the following consistency condition holds:

$$\|K_{l-1}r_{l-1,l} - r_{l-1,l}K_l\|_{H_l^s \to L_{l-1}^2} \leqq Ch_{l-1}^s, \qquad 0 \leqq s \leqq 2. \tag{3.3}$$

STATEMENT 3.1. *Let h_0 be small enough. The assumptions* (2.5), (3.2), *and* (3.3) *are the main conditions for proving that the following algorithm has a rate of convergence proportional to h_l^s.*

Proof. Compare [3].

The following algorithm is explained in [3]:

ALGORITHM. Let l be the actual level number, i be the number of iterations, and f be the right-hand side f_l of (2.4). The input value u contains the starting value $u_l^{(k)}$ (kth iterate), while the output value is $u_l^{(k+i)}$ ($(k+i)$th iterate).

procedure mgm (i, l, u, f); **integer** i, l; **array** u, f;
if $l = 0$ **then** $u := (I - K_0)^{-1} * f$ **else**
begin integer j; **array** d, v;
  for $j := 1$ **step** 1 **until** i **do**
  begin $u := K_l * u + f$; $d := r_{l-1,l} * (u - K_l * u - f)$;
    $v := 0$; mgm $(2, l-1, v, d)$;
    $u := u - p_{l,l-1} * v$;
end end;

Since the rate of convergence rapidly tends to zero (cf. Statement 3.1), one step of the iteration usually yields a result with sufficiently small iteration error, provided that the coarsest grid size h_0 is small enough. It is advisable to call the procedure mgm once at every level $1, 2, \cdots, l$:

$$u_0 := (I - K_0)^{-1} * f_0;$$

(3.4) **for** $j := 1$ **step** 1 **until** l **do**

begin $u_j := p_{j,j-1} * u_{j-1}$; mgm $(1, j, u_j, f_j)$ **end**;

The procedure mgm requires a repeated performance of the mapping $v_l \mapsto K_l v_l + f_l$. If $h = h_l$ is the finest grid size, i.e., the grid size h of the original discrete problem (1.4a, b), the mapping $v_l \mapsto K_l v_l + f_l := y_l(\cdot, T)$ requires the solution of (1.4a) with $y_l(\cdot, 0) := v_l$ instead of (1.4b). The computation of f_l is not necessary.

If $h_l > h$ is an auxiliary grid size, $K_l v_l$ can be computed by solving the *homogeneous* problem (1.4a) with $g_h = 0$ and $y_l(\cdot, 0) := v_l$. Then $K_l v_l$ results from $y_l(\cdot, T)$.

For the coarsest grid size h_0 the solution of $(I - K_0)^{-1} f_0$ can be obtained in two different ways. The *first method* requires the computation of the matrix $I - K_0$ and its LU-decomposition in a preprocessing phase. Then $(I - K_0)^{-1} f_0$ is computed as $U^{-1} L^{-1} f_0$. This method is preferred if Δx_0 is coarse enough, i.e., if the number of equations (=dim L_0^2) is small. The *second method* can be applied if any other method for solving (1.4a, b) ($h = h_0$) is available. Let $h := h_0$. For given f_0 define the right-hand side g_h of (1.4a) by $g_h := M_h \tilde{y}_h$, where $\tilde{y}_h(\cdot, t) := 0$ for $t < T$ and $\tilde{y}_h(\cdot, T) := f_0$. Solve (1.4a, b) with this choice of g_h resulting in y_h. Then $u_0 = (I - K_0)^{-1} f_0$ is the initial and final value of y_h: $u_0 = y_h(\cdot, T)$.

STATEMENT 3.2. (Operation count). *One step of the algorithm* mgm *at the level l requires 2-fold solving of $v_l \mapsto K_l v_l + f_l$ at $h = h_l$ and $3 \cdot 2^{\nu-1}$-fold solving of $v_{l-\nu} \mapsto K_{l-\nu} v_{l-\nu}$ at $h = h_{l-\nu}$ ($\nu = 1, 2, \cdots, l-1$). At the lowest level, $h = h_0$, $3 \cdot 2^{l-2}$ linear systems $(I - K_0)^{-1} f_0$ are to be solved.*

If the computational work of $v_l \mapsto K_l v_l$ is

$$W_l \approx C \Delta t_l^{-1} \Delta x_l^{-n} = C \lambda^{-1} h_l^{-n-2} = C' \cdot 2^{(n+2)l} \qquad (cf.\ (3.1)),$$

and if the work taken by solving $(I - K_0)^{-1} f_0$ is neglected, one step of the iteration requires work of

$$I_t = \left(2 + \frac{3}{2 - 2^{-n}}\right) W_l.$$

For $n = 1$ (one space variable) we have $I_l \leqq 4W_l$. The use of the program (3.4) is only slightly more expensive than only one iteration at the level l. Under the condition (3.5) the work required by (3.4) amounts to $[1-2^{-n-2}]^{-1} I_l \leqq \frac{8}{7} I_l \leqq 4.6\, W_l$.

4. Numerical example for the linear algorithm. Consider the periodic problem

$$y_t = y_{xx} + \frac{2t-1}{2-x}\left[x(1-x) + \frac{2t-1}{(2-x)^2}\right], \qquad 0 \leqq t \leqq 1, \quad 0 \leqq x \leqq 1, \tag{4.1a}$$

$$y(0, t) = y(1, t) = 0, \qquad 0 \leqq t \leqq 1, \tag{4.1b}$$

$$y(x, 0) = y(x, 1), \qquad 0 \leqq x \leqq 1. \tag{4.1c}$$

The exact solution is $y(x, t) = (t - \frac{1}{2})^2 x(1-x)/(2-x)$.

According to (3.1) we choose the step sizes

$$h_l = \Delta x_l = 2^{-2-l}, \qquad \Delta t_l = 4^{-l}.$$

The finest grid ($l = 4$) is defined by $h = \Delta x = \frac{1}{64}$ and $\Delta t = \frac{1}{256}$. The rates ρ_l of convergence of the multigrid iteration at the level l are: $\rho_1 = 4\text{E}-4$, $\rho_2 = 2\text{E}-6$, $\rho_3 = 2\text{E}-7$, $\rho_4 = 6\text{E}-8$. Table 1 contains the results u_j ($1 \leqq j \leqq 4$) of the loop (3.4). The total error is $\max\{|u_l(x) - y(x, 0)|: x = 0, \Delta x_l, 2\Delta x_l, \cdots, 1\}$. It is the sum of the discretization error and of the iteration error. But because of the very fast convergence the iteration error is much smaller than the discretization error, as can be seen from the last column.

TABLE 1

Errors of the first iterate $u_l^{(1)}$ resulting from (3.4) for the example (4.1a–c)

l	$h = \Delta x_l$	Δt_l	total error	iteration error
0	$\frac{1}{4}$	1	1.71E−2	—
1	$\frac{1}{8}$	$\frac{1}{4}$	4.19E−3	3.4E−8
2	$\frac{1}{16}$	$\frac{1}{16}$	1.04E−3	1.7E−12
3	$\frac{1}{32}$	$\frac{1}{64}$	2.61E−4	1.5E−14
4	$\frac{1}{64}$	$\frac{1}{256}$	6.52E−5	0

5. Nonlinear algorithm. Let (1.5a) and (1.4b) be the discretization of the nonlinear periodic problem (1.3a, b), (1.1c). According to § 2 we denote the mapping $y_h(\cdot, 0) \mapsto y_h(\cdot, T)$ [y_h solution of (1.5a)] by $\mathfrak{R}_h(\cdot)$: $y_h(\cdot, T) = \mathfrak{R}_h(y_h(\cdot, 0))$. Again, the problem (1.5a), (1.4b) is equivalent to $u_h = \mathfrak{R}_h(u_h)$. Replacing the subscript $h = h_l$ by l we write

$$u_l = \mathfrak{R}_l(u_l). \tag{5.1}$$

In the following we need the more general problem

$$u_l = \mathfrak{R}_l(u_l) + f_l, \tag{5.2}$$

where f_l is some *small* perturbation.

Assume that (5.2) has at least one solution, denoted by $u_l = \phi_l(f_l)$. For $l = 0$ there must be some iteration

$$v_0 \mapsto \tilde{\phi}_0(v_0, f_0) \tag{5.3}$$

that converges to the solution $\phi_0(f_0)$ of (5.2), e.g., Newton's iteration.

The nonlinear multigrid method nmgm described below can be used as proposed in (3.4):

(5.4)
```
ũ_0 := suitable approximation of u_0 = φ_0(0);
for j := 1 step 1 until l do
begin f̃_{j-1} := ũ_{j-1} − ℜ_{j-1}(ũ_{j-1});
      ũ_j := p_{j-1,j} * ũ_{j-1}; nmgm (1, j, ũ_j, 0);
end;
```

The parameter of nmgm have the same meaning as those of the procedure mgm. $\tilde{u}_{l-1}$ must be an approximation of $u_{l-1} = \phi_{l-1}(0)$ (=solution of (5.1)). $\tilde{f}_{l-1}$ is the defect $\tilde{u}_{l-1} - \mathfrak{R}_{l-1}(\tilde{u}_{l-1})$ of $\tilde{u}_{l-1}$.

```
procedure nmgm (i, l, u, f): integer i, l; array u, f;
if l = 0 then u := φ̃_0(u, f) else
begin integer j; real s; array d, v;
    for j := 1 step 1 until i do
    begin u := ℜ_l(u) + f;
        d := r_{l-1,l} * (u − ℜ_l(u) − f);
        s := if ‖d‖ = 0 then 1 else ρ/‖d‖;
        d := f̃_{l-1} + s * d;
        v := ũ_{l-1}; nmgm (2, l − 1, v, d);
        u := u − p_{l,l-1} * (v − ũ_{l-1})/s
end end;
```

$\|\cdot\|$ is the norm of L_l^2. The positive number ρ ensures that $\|d\| \leqq \|\tilde{f}_{l-1}\| + \rho$ is sufficiently small. For a discussion of this algorithm we refer to [5]. Another application of nmgm is reported in [6].

Note that the algorithm can be improved. Whenever nmgm is called with $u := \tilde{u}_l$ the statement $u := \mathfrak{R}_l(u)$ can be replaced by $u := u - \tilde{f}_l$.

It must be emphasized that the algorithm nmgm needs *no* derivatives $\mathfrak{R}_l' = \partial \mathfrak{R}_l / \partial u_l$. For $l = 0$ only we need a derivative if $u_0 = \mathfrak{R}_0(u_0) + f_0$ is solved by Newton's iteration. Usually, the modified Newton iteration with $\mathfrak{R}_0'(\tilde{u}_0)$, $\tilde{u}_0$ fixed, instead of $\mathfrak{R}_0'(u_0)$ is applied.

6. Numerical example for the nonlinear algorithm. We consider the example of Steuerwalt [10]:

$$u_t = u_{xx} + c_1 \cdot (c_2^4 - u^4) + c_3, \qquad 0 \leqq x \leqq 5, \quad 0 \leqq t \leqq p, \tag{6.1a}$$

$$u_x(0, t) = u_x(5, t) = 0, \qquad 0 \leqq t \leqq p, \tag{6.1b}$$

$$u(x, p) = u(x, 0), \qquad 0 \leqq x \leqq 5, \tag{6.1c}$$

where

$$p = .0085, \quad c_1 = .1958291075\mathrm{E}-8, \quad c_2 = 10,$$

$$c_3(x, t) = \begin{cases} 24\,933.892\,52 \cdot \exp\left(\dfrac{-x^2}{2}\right) & \text{if } 0 \leqq t \leqq .0005, \\ 0 & \text{otherwise.} \end{cases}$$

For the solution of the initial-boundary value problem (6.1a, b) and (2.1) we introduce the step sizes $\Delta x = 5/N$ and $\Delta t = p/M$. Here, Δt denotes the *maximal* time step. According to the definition of c_3, the actual step size is $t_{\nu+1} - t_\nu$, where

$t_{\nu+1} = \min(t_\nu + \Delta t, t_M)$ with $t_M = .0005$ if $t_\nu < .0005$ and $t_M = p$, otherwise. The iteration (5.3) is chosen as Newton's iteration with a derivative evaluated at a fixed approximation $\tilde{u}_0$. Note that $\tilde{u}_0$ has to approximate only $u_0 = \phi_0(0)$, not the actual value u_l, since we have to approximate the solution of (5.3) only for *small* f_0.

The direct application of the multigrid algorithm to the problem (6.1a–c) yields *no* fast convergence. The choice $h_0 = \Delta x_0 = \frac{5}{2}$ as coarsest grid size results in a divergent iteration. For $h_0 = \Delta x_0 = \frac{5}{4}$ we obtain slow convergence. The rates of convergence are about 0.6. The reason of this behavior is the smallness of the time interval $[0, p]$. The smoothing effect is too weak. Note that the constant C of (2.5) tends to infinity as $1/p$ if $p \to 0$. This is also true if Δt is decreased for fixed $h = \Delta x$.

Nevertheless, there is an easy remedy. Extend the coefficients of (6.1a) periodically to the interval $[0, 5p]$ or $[0, 10p]$. Obviously, a periodic solution of (6.1a–c) is a solution of the extended problem, too.

Table 2 contains the rates of convergence for different choices of the finest and coarsest grid sizes $\Delta x_l = h_l$.

TABLE 2
Rates of convergence of the nonlinear algorithm for the example (6.1a–c)

Finest grid size	Time interval: $[0, 5p]$			Time interval: $[0, 10p]$		
	Coarsest grid size Δx_0:			Coarsest grid size Δx_0:		
	$\frac{5}{2}$	$\frac{5}{4}$	$\frac{5}{8}$	$\frac{5}{2}$	$\frac{5}{4}$	$\frac{5}{8}$
$\frac{5}{8}$	.64	.55	—	.53	.41	—
$\frac{5}{16}$	.55	.42	.28	.24	.16	.08
$\frac{5}{32}$	.20	.11	.05	.05	.03	.02

If we apply the loop (5.4) with $h_0 = \Delta x_0 = \frac{5}{2}$ and

$$\tilde{u}_0 = (900.16, 488.39, 371.66)$$

we obtain one iterate $u_l^{(1)}$ per level. The results at $x = 0$ are shown in Table 3. The iteration error of $u_4^{(1)}$ is of the order of the discretization error of u_4.

TABLE 3
Results of (5.4) *at* $x = 0$ *for the example* (6.1a–c) *with* $[0, 10p]$ *as time interval.*

Level l	$\Delta x_l = h_l$	Δt_l	$u_l^{(1)}(0)$	Exact discrete solution at $x = 0$
0	$\frac{5}{2}$	$10p$	—	900.
1	$\frac{5}{4}$	$10p/4$	881.	890.
2	$\frac{5}{8}$	$10p/16$	879.4	888.6
3	$\frac{5}{16}$	$10p/64$	888.20	888.42
4	$\frac{5}{32}$	$10p/256$	888.3688	888.3730

Acknowledgment. The author wishes to thank the referees for their helpful comments.

REFERENCES

[1] J. H. BRAMBLE AND V. THOMÉE, *Semidiscrete least square methods for a parabolic boundary value problem*, Math. Comp., 26 (1972), pp. 633–648.

[2] W. HACKBUSCH, *A numerical method for solving parabolic equations with opposite orientations*, Computing, 20 (1978), pp. 229–240.

[3] ———, *Die schnelle Auflösung der Fredholmschen Integralgleichung zweiter Art*, Beitr. Numer. Math. 9, to appear

[4] ———, *On the regularity of difference schemes*, Ark. Mat., 19 (1981), pp. 71–95.

[5] ———, *An error analysis of the nonlinear multi-grid method of the second kind*, Aplikace Matematiky, 25 (1980), to appear.

[6] ———, *On the fast solution of nonlinear elliptic equations*, Numer. Math., 32 (1979), pp. 83–95.

[7] M. R. OSBORNE, *A note on the numerical solution of a periodic parabolic problem*, Numer. Math. 7 (1965), pp. 155–158.

[8] ———, *The numerical solution of a periodic parabolic problem subject to a nonlinear boundary condition II*, Numer. Math., 12 (1968), pp. 280–287.

[9] R. D. RICHTMEYER AND K. W. MORTON, *Difference Methods for Initial-Value Problems*, Interscience, New York, London, Sydney, 1967.

[10] M. STEURWALT, *The existence, computation, and number of solutions of periodic parabolic problems*, SIAM J. Numer. Anal., 16 (1979), pp. 402–420.

[11] H. TANABE, *Equations of Evolution*, Pitman Press, Bath, 1979.

[12] G. J. TEE, *An application of p-cyclic matrices for solving periodic parabolic problems*, Numer. Math. 6 (1968), pp. 142–159.

[13] L. B. WAHLBIN, *A remark on parabolic smoothing and the finite element method*, SIAM J. Numer. Anal., 17 (1980) pp. 33–38.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0009 $01.00/0

A PROJECTED LAGRANGIAN ALGORITHM FOR NONLINEAR l_1 OPTIMIZATION*

WALTER MURRAY† AND MICHAEL L. OVERTON‡

Abstract. The nonlinear l_1 problem is an unconstrained optimization problem whose objective function is not differentiable everywhere, and hence cannot be solved efficiently using standard techniques for unconstrained optimization. The problem can be transformed into a nonlinearly constrained optimization problem, but it involves many extra variables. We show how to construct a method based on projected Lagrangian methods for constrained optimization which requires successively solving quadratic programs in the same number of variables as that of the original problem. Special Lagrange multiplier estimates are used to form an approximation to the Hessian of the Lagrangian function, which appears in the quadratic program. A special line search algorithm is used to obtain a reduction in the l_1 objective function at each iteration. Under certain conditions the method is locally quadratically convergent if analytical Hessians are used.

Key words. l_1-approximation, l_1-norm minimization, data fitting, absolute deviation curve fitting, nondifferentiable optimization

1. Introduction.

The problem we wish to solve is

$$l_1P: \min_x \{F_1(x) \mid x \in R^n\}$$

where

$$F_1(x) = \sum_{i=1}^{m} |f_i(x)|$$

and the functions $f_i: R^n \to R^1$ are twice continuously differentiable. The function $F_1(x)$ is called the l_1 function, and l_1P is referred to as the l_1 problem. The l_1 problem is an unconstrained optimization problem in which the objective function has discontinuous derivatives and hence it is inappropriate to use a standard unconstrained minimization method to solve it. The problem is equivalent to the following nonlinearly constrained problem in which both the objective and constraint functions are twice continuously differentiable:

$$\text{ELP:} \quad \min \left\{ \sum_{i=1}^{m} u_i \mid x \in R^n, u \in R^m \right\}$$

$$\text{subject to } c_i^{(\sigma)}(x, u) \geqq 0, \quad i = 1, 2, \cdots, m; \quad \sigma = -1, 1,$$

$$\text{where } c_i^{(\sigma)}(x, u) = u_i - \sigma f_i(x).$$

We could solve ELP using a method for the general nonlinear programming problem, but this is very unattractive since m, the number of extra variables, may be large. A method can be derived which exploits the special structure of problem ELP, essentially reducing it back to a problem with n variables. One special feature of ELP is that the l_1 function F_1 is a natural merit function which can be used to measure progress towards the solution of ELP. Such a merit function is not generally available for the nonlinear

* Received by the editors April 28, 1980.

† Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California 94305. The work of this author was supported by the U.S. Department of Energy under contract DE-AS03-76-SF00326, PA No. DE-AT-03-76ER72018; the National Science Foundation under grant MCS-7926009; and the U.S. Army Research Office under contract DAAG29-79-C-0110.

‡ Courant Institute of Mathematical Sciences, New York University, New York, New York, 10012. The work of this author was supported by the Computer Science Department, Stanford University, under U.S. Department of Energy contract DE-AS03-76SF00326, PA No. 30 and National Science Foundation grant MCS78-11985.

programming problem without the introduction of a parameter such as a penalty parameter.

The method we adopt to solve l_1P consists of two parts at each iteration: (1) obtain a direction of search by solving and perhaps modifying a quadratic program based on a projected Lagrangian algorithm for ELP, and (2) take a step along the search direction which reduces the l_1 function. The general approach is similar to that described for the minimax problem by Murray and Overton [24]. The structure of the quadratic program to be solved is, however, considerably different from the minimax case; this is described in full in subsequent sections. We use a special line search algorithm which is closely related to the one used in the minimax case. This is discussed in § 7; the details may be found in [23]. We note that no convexity assumptions are made. We concern ourselves only with local minima.

A number of other algorithms have been proposed for solving the nonlinear l_1 problem. These will be discussed further in § 10, after our algorithm has been described in full. At the time of this writing, no other algorithms related to projected Lagrangian methods using second order information have, to our knowledge, been published. However, Bartels and Conn are currently doing some related work.

It is important to distinguish l_1P from the problem of solving a general nonlinear equality-constrained optimization problem by minimizing an "l_1 penalty function", i.e., $P(x)=F^{(G)}(x)+\rho\sum_{i=1}^{k}|c_i^{(G)}(x)|$, where $\rho>0$ and $F^{(G)}$ and $c_i^{(G)}$, $i=1,\cdots,k$, are respectively the objective function and equality constraints. Clearly a suitable positive constant can be added to $F^{(G)}$ so that $P(x)$ has the form $F_1(x)$ on any given region of interest. However, the important point is that minimizing such a penalty function is a very special case of l_1P since the solution is of no interest unless ρ is large enough that all the constraints are zero at the solution, and hence that only *one* of the terms in the sum, namely $F^{(G)}(x)$, is nonzero at the solution. A related comment can be made for a similar penalty function constructed for inequality constraints, although in this case the connection with l_1P is not so direct. We mention this point because such penalty functions have received a lot of attention recently, e.g., [10], [18]. In this paper, however, we are concerned with solving l_1P without any such restrictions on the solution.

1.1. Notation. All vectors are column vectors, but for convenience we will write (x, u) for $\binom{x}{u}$. Define $(\overset{*}{x}, \overset{*}{u})$ to be a solution of ELP. It follows that $\overset{*}{x}$ is a solution to l_1P and

$$F_1(\overset{*}{x})=\sum_{i=1}^{m}\overset{*}{u}_i.$$

Let $(x^{(k)}, u^{(k)})$ denote the kth approximation to $(\overset{*}{x}, \overset{*}{u})$.

At each iteration of the algorithm $(x^{(k+1)}, u^{(k+1)})$ is obtained by setting

$$x^{(k+1)}=x^{(k)}+\alpha p \quad \text{and} \quad u^{(k+1)}=|f(x^{(k+1)})|$$

where p is the direction of search in R^n and α, a positive scalar, is the steplength, and the absolute value of a vector denotes the vector of the absolute values of the components. Note that this choice of $(x^{(k+1)}, u^{(k+1)})$ immediately guarantees that all the points $\{(x^{(k)}, u^{(k)})\}$ are *feasible* for ELP, i.e., $c_i^{(\sigma)}(x^{(k)})\geqq 0$, $i=1,\cdots,m$, $\sigma=-1,+1$. It also follows that for each i, at least one of the pair of constraints $(c_i^{(-1)}, c_i^{(+1)})$ must have the value zero. We will be interested in the case where the other constraint in the pair is also zero at the solution, i.e., the corresponding function f_i is zero. Therefore, at any point x we define the *active set of functions* as those which we think

will have the value zero at the solution $\hat{x}$, based on the information at x. This set will usually include all functions with the value zero at the point x and may also include some with nonzero values. The exact procedure for selecting the active set at each iteration will be discussed in § 8 and procedures for modifying this choice will be discussed in § 5.

We define $t = t(x)$ to be the number of active functions at x and write the vector of active functions as $\hat{f}(x) \in R^t$. Define $\hat{\Sigma}(x)$ to be the diagonal square matrix of order t whose ith diagonal component is 1 if $\hat{f}_i(x) \geqq 0$ and -1 otherwise. Define $\hat{V}(x)$ to be the $n \times t$ matrix whose columns $\{\hat{v}_i(x)\}$ are the gradients of the active functions. Similarly we define $\bar{f}(x) \in R^{m-t}$ to be the vector of inactive functions at x and define $\bar{\Sigma}(x)$ and $\bar{V}(x)$ to be respectively the $(m-t)\times(m-t)$ diagonal matrix of the signs corresponding to $\bar{f}(x)$ and the $n \times (m-t)$ matrix of gradients of the inactive functions. We also define $\bar{u}$ and $\hat{u}$ to be the subvectors of u corresponding to $\bar{f}$ and $\hat{f}$.

We define the *active constraints* at x to be *both* constraints of each pair corresponding to the active functions plus the one constraint with zero value of each pair corresponding to the inactive functions. We can order the active constraints so that the vector of active constraint values is given by

$$\hat{c}(x) = \begin{bmatrix} \bar{u} - \bar{\Sigma}(x)\bar{f}(x) \\ \hat{u} - \hat{f}(x) \\ \hat{u} + \hat{f}(x) \end{bmatrix}$$

with $\bar{u} = \bar{\Sigma}(x)\bar{f}(x)$, $\hat{u} = \hat{\Sigma}(x)\hat{f}(x)$, by definition of u. Define $\hat{A}(x)$ to be the $(m+n)\times(m+t)$ matrix whose columns $\{\hat{a}_i\}$ are the active constraint gradients. We can order the variables $\{u_i\}$ so that

$$\hat{A}(x) = \begin{bmatrix} -\bar{V}(x)\bar{\Sigma}(x) & -\hat{V}(x) & \hat{V}(x) \\ I_{m-t} & 0 & 0 \\ 0 & I_t & I_t \end{bmatrix}.$$

Here I_s is the identity matrix of order s. Note that $\hat{A}$ has full rank if and only if $\hat{V}$ has full rank.

We define $Y(x)$ to be a matrix with orthonormal columns spanning the range space of $\hat{V}(x)$ and define $Z(x)$ to be a matrix with orthonormal columns spanning the null space of $\hat{V}(x)^T$. Provided $\hat{V}(x)$ has full rank we have that $Y(x)$ has dimension $n \times t$, $Z(x)$ has dimension $n \times (m-t)$, and

$$Y(x)^T Y(x) = I_t, \qquad Z(x)^T Z(x) = I_{n-t},$$
$$Y(x)^T Z(x) = \hat{V}(x)^T Z(x) = 0.$$

Let g be the gradient of the objective function of ELP, i.e., the $(n+m)$-vector:

$$g = \begin{pmatrix} 0 \\ \bar{e} \\ \hat{e} \end{pmatrix}$$

where $\bar{e} \in R^{m-t}$ and $\hat{e} \in R^t$ are vectors of all ones.

The *Lagrangian function* associated with ELP is

$$L(x, u, \lambda) = \bar{e}^T\bar{u} + \hat{e}^T\hat{u} - \lambda^T\hat{c}(x)$$

where $\lambda \in R^{m+t}$ is a vector of *Lagrange multipliers*. The gradient of $L(x, u, \lambda)$ with respect to x is $g - \hat{A}\lambda$. Define W_E to be the Hessian of the Lagrangian function with

respect to (x, u). Then

$$W_E = \begin{bmatrix} W & 0 \\ 0 & 0 \end{bmatrix}$$

where W is the Hessian of $L(x, u, \lambda)$ with respect to x only, i.e.,

$$W(x, \lambda) = \sum_{i=1}^{m+t} \lambda_i \nabla_x^2 \hat{c}_i(x).$$

Define Z_E to be a matrix with orthonormal columns spanning the null space of $\hat{A}^T$, i.e., $\hat{A}^T Z_E = 0$. It follows from the definition of $\hat{A}$ that the first n rows of Z_E can be taken to be Z, the matrix which is orthogonal to $\hat{V}^T$. Thus $Z_E^T W_E Z_E$, the Hessian of $L(x, u, \lambda)$ projected into the null space of $\hat{A}^T$, can also be written as $Z^T W Z$.

We will use p_E to denote a vector in R^{n+m} whose first n components are the direction of search vector p. We write

$$p_E = \begin{pmatrix} p \\ \bar{p} \\ \hat{p} \end{pmatrix}$$

where $\bar{p}$ and $\hat{p}$ correspond to $\bar{u}$ and $\hat{u}$.

Often we will omit the arguments from the various vectors and matrices $\hat{f}$, $\bar{V}$, $\hat{A}$, etc. when it is clear that they are evaluated at $x^{(k)}$. We use the notation $\overset{*}{A}$, $\overset{*}{V}$, $\overset{*}{Z}$ to denote $\hat{A}$, $\hat{V}$, Z evaluated at $(\overset{*}{x}, \overset{*}{u})$ with the active set of functions correctly chosen, i.e., consisting of all those functions with the value zero at $\overset{*}{x}$.

1.2. Necessary and sufficient conditions. In the following, we refer to the constraint qualifications and the necessary and sufficient conditions for a point to be a minimum of the general nonlinear programming problem as defined in [12]. The necessary and sufficient conditions for $(\overset{*}{x}, \overset{*}{u})$ to be a local minimum of problem ELP, and therefore for $\overset{*}{x}$ to be a local minimum of problem l_1P, are simplifications of these general conditions. It can be shown that the first-order constraint qualification always holds for ELP. The conditions therefore reduce to the following:

First-order necessary condition. If $(\overset{*}{x}, \overset{*}{u})$ is a local minimum of ELP then there exists a vector of Lagrange multipliers $\overset{*}{\lambda} \in R^{m+t}$ such that

$$g - \overset{*}{A}\overset{*}{\lambda} = 0 \quad \text{and} \quad \overset{*}{\lambda} \geqq 0.$$

Second-order necessary condition. If $(\overset{*}{x}, \overset{*}{u})$ is a local minimum of ELP and the second-order constraint qualification holds, then $\overset{*}{Z}^T W(\overset{*}{x}, \overset{*}{\lambda})\overset{*}{Z}$, the projected Hessian of the Lagrangian function, is positive semi-definite.

Sufficient condition. If the first-order necessary condition holds at $(\overset{*}{x}, \overset{*}{u})$, the Lagrange multipliers are all strictly positive, i.e., $\overset{*}{\lambda} > 0$, and $\overset{*}{Z}^T W(\overset{*}{x}, \overset{*}{\lambda})\overset{*}{Z}$ is positive definite, then $(\overset{*}{x}, \overset{*}{u})$ is a strong local minimum of ELP. Thus in terms of l_1P, $F_1(\overset{*}{x}) < F_1(x)$ for all x such that $|x - \overset{*}{x}| < \delta$, for some $\delta > 0$.

In the case where all the $\{f_i\}$ are linear it is well known that a solution must exist with n active functions at $\overset{*}{x}$. Then, normally the matrix $\overset{*}{Z}$ is null, which implies the second-order conditions are also null. The nonlinear problem, however, can have a unique solution with anything from zero to n functions active at $\overset{*}{x}$.

Alternative derivations of optimality conditions for l_1P are given in [7], [14].

2. Use of the equivalent problem ELP. At every iteration we wish the search direction p to be a descent direction for F_1, i.e.,

$$F_1'(x^{(k)}, p) < 0,$$

where $F_1'(x^{(k)}, p)$ is the directional derivative

$$\lim_{h \to 0^+} \frac{1}{h}(F_1(x^{(k)} + hp) - F_1(x^{(k)})).$$

It is easy to see that $F_1'(x^{(k)}, p)$ is also given by

$$\sum_{i|f_i \neq 0} \frac{f_i}{|f_i|} v_i^T p + \sum_{i|f_i = 0} |v_i^T p| \tag{2.1}$$

where $v_i = \nabla f_i$. We have the following:

THEOREM 1. *If p_E is a first-order feasible descent direction for* ELP, *i.e.,*

$$g^T p_E < 0 \tag{2.2}$$

and

$$\nabla c_i^{(\sigma)T} p_E \geqq 0 \text{ for all } i, \sigma \text{ such that } c_i^{(\sigma)} = 0, \tag{2.3}$$

then p is a descent direction for F_1 and hence a sufficiently small step along it must result in a reduction in F_1.

Proof. Suppose for the moment that the active set consists of those and only those functions which are zero at $x^{(k)}$, so that we can use the notation developed for this. We then have $\hat{A}^T p_E \geqq 0$, and hence

$$\begin{aligned} -\bar{\Sigma}\bar{V}^T p + \bar{p} &\geqq 0, \\ -\hat{V}^T p + \hat{p} &\geqq 0, \\ \hat{V}^T p + \hat{p} &\geqq 0. \end{aligned}$$

It follows from (2.1) and (2.2) that

$$F_1'(x^{(k)}, p) \leqq \bar{e}^T \bar{p} + \hat{e}^T \hat{p} < 0. \qquad \square$$

It is possible for p_E to be a first-order feasible direction without being a feasible direction for ELP. This causes no difficulty since $u^{(k+1)}$ is set to $|f(x^{(k+1)})|$ and hence it is always possible to obtain a lower feasible point for ELP if (2.2) and (2.3) hold, by reducing F_1 along p.

A second desirable property for p arises from considering the active set of functions, i.e., those we expect to have the value zero at $\overset{*}{x}$. We wish to choose p so that the first-order change in these functions predicts that they will all have the value zero at $x^{(k)} + p$. An equivalent condition is:

$$\hat{V}^T p = -\hat{f}. \tag{2.4}$$

This condition is implied by the following condition on the $(n+m)$-vector p_E:

$$\hat{A}^T p_E = -\hat{c}. \tag{2.5}$$

Strictly speaking, (2.5) is a stronger condition than the pair of conditions (2.4) and

$$\hat{A}^T p_E \geqq -\hat{c}. \tag{2.6}$$

However, since the only difference is that the variables $\{u_i\}$ are required to be on their bounds and it will become evident later that this does not affect the choice of search direction, for simplicity we will require that p_E satisfy (2.5).

Thus we see that one view of ELP is as a device to obtain a search direction p along which F_1 can be reduced in the line search. We emphasize again that we wish (2.2) and (2.3) to hold so that p is a descent direction for F_1, and that the active set nature of the algorithm indicates that (2.4) and hence (2.5) should also hold.

3. Derivation and solution of QP subproblem. The solution of ELP is at a minimum of the Lagrangian function in the null space of the active constraint Jacobian. The usual method for solving a general linearly constrained problem is to approximate the objective function by a quadratic function and then determine the search direction by solving some appropriate quadratic program (QP). Consider therefore the quadratic program:

$$\text{QP1:} \quad \min_{p_E} \tfrac{1}{2} p_E^T W_E(x^{(k)}, \lambda^{(k)}) p_E + g^T p_E$$

$$\text{subject to } \hat{A}^T p_E = -\hat{c},$$

$$\text{where } \lambda^{(k)} \text{ is an estimate of } \overset{*}{\lambda}.$$

The constraints of QP1 are equivalent to (rearranging equations):

$$\begin{aligned} -\bar{\Sigma}\bar{V}^T p + \bar{p} &= 0, \\ -\hat{\Sigma}\hat{V}^T p + \hat{p} &= 0, \\ \hat{\Sigma}\hat{V}^T p + \hat{p} &= -2\hat{\Sigma}\hat{f}. \end{aligned} \tag{3.1}$$

The last two equations imply that

$$\hat{p} = \hat{\Sigma}\hat{V}^T p = -\hat{\Sigma}\hat{f}. \tag{3.2}$$

Since $g^T p_E = \bar{e}^T \bar{p} + \hat{e}^T \hat{p}$ it follows that p can be obtained by solving the following QP in only n variables:

$$\text{QP2:} \quad \min_{p} p^T W(x^{(k)}, \lambda^{(k)}) p + \bar{e}^T \bar{\Sigma} \bar{V}^T p$$

$$\text{subject to } \hat{V}^T p = -\hat{f}.$$

In order to solve QP2 we introduce the matrices Y and Z defined in § 1.1. These may be determined from the QR factorization of $\hat{V}$:

$$\hat{V} = Q\begin{bmatrix} R \\ 0 \end{bmatrix} = [Y \quad Z]\begin{bmatrix} R \\ 0 \end{bmatrix}$$

where R is an upper triangular matrix of order t. If $\hat{V}$ has full rank and $Z^T W Z$ is positive definite, then the unique solution of QP2 may be expressed as the sum of

two orthogonal components

$$p = Yp_Y + Zp_Z \tag{3.3}$$

where $p_Y \in R^t$ and $p_Z \in R^{n-t}$. We have

$$\hat{V}^T p = R^T p_Y = -\hat{f}, \tag{3.4}$$

and p_Y is determined entirely by the constraints of QP1. The vector p_Z is given by the solution of

$$(Z^T W Z)p_Z = -Z^T(\bar{V}\bar{\Sigma}\bar{e} + WYp_Y). \tag{3.5}$$

We shall also wish to refer to the related QP with homogeneous constraints:

QP3: $\min_p \frac{1}{2}p^T W p + \bar{e}^T \bar{\Sigma} \bar{V}^T p$

subject to $\hat{V}^T p = 0$.

The solution of this is given by $p = Zq_z$, where

$$(Z^T W Z)q_z = -Z^T \bar{V}\bar{\Sigma}\bar{e}. \tag{3.6}$$

At every iteration of our algorithm an attempt is made to set the search direction p to the solution of QP2, but for various reasons this may be inadequate. In subsequent sections we explain what action is taken in these circumstances.

4. Lagrange multiplier estimates. Let us first suppose that $x^{(k)}$ is a minimum on the manifold defined by the current active set. Then for some λ we have

$$\hat{A}\lambda = g.$$

If we write $\lambda = (\bar{\lambda}, \hat{\lambda}^+, \hat{\lambda}^-)$, this is equivalent to

$$-\bar{V}\bar{\Sigma}\bar{\lambda} - \hat{V}\hat{\lambda}^+ + \hat{V}\hat{\lambda}^- = 0, \tag{4.1}$$

$$\bar{\lambda} = \bar{e}, \tag{4.2}$$

$$\hat{\lambda}^+ + \hat{\lambda}^- = \hat{e}. \tag{4.3}$$

Let us therefore define

$$\pi = \hat{\lambda}^+ - \hat{\lambda}^-. \tag{4.4}$$

Equations (4.1) and (4.2) then reduce to

$$\hat{V}\pi = -\bar{V}\bar{\Sigma}\bar{e}. \tag{4.5}$$

It follows from (4.3) and (4.4) that the first order necessary condition $\overset{*}{\lambda} \geqq 0$ is equivalent to

$$|\overset{*}{\pi}| \leqq 1,$$

where $\overset{*}{\pi}$ is π at $\overset{*}{x}$. The vector π here plays the same role as the vector w in the linear case described by Bartels, Conn and Sinclair [6].

The question we face in this section is how to define the vector of Lagrange multiplier estimates at any point $x^{(k)}$, dropping the assumption that it is the minimum on the manifold. Multiplier estimates are needed to define the matrix W and to determine whether constraints should be deleted from the active set. It is better to use new information obtained at the current point $x^{(k)}$ rather than use the multipliers of

the QP solved at the previous iteration. Such an estimate should in some sense approximately satisfy the overdetermined system based on the first order necessary conditions:

$$\hat{A}\lambda \approx g. \tag{4.6}$$

Clearly it makes no sense to delete an active constraint corresponding to an inactive function, since the corresponding variable $\bar{u}_i$ will be reduced at the end of the iteration to make the constraint active again. Similarly, only one active constraint of the pair corresponding to an active function should be considered for deletion. These facts combined with the fact that the search direction is being determined for a QP involving only n variables indicate that a special estimate taking into account the structure of ELP should be used, as opposed to the least squares solution of (4.6).

λ_C: *The special estimate for* ELP. The special estimate is required to satisfy exactly those equations in (4.6) which are exactly the same as the equations holding at the minimum on the manifold defined by the active set. Thus we define λ_C to be the least squares solution to the approximate equation (4.1), subject to the constraint that (4.2) and (4.3) hold exactly. Equivalently we can define π_C as the solution of the least squares problem

$$\min \|\hat{V}\pi + \bar{V}\bar{\Sigma}\bar{e}\|_2^2. \tag{4.7}$$

It is then not necessary to explicitly form λ_C, since checking whether a component of π is greater than one in modulus is equivalent to checking whether a component of $\hat{\lambda}^+$ or $\hat{\lambda}^-$ is negative. Furthermore, using λ_C to define W results in

$$\begin{aligned} W &= \sum_{i=1}^{m-t} (\bar{\lambda}_C)_i \bar{\sigma}_i \nabla^2 \bar{f}_i + \sum_{i=1}^{t} (\hat{\lambda}_C^+)_i \nabla^2 \hat{f}_i + \sum_{i=1}^{t} -(\hat{\lambda}_C^-)_i \nabla^2 \hat{f}_i \\ &= \sum_{i=1}^{m-t} \bar{\sigma}_i \nabla^2 \bar{f}_i + \sum_{i=1}^{t} (\pi_C)_i \nabla^2 \hat{f}_i \end{aligned}$$

where $\bar{\sigma}_i$ is the ith diagonal element of $\bar{\Sigma}$. Thus π_C may also be used to define W directly.

The vector π_C can be computed directly from the QR factorization of $\hat{V}$ which we introduced in the last section to solve QP2. The estimate is a first-order multiplier estimate in the sense defined in [24].

Because at every iteration computing the search direction involves only the first n variables of ELP, the multiplier estimate which is relevant to predicting whether the steepest descent step in a subspace of R^n will be first-order feasible is λ_C, *not* the least squares solution to (4.6). Let us suppose that $(\pi_C)_j > 1$ and that the constraint $\hat{u}_j \geqq -\hat{f}_j(x)$ is to be deleted from the active constraint set (i.e., $\hat{f}_j$ is to be deleted from the active function set). Define $\tilde{V}$ as $\hat{V}$ with $\hat{v}_j$ deleted, and $\tilde{Z}$ by

$$\tilde{V}^T\tilde{Z} = 0, \quad \tilde{Z}^T\tilde{Z} = I_{n-t+1}, \quad \tilde{Z} = [Z \quad z]. \tag{4.8}$$

Now consider the gradient of the linear term in QP2, i.e., $\bar{V}\bar{\Sigma}\bar{e}$. Since $\hat{v}_j$ is being deleted from the active set it should be included in the gradient of an objective function to be minimized in the null space of $\tilde{V}^T$. Therefore define the steepest descent step in the new null space to be

$$\tilde{Z}s_{\tilde{Z}} = -\tilde{Z}\tilde{Z}^T(\bar{V}\bar{\Sigma}\bar{e} + \hat{v}_j).$$

(Here $\hat{v}_j$ has a positive sign since $(\pi_C)_j > 1$.) We then have the following result relating the estimate π_C to the first-order feasibility of $\tilde{Z}s_{\tilde{Z}}$.

THEOREM 2. *Assume $\hat{V}$ has full rank. If $(\pi_C)_j > 1$, then*

$$\hat{v}_j^T \tilde{Z} s_{\tilde{Z}} > 0.$$

Proof. We have the least squares characterization:

$$\hat{V}\pi_C = -(I - ZZ^T)\bar{V}\bar{\Sigma}\bar{e}.$$

Thus

$$(z^T\hat{v}_j)(\pi_C)_j = -z^T\bar{V}\bar{\Sigma}\bar{e}.$$

By definition of $\tilde{Z}s_{\tilde{Z}}$ we have:

$$\begin{aligned}
\hat{v}_j^T \tilde{Z} s_{\tilde{Z}} &= \hat{v}_j^T [Z \quad z] s_{\tilde{Z}} \\
&= -[0 \quad \hat{v}_j^T z] \begin{bmatrix} Z^T \\ z^T \end{bmatrix} (\bar{V}\bar{\Sigma}\bar{e} + \hat{v}_j) \\
&= -(\hat{v}_j^T z)(z^T\bar{V}\bar{\Sigma}\bar{e} + z^T\hat{v}_j) \\
&= -(\hat{v}_j^T z)^2(1 - (\pi_C)_j) > 0.
\end{aligned}$$

(The fact that $\hat{V}$ has full rank implies that $\hat{v}_j^T\tilde{Z} \neq 0$ and hence $\hat{v}_j^T z \neq 0$.) □

It follows from Theorem 2 that setting $p = \tilde{Z}s_{\tilde{Z}}$ defines a vector p_E which is first-order feasible with respect to the deleted constraint $\hat{u}_j \geqq -\hat{f}_j(x)$, since p_E is first-order feasible with respect to the retained active constraint $\hat{u}_j \geqq \hat{f}_j(x)$. Note that $(\pi_C)_j$ being >1 is equivalent to $(\hat{\lambda}_C^-)_j$, the multiplier corresponding to the deleted constraint, being negative. Clearly if $(\pi_C)_j < -1$ and hence $(\hat{\lambda}_C^+)_j < 0$ then the deleted constraint would be $\hat{u}_j \geqq \hat{f}_j(x)$, the steepest descent step would be $-\tilde{Z}\tilde{Z}^T(\bar{V}\bar{\Sigma}\bar{e} - \hat{v}_j)$, and this would have a negative inner product with $\hat{v}_j$.

λ_L: *The least squares estimate in the larger space.* Define λ_L to be the least squares solution to (4.6). It is worth emphasizing that although the signs of the components of λ_L determine the feasibility of steepest descent steps for ELP in the null space of $\hat{A}^T$ with a row deleted (see [17] and [24]), the estimate λ_L is not relevant to the algorithm we describe for solving the l_1 problem. This is because (unlike in the minimax case of [24]), it is the range and null spaces of $\hat{V}$, not $\hat{A}$, which determine the search direction at each iteration. Unlike the minimax case, λ_L and λ_C are not scalar multipliers of each other. It will often be the case that a component of λ_C is negative while the corresponding component of λ_L is positive, indicating that if the corresponding constraint is deleted, the steepest descent step (with respect to ELP) in the new null space in the $(n+m)$-dimensional space will not be feasible with respect to the deleted constraint, while the steepest descent step (with respect to QP2) in the new null space in the n-dimensional space will be first-order feasible. The converse is much less likely to happen, i.e., where a component of λ_C is positive while the corresponding component of λ_L is negative, but it is possible to construct such an example.

Quite apart from its deficiencies, the estimate λ_L is more expensive to compute than λ_C, since it involves the factorization of a larger matrix. If m is large compared to n, then the additional effort may be prohibitively high. There is a slight simplification, however: since the last t rows of $\hat{A}$ have full rank and are orthogonal to the others, it follows that the last t equations in (4.6) must hold exactly and hence a least squares problem of slightly reduced dimension can be solved.

Clearly, it is undesirable to compute λ_L and we will not discuss this estimate any further.

μ_W: *A second-order estimate.* A second-order multiplier estimate can be defined as the solution to the consistent set of overdetermined equations

$$\hat{A}\mu_W = g + W_E p_E.$$

The fact that the system is consistent implies that μ_W can be obtained from solving

$$\hat{V}\pi_W = -\bar{V}\bar{\Sigma}\bar{e} - Wp$$

and there is no concern about solving the larger system. A negative component of μ_W does not guarantee that either a steepest descent or Newton step will be first-order feasible with respect to the deleted constraint.

Using the estimate π_C to define W. Both the estimates π_C and π_W will be used to decide when to delete functions from the active set. Since a function corresponding to $|(\pi_C)_i| > 1$ will not necessarily be deleted from the active set, we note here that we use π_C to define W as follows:

$$W = \sum_{i=1}^{t} \pi_i' \nabla^2 \hat{f}_i + \sum_{i=1}^{m-t} \bar{\sigma}_i \nabla^2 \bar{f}_i \qquad \text{where} \tag{4.9}$$

$$\pi_i' = \begin{cases} -1 & \text{if } (\pi_C)_i < -1, \\ 1 & \text{if } (\pi_C)_i > 1, \\ (\pi_C)_i & \text{otherwise.} \end{cases}$$

5. Properties of solution of QP subproblem. In this section we examine the properties of the solution to QP2. Initially, we assume that all functions with zero value are included in the active set, and that $\hat{V}$ has full rank and $Z^T WZ$ is positive definite so that the solution p given by (3.3), (3.4) and (3.5) is unique. The corresponding solution to QP1 is p_E, where $\bar{p}$ and $\hat{p}$ are given by (3.1) and (3.2). We would like p_E to satisfy (2.2), (2.3) and (2.5). Clearly the constraints of QP1 ensure that (2.3) and (2.5) hold. Thus the only question is whether p_E is a descent direction for ELP, i.e., whether (2.2) holds. If all the active functions have the value zero, then the following applies:

THEOREM 3. *Suppose that $\hat{f} = 0$, $\hat{V}$ has full rank and $Z^T WZ$ is positive definite. Then p_E, the solution to* QP1, *is a descent direction for* ELP *provided it is not zero.*

Proof. We have $g^T p_E = \bar{e}^T \bar{p} + \hat{e}^T \hat{p} = \bar{e}^T \bar{\Sigma} \bar{V}^T p$ by (3.1) and (3.2). Since $\hat{f} = 0$ we have $p_Y = 0$ and $p = Zp_Z$ as defined by (3.5). Thus

$$g^T p_E = \bar{e}^T \bar{\Sigma} \bar{V}^T Z p_Z = -p_Z^T (Z^T WZ) p_Z$$

by (3.5). Since $Z^T WZ$ is positive definite, $g^T p_E$ must be negative if $p_Z \neq 0$, i.e., $p \neq 0$. □

If $p = 0$, then by (3.5) $Z^T \bar{V} \bar{\Sigma} \bar{e} = 0$ and $x^{(k)}$ is a minimum on the manifold defined by the current active constraints, and hence (4.6) is a consistent set of equations with $\lambda_C = \lambda_L$. Thus either $x^{(k)}$ is the required solution, or one of the components of λ_C is negative or zero, i.e., $|(\pi_C)_j| \geqq 1$ for some j.

If $p = 0$ and at least one of the multipliers is negative, i.e., $|(\pi_C)_j| > 1$, then it is necessary to delete a corresponding function from the active set to obtain a descent direction. If $p = 0$, and the component of π_C with largest modulus is ± 1, corresponding to a zero multiplier, then $x^{(k)}$ may or may not be a solution. We refer to Gill and Murray [17] for the treatment of zero multipliers.

5.1. Nonzero active functions. In practice it will rarely be the case that $\hat{f} = 0$, so we now drop this assumption. If we were sufficiently restrictive in the definition of the active set (e.g., no active functions) we could force this condition to be true, but it is

important for the efficiency of the algorithm not to be too restrictive in the choice of active set. It could then happen that p_E is an ascent direction for ELP. It is now necessary to introduce some further notation. We denote the components of p_E which correspond to the orthogonal n-vectors Yp_Y and Zp_Z by p_{EY} and p_{EZ} respectively. More specifically, we define:

$$p_{EY} = \begin{bmatrix} Yp_Y \\ \bar{\Sigma}\bar{V}^T Yp_Y \\ -\hat{\Sigma}\hat{f} \end{bmatrix} \quad \text{and} \quad p_{EZ} = \begin{bmatrix} Zp_Z \\ \bar{\Sigma}\bar{V}^T Zp_Z \\ 0 \end{bmatrix}. \tag{5.1}$$

We have $p_E = p_{EY} + p_{EZ}$.

Without the assumption that $\hat{f} = 0$, both p_{EY} and p_{EZ} could be ascent directions for ELP. If the component p_{EZ} is an ascent direction it can be replaced by

$$\begin{bmatrix} Zq_Z \\ \bar{\Sigma}\bar{V}^T Zq_Z \\ 0 \end{bmatrix}$$

where Zq_Z is the solution to QP3 and q_Z is given by (3.6). It is clear from Theorem 3 that this direction is descent for ELP unless it is zero. The motivation here is that if x is too far away from the solution $\overset{*}{x}$, the inhomogeneous constraints of QP2 may not approximate the active functions at $\overset{*}{x}$ and may impede the search direction, but minimizing the quadratic form subject to $\hat{V}^T p = 0$ will give an adequate reduction in the l_1 function. We prefer to solve QP2 if possible since this is ultimately required for quadratic convergence.

If the component p_{EY} is an ascent direction the following theorem shows that a constraint can always be found with a negative multiplier estimate. The interpretation of this situation is that too many functions have been selected to be active and are being forced to be approximately zero at $x^{(k)} + p$, thus forcing the inactive functions to increase in modulus more than the active ones are decreasing.

THEOREM 4. *Assume $\hat{V}$ has full rank and let p_{EY} be defined by* (5.1). *If $g^T p_{EY} > 0$, then for some j either $\hat{f}_j > 0$ and $(\pi_C)_j > 1$, or $\hat{f}_j < 0$ and $(\pi_C)_j < -1$.*

Proof. It follows from $g^T p_{EY} > 0$ that

$$\bar{e}^T \bar{\Sigma}\bar{V}^T Yp_Y - \hat{e}^T \hat{\Sigma}\hat{f} > 0.$$

A characterization of the solution of the least squares problem (4.7) is

$$\hat{V}\pi_C = -YY^T \bar{V}\bar{\Sigma}\bar{e}.$$

Multiplying both sides by Yp_Y, we have from (3.4) that

$$\pi_C^T \hat{f} = \bar{e}\bar{\Sigma}\bar{V}^T Yp_Y$$

and hence

$$\pi_C^T \hat{f} > \hat{e}^T \hat{\Sigma}\hat{f}.$$

Therefore for some j, $(\pi_C)_j \hat{f}_j > |\hat{f}_j|$ and the result follows. □

It also follows from the above that if $g^T p_{EY} = 0$, then either $\hat{f} = 0$ (covered by Theorem 3) or, for some j, $\operatorname{sgn}(\hat{f}_j)(\pi_C)_j \geqq 1$ (a multiplier estimate is zero or negative). It is worth noting that Theorem 4 does not hold in general if other multiplier estimates such as λ_L and μ_W are substituted for λ_C.

It follows from Theorem 4 in conjunction with Theorem 2 that if p_{EY} is an ascent direction we can delete the active constraint corresponding to a negative component of λ_C to obtain a first-order feasible direction. As in [24], we will not actually take the steepest descent direction $\tilde{Z}s_{\tilde{Z}}$ in the new null space. Instead, we will first try computing the Newton step $\tilde{Z}q_{\tilde{Z}}$, defined by (4.8) and

$$(\tilde{Z}^T W \tilde{Z}) q_{\tilde{Z}} = -\tilde{Z}^T (\bar{V}\bar{\Sigma}\bar{e} + \operatorname{sgn}(\pi_C)_j \hat{v}_j)$$

where the jth active function was deleted. If $\operatorname{sgn}(\pi_C)_j \hat{v}_j^T \tilde{Z} q_{\tilde{Z}} > 0$, i.e., $\tilde{Z}q_{\tilde{Z}}$ is first-order feasible in the sense of Theorem 2, then we take this as our search direction. Otherwise we use $\tilde{Z}r_{\tilde{Z}}$, where

$$r_{\tilde{Z}} = \begin{bmatrix} -(Z^T W Z)^{-1} b \\ -z^T b \end{bmatrix},$$

$b = \bar{V}\bar{\Sigma}\bar{e} + \operatorname{sgn}(\pi_c)_j \hat{v}_j$ and z is given by (4.8). The proof that $\tilde{Z}r_{\tilde{Z}}$ is a descent direction and is first-order feasible combines the methods of proof of Theorem 5 of [24] and Theorem 2 above in a straightforward way, so we do not present it here. The motivation for the use of $\tilde{Z}r_{\tilde{Z}}$ is that it combines the Newton step in the null space of the previous active function Jacobian with the steepest descent step in the new direction made available by moving off an active constraint. The steepest descent component may be necessary to ensure the first-order feasibility, but we use a Newton component where possible since it is likely to give a larger reduction in the l_1 function.

The above explains how the direction of search is selected when a constraint (or equivalently a function) is deleted from the active set in order to obtain a descent direction when $g^T p_{EY} > 0$. However, it also applies when we wish to delete a constraint because it corresponds to a negative multiplier estimate which we consider reliable. It is well known in the context of constrained optimization that it is inefficient to accurately approximate a minimum on the manifold corresponding to the current active set before moving off constraints with negative multiplier estimates. Section 9 gives the details of the conditions under which such constraints are deleted from the active set.

5.2. Avoiding a rank-deficient Jacobian or an indefinite projected Hessian. If $\hat{V}$ is rank-deficient then the constraints of QP2 may not be compatible. Clearly it is desirable to restrict the number of functions in the active set so that $\hat{V}$ has full rank. The technique we use for doing this is identical to that described in [24] for the minimax problem (except that instead of ordering the potential columns of $\hat{A}$ by the size of $\{c_j\}$ we order the potential columns of $\hat{V}$ by the size of $|f_j|$).

If $Z^T W Z$ is not positive definite then the vector p given by (3.3), (3.4) and (3.5) is a step to a stationary point which is not a minimum of the quadratic form on the linear manifold defined by the constraints and may even be a maximum. An alternative direction of search which is satisfactory uses the modified Cholesky factorization of [16]. This computes the Cholesky factorization of $Z^T W Z + E$, where E is a nonnegative diagonal matrix large enough to make $Z^T W Z + E$ numerically positive definite. The modified Newton direction in the null space of $\hat{A}^T$ is then taken as Zq_Z where q_Z is given by $(Z^T W Z + E) q_Z = -Z^T \bar{V}\bar{\Sigma}\bar{e}$. It is easy to show that this vector is a descent direction for $l_1 P$ provided it is not zero. For further discussion of alternative directions of search and the possibility of encountering a constrained saddle point see [24].

6. Finite difference and quasi-Newton approximations to the Hessian. In practice we may wish to use a finite difference or quasi-Newton approximation instead of the analytical Hessians $\{\nabla^2 f_i\}$. In the former case we approximate WZ by differencing the gradients $\{\nabla f_i\}$ along the columns of Z. It is not necessary to approximate W itself,

although it is necessary to approximate $W(Yp_Y)$ if we wish to compute π_W. One point to note here is that for the l_1 problem W involves the Hessians of *all* the m functions $\{f_i\}$, whereas in the minimax problem of [24] W involves only the Hessians of the t active terms. Since for many applications (particularly arising from data approximation) m is much larger than t this means that a finite difference approximation may be considerably more expensive than a quasi-Newton approach for an l_1 problem, while a finite difference approach may be more efficient for a comparable minimax problem.

When a quasi-Newton method is used, two approaches are possible: approximating the full matrix W (see for example Powell [27]) or the projected matrix Z^TWZ (see for example Murray and Wright [25]).

7. Determining the steplength. We use a special steplength algorithm tailored to the l_1 problem to obtain the steplength α at each iteration. This algorithm is presented in [23]. The initial step α_0 is set to either one, or the shortest estimated step to a zero of an inactive function, if this is less than one. Thus

$$\alpha_0 = \min\{1, \alpha_0'\},$$

where

$$\alpha_0' = \min\left\{\frac{-\bar{f}_i}{\bar{v}_i^T p}\middle| \bar{v}_i^T p \neq 0 \text{ and } \frac{\bar{f}_i}{\bar{v}_i^T p} < 0\right\}.$$

8. Selecting the active set. Clearly it is necessary to be able to make a reasonable choice of the active set at each iteration. If functions are required to have very small magnitude to be included in the active set then the iterates will follow the constraint boundaries very slowly and convergence will be slow. However if too many functions are selected as active the directions of search may be poor. The active set strategy which has been most successful in our numerical experiments is the same as that described for problem $l_\infty P$ in [24], with some slight modifications, as follows. We define the scaled function values by $\bar{c}_i = (m|f_i|)/F_1$. Since there may be no active functions the first decision is whether to include the smallest one (in absolute value) in the active set. This decision is made in the same way as the decision of whether to include a second active constraint in the minimax case, replacing the gradient v_1 by the gradient of the l_1 function when no functions are active, i.e., $\bar{V}\bar{\Sigma}\bar{e}$.

9. Summary of the algorithm. The kth iteration of the algorithm using analytical Hessians or a finite difference approximation to WZ is summarized as follows. The case where W or Z^TWZ is approximated by a quasi-Newton method differs only in that second-order multiplier estimates are not used. Some details of the algorithm may be changed with more numerical experience; for example the multiplier test in Step 10 is somewhat arbitrary. All vector and matrix functions are to be evaluated at the current point x.

ALGORITHM.

1. [Select active set.] Form $\hat{f}$, $\bar{f}$, $\hat{V}$, $\bar{V}$, $\bar{\Sigma}$. Let $h = \bar{V}\bar{\Sigma}\bar{e}$.
2. [QR factorization.] Factorize $\hat{V} = [Y \quad Z]\begin{bmatrix} R \\ 0 \end{bmatrix}$.
3. [First order multiplier estimate and one component of search direction.] Solve $R\pi_C = -Y^Th$ and $R^Tp_Y = -\hat{f}$.
4. [Projected Hessian.] Form Z^TWZ, where W is given by (4.9).
5. [Modified Cholesky factorization.] Factorize $Z^TWZ + E = LDL^T$.
6. [Termination criteria.] If $\|\hat{f}\|$ and $\|Z^Th\|$ are greater than prescribed tolerances then go to Step 7.

Otherwise: – if $|\pi_C| < 1$ and $E = 0$ ($Z^T W Z$ numerically positive definite) then STOP—x satisfies convergence criteria.
– if $\max |(\pi_C)_i| > 1$ then go to Step 13.
– if $\max |(\pi_C)_i| = 1$ then optionally try zero multiplier procedure (see [17]).
– if $E \neq 0$ then optionally try saddle point procedure (see [24] and [16]).

7. [Other component of search direction.] If $E \neq 0$ then go to Step 12. Otherwise solve $(LDL^T)p_Z = -Z^T(h + WYp_Y)$.
8. [Second-order multiplier estimates.] Solve $R\pi_W = -Y^T(h + W(Yp_Y + Zp_Z))$.
9. [Check whether too many functions active.] If $h^T Y p_Y - \hat{e}^T \hat{\Sigma} \hat{f} > 0$ (i.e., $g^T p_{EY} > 0$) then go to Step 13.
10. [Check multiplier estimates.] Let $|(\pi_C)_j| = \max |(\pi_C)_i|$ and let $\rho_1 = \max |(\pi_C)_j|$, $|(\pi_W)_j|) - 1$ and $\rho_2 = (\|Z^T h\|_2 + \|\hat{f}\|_2)/(1 + F_1/m)$. If $\rho_2 < \min(1, \rho_1)$ and $2|(\pi_C)_j - (\pi_W)_j| < \rho_1$ then go to Step 13. (This condition ensures that the accuracy with which the minimum on the manifold has been approximated and the accuracy of the multiplier estimates are sufficiently high compared to the uncertainty of the signs of the multipliers.)
11. [Direction of search.] If $h^T Z p_Z < 0$ (i.e., $g^T p_{EZ} < 0$) then set $p = Yp_Y + Zp_Z$ and go to Step 14.
12. [Alternative direction of search.] Solve $(LDL^T)q_Z = -Z^T h$ and set $p = Zq_Z$. Go to Step 14.
13. [Delete a term from the active set.] Delete $\hat{f}_j$ from the active set where $|(\pi_C)_j| = \max |(\pi_C)_i|$, and compute the direction of search p as explained in § 5.1.
14. [Line search.] Replace x by $x + \alpha p$, where α is obtained from the steplength algorithm discussed in § 7. That p is guaranteed to be a descent direction for $l_1 P$ follows from Theorem 1 and the discussion in § 5.

10. Relationships to other algorithms. We begin this section by discussing algorithms for problem $l_1 P$ when the functions $\{f_i\}$ are linear. The equivalent problem ELP is then a linear programming problem, although it is not in standard form. The connection between the linear l_1 problem and linear programming (although using a different formulation from ELP) was observed by Charnes, Cooper and Ferguson [8]. Since then a large number of different methods have been proposed for solving the linear l_1 problem by various linear programming formulations. References to many such methods may be found in [2], [4], [5], [31]. Probably the most widely used method is that of Barrodale and Roberts [4], which solves a variation of ELP put in standard form by a primal simplex method taking account of the special structure. An alternative approach taken by Claerbout and Muir [9] and Bartels, Conn and Sinclair [6] is to solve the problem directly by minimizing the piecewise linear function. However, it is possible to think of these methods as linear programming methods applied to ELP, by considering the connection between the vector w in Bartels, Conn and Sinclair (which corresponds to π in our nonlinear algorithm) and the simplex (Lagrange) multipliers of ELP, just as we do for the nonlinear problem. This is the approach we prefer since it retains the familiar linear programming terminology while avoiding transforming ELP to standard form.

A completely different approach to the linear l_1 problem is to solve a sequence of weighted least squares problems as the Lawson algorithm (see [28]) does for the linear

l_∞ problem. This was suggested by Schlossmacher [30] but Gallant and Gerig [15] show that this method can be unstable.

It is somewhat surprising, given the number of linear l_1 algorithms, that there has been comparatively little work done on the nonlinear l_1 problem. Osborne and Watson [26] solve the nonlinear l_1 problem by solving a sequence of linear l_1 problems. The solution to each linear l_1 problem, obtained by a linear programming technique, is used as a search direction along which the minimum of F_1 is found by an exact line search. We are aware of only two published methods for the nonlinear l_1 problem which use second-order information, both of which appeared only recently. El-Attar, Vidyasagar and Dutta [11] suggest a method related to the penalty method for nonlinear programming in which a sequence of increasingly ill-conditioned unconstrained optimization problems are solved. McLean and Watson [22] propose both a first-order Levenberg-type of method similar to those of [1], [21] for the minimax problem, and a method which uses second-order information. The latter is a two-stage method similar to that of Watson [32] for the minimax problem, in which successive linear programming problems are solved until it is thought that the active set has been identified, whereupon a switch is made to solving a system of nonlinear equations by Newton's method. The system has order $n+t$, since the variables and multipliers are obtained together. Since t may often be close to n (t equals n in the linear case), the systems of equations which are solved may be much larger than the ones we solve.

We are not aware of any published methods of the projected Lagrangian type for problem l_1P, although we understand that Bartels and Conn are currently doing some related work. It would be possible to construct a method related to ours but which solves an inequality-constrained quadratic program variant of QP1 at each iteration as Han [19] does for the minimax problem. However, such a QP has $n+m$ variables and it is *not* possible to transform this directly to an inequality-constrained QP in n variables (as we transform QP1 to QP2). It would be necessary to solve the inequality-constrained QP by a special-purpose method taking into account the special structure, just as the Bartels, Conn and Sinclair method essentially solves the linear program equivalent of ELP by a special-purpose method. See [24] for remarks concerning the relative merits of solving the equality and inequality-constrained QP's.

11. Asymptotic convergence results. We can make use of the known asymptotic convergence results for nonlinear programming since in the limit our method becomes a projected Lagrangian method for constrained optimization applied to the special case ELP. We assume that the $\{f_i\}$ are twice continuously differentiable, that $\overset{*}{Z}^T\overset{*}{W}\overset{*}{Z}$ is positive definite, that $\overset{*}{V}$ has full rank, that $|\overset{*}{\pi}|<1$ ($\lambda^*>0$), and that in a neighborhood of the solution the steplength is always one. All of these assumptions correspond to standard assumptions in proofs of superlinear convergence for constrained optimization. Under these conditions our algorithm is locally quadratically convergent in the case that analytical Hessians are used. The proof of this follows from the proof of Robinson [29] that Wilson's algorithm is quadratically convergent combined with the proof of Fletcher [13] that using first-order multiplier estimates does not inhibit quadratic convergence. Superlinear convergence in the case of a particular quasi-Newton approximation to W follows from Powell [27].

12. Computational results. We present the results from applying the algorithm to l_1 problems with the same $\{f_i(x)\}$ as the first four problems presented in [24]. The solutions obtained are listed below, together with references where the definitions of the $\{f_i(x)\}$ and the starting point used may be found.

Problem 1. (Bard [3, p. 170].)

$$F_1(\overset{*}{x}) = 0.12434 \quad \text{with} \quad \overset{*}{x} = (0.10094, 1.52515, 1.97211)^T.$$

Problem 2. (Kowalik and Osborne [20, p. 104, ex. (v)].)

$$F_1(\overset{*}{x}) = 0.0038768 \quad \text{with} \quad \overset{*}{x} = (0.19337, 0.19377, 0.10893, 0.13973)^T.$$

Problem 3. (Madsen [21, p. 326, ex. 2].)

$$F_1(\overset{*}{x}) = 1.00000 \quad \text{with} \quad \overset{*}{x} = (0.0000, 0.0002)^T.$$

Problem 4. (El-Attar et al. [11, p. 81, ex. 2].)

$$F_1(\overset{*}{x}) = 7.8942 \quad \text{with} \quad \overset{*}{x} = (0.53597, 0.00000, 0.031918)^T.$$

The results are summarized in Table 1. The termination conditions were that $\|\hat{f}\|_2 < 10^{-6}$, $\|Z^T\bar{V}\bar{\Sigma}\bar{e}\|_2 < 10^{-6}$, Z^TWZ numerically positive definite and $|\pi_C| \leqq 1$. The line search accuracy parameter η was set to 0.9 (see [23] for the definition of this parameter). Several other choices of η were tried, but $\eta = 0.9$ was the most efficient, indicating, as expected, that a slack line search is desirable, at least on these problems. The machine used was an IBM 370/168 in double precision, i.e., with 16 decimal digits of accuracy. The column headed NI reports the number of iterations required, which is also the number of times the Hessian was approximated using finite differences. The column headed NF gives the number of function evaluations (not including gradient evaluations for the Hessian approximation).

TABLE 1

	Problem	n	m	$n-\bar{t}$	NI	NF
1	(Bard [3])	3	15	0	20	20
2	(Kowalik and Osborne [20])	4	11	0	11	14
3	(Madsen [21])	2	3	0	15	15
4	(El-Attar et al. [11, #2])	3	6	2	10	11

These results demonstrate that our algorithm can be very efficient. Final quadratic convergence was observed in all cases except Problem 3, for which $\overset{*}{V}$ is rank deficient. The results must, however, be regarded as preliminary since further work needs to be done regarding the active set strategy.

13. Concluding remarks. The motivation for requiring the search direction at each iteration to be a first-order feasible descent direction for ELP and, if possible, to be a solution of the quadratic program QP1, is quite similar to that presented in [24] for the minimax problem. This is because both the minimax and l_1 problems are equivalent to differentiable constrained optimization problems with natural merit functions available to measure progress towards a solution. However the details of the algorithm presented here for the l_1 problem are considerably different from the algorithm of [24]. In particular, although ELP and QP1 involve m extra variables, we have shown how to derive a method which solves successive quadratic programming problems in only n variables. Other points which have been emphasized include the

different roles of multiplier estimates and directions of search in the $(n+m)$- and n-dimensional spaces.

We could repeat many of the concluding remarks of [24] here. For example, we observe that linear constraints can be incorporated directly into the algorithm but that nonlinear constraints increase the complexity of problem l_1P to that of the general nonlinear constrained optimization problem. In summary, the method of this paper has been designed to take advantage of all the special properties of the l_1 problem which are not available for general constrained optimization problems.

Acknowledgments. The authors would like to thank Philip E. Gill, Gene H. Golub, and Margaret H. Wright for a number of helpful discussions.

REFERENCES

[1] D. H. ANDERSON AND M. R. OSBORNE, *Discrete, nonlinear approximations in polyhedral norms: a Levenberg-like algorithm*, Numer. Math. 28, (1977), pp. 157–170.

[2] R. D. ARMSTRONG AND J. P. GODFREY, *Two linear programming algorithms for the linear discrete L_1 norm problem*, Math. Comp. 33, (1979), pp. 289–300.

[3] Y. BARD, *Comparison of gradient methods for the solution of nonlinear parameter estimation problems*, SIAM J. Numer. Anal., 7 (1970), pp. 157–186.

[4] I. BARRODALE AND F. D. K. ROBERTS, *An improved algorithm for discrete l_1 linear approximation*, SIAM J. Numer. Anal., 10 (1973), pp. 839–848.

[5] ———, *An efficient algorithm for discrete l_1 linear approximation with linear constraints*, SIAM J. Numer. Anal. 15, (1978), pp. 603–611.

[6] R. BARTELS, A. R. CONN AND J. W. SINCLAIR, *Minimization techniques for piecewise differentiable functions: The l_1 solution to an overdetermined linear system*, SIAM J. Numer. Anal., 15 (1978), pp. 224–241.

[7] C. CHARALAMBOUS, *On the conditions for optimality of the nonlinear l_1 problem*, Math. Programming 17 (1979), pp. 123–135.

[8] A. CHARNES, W. W. COOPER AND R. FERGUSON, *Optimal estimation of executive compensation by linear programming*, Management Sci., 2 (1955), pp. 138–151.

[9] J. F. CLAERBOUT AND F. MUIR, *Robust modelling with erratic data*, Geophysics, 38 (1973), pp. 826–844.

[10] A. R. CONN AND T. PIETRZYKOWSKI, *A penalty function method converging directly to a constrained optimum*, SIAM J. Numer. Anal., 14 (1977), pp. 348–375.

[11] R. A. EL-ATTAR, M. VIDYASAGAR AND S. R. K. DUTTA, *An algorithm for l_1-norm minimization with application to nonlinear l_1-approximation*, SIAM J. Numer. Anal., 16 (1979), pp. 70–86.

[12] A. V. FIACCO AND G. P. MCCORMICK, *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York, 1968.

[13] R. FLETCHER, *Methods related to Lagrangian functions*, in *Numerical Methods for Constrained Optimization*, P. E. Gill and W. Murray, eds., Academic Press, New York and London, 1974, pp. 219–240.

[14] R. FLETCHER AND G. A. WATSON, *First and second-order conditions for a class of nondifferentiable optimization problems*, Math. Programming, 18 (1980), pp. 291–307.

[15] A. R. GALLANT AND T. M. GERIG, *Comments on computing minimum absolute deviations regressions by iterative least squares regressions and by linear programming*, Institute of Statistics Rep. 911, North Carolina State University, 1974.

[16] P. E. GILL AND W. MURRAY, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Programming, 7 (1974), pp. 311–350.

[17] ———, *The computation of Lagrange multiplier estimates for constrained minimization*, National Physical Laboratory Rep. NAC 77, Teddington, England, 1977.

[18] S. P. HAN, *A globally convergent method for nonlinear programming*, J. Optim. Theory Appl., 22 (1977), pp. 297–309.

[19] ———, *Variable metric methods for minimizing a class of nondifferentiable functions*, Dept. of Computer Science Rep., Cornell University, Ithaca, New York, 1977.

[20] J. KOWALIK AND M. R. OSBORNE, *Methods for Unconstrained Optimization Problems*, Elsevier, New York, 1968.

[21] K. MADSEN, *An algorithm for the minimax solution of overdetermined systems of nonlinear equations*, J. Inst. Math. Appl., 16, (1975), pp. 321–328.

[22] R. A. MCLEAN AND G. A. WATSON, *Numerical methods for nonlinear discrete L_1 approximation problems*, Proc. Oberwolfach Conf. on Approx. Theory, 1979, to appear.

[23] W. MURRAY AND M. L. OVERTON, *Steplength algorithms for minimizing a class of nondifferentiable functions*, Computing, 23 (1979), pp. 309–331.

[24] ——— *A projected Lagrangian algorithm for nonlinear minimax optimization*, this Journal, 1 (1980), pp. 345–370.

[25] W. MURRAY AND M. H. WRIGHT, *Projected Lagrangian methods based on the trajectories of penalty and barrier functions*, Systems Optimization Laboratory Report SOL-78-23, Stanford University, Stanford, California, 1978.

[26] M. R. OSBORNE AND G. A. WATSON, *On an algorithm for discrete nonlinear L_1 approximation*, Comput. J., 14 (1971), pp. 184–188.

[27] M. J. D. POWELL, *The convergence of variable metric methods for nonlinearly constrained optimization calculations*, Rep. DAMTP77/NA3, University of Cambridge, Cambridge, England, 1977.

[28] J. R. RICE AND K. H. USOW, *Lawson's algorithm and extensions*, Math. Comp. 22, (1968), pp. 118–127.

[29] S. M. ROBINSON, *Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear programming algorithms*, Math. Programming, 7 (1974), pp. 1–16.

[30] E. J. SCHLOSSMACHER, *An iterative technique for absolute deviation curve fitting*, J. Amer. Statist. Assoc., 56 (1973), pp. 359–362.

[31] D. R. SCHUETTE, *A defense of the Karst algorithm for finding the line of best fit under the L_1 norm*, Math. Res. Center Rep. 1735, University of Wisconsin, Madison, 1977.

[32] G. A. WATSON, *The minimax solution of an overdetermined system of nonlinear equations*, J. Inst. Math. Appl., 23 (1979), pp. 167–180.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0010 $01.00/0

ALGORITHMS AND DATA STRUCTURES FOR SPARSE SYMMETRIC GAUSSIAN ELIMINATION*

STANLEY C. EISENSTAT†, MARTIN H. SCHULTZ† AND ANDREW H. SHERMAN‡

Abstract. In this paper we present algorithms and data structures that may be used in the efficient implementation of symmetric Gaussian elimination for sparse systems of linear equations with positive definite coefficient matrices. The techniques described here serve as the basis for the symmetric codes in the Yale Sparse Matrix Package.

Key words. sparse matrices, sparse Gaussian elimination, data structures

1. Introduction.

A central task in the numerical solution of important scientific and engineering problems is quite often the solution of a system of linear equations

$$Ax = b, \tag{1.1}$$

where A is an $N \times N$ sparse symmetric positive definite matrix. For instance, this is frequently the situation when finite difference or finite element methods are used to discretize linear partial differential equations arising from mathematical models in such areas as structural analysis. As another example, the modeling of nonlinear phenomena may lead to a large sparse system of nonlinear equations that can often be solved using a variant of Newton's method in which, at each step, a system of linear equations like (1.1) must be solved.

In the past several years, much attention has been focused on the use of Gaussian elimination for the solution of (1.1). With a variety of theoretical and practical tools, great progress has been made towards the joint goals of numerical accuracy in the solution and economy in terms of computing time and memory space. Several packages of Fortran subprograms [5], [10], [12] for the solution of (1.1) have been developed and widely distributed as a part of research in this area, and these have been gaining increasing acceptance in the scientific community.

The intent of this paper is to present and analyze several algorithms that are used in the Yale Sparse Matrix Package [5], one of the Fortran packages mentioned above. Publications elsewhere [5]–[7], [16] describe the software package and its use and discuss the rationale behind some of the design decisions. In this paper, however, we provide a complete detailed analysis. The conclusions of the analysis are borne out by a variety of experimental results reported, for example, in [4], [13], [17].

It is well known that the matrix A of (1.1) can always be factored in the form

$$A = U^T D U \tag{1.2}$$

where U is unit upper triangular and D is diagonal. Symmetric Gaussian elimination

* Received by the editors March 31, 1980, and in revised form, February 5, 1981. This work was supported in part by the U.S. Office of Naval Research under grant N00014-76-C-0277 and by the U.S. Air Force Office of Scientific Research under contract F49620-77-C-0037.

† Department of Computer Science, Yale University, New Haven, Connecticut 06520.

‡ Department of Computer Sciences, University of Texas at Austin, Austin, Texas 78712. Part of the work of this author was completed while on leave at the Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A7.

for (1.1) is equivalent to first factoring A in this way and then successively solving the systems:

$$U^T y = b, \quad Dz = y, \quad Ux = z \tag{1.3}$$

to obtain x. For reasons of economy, we wish to factor A and compute x without storing or operating on zeros in A and U. This requires a certain amount of storage and operational overhead; that is, extra storage for pointers to the locations of the nonzeros in addition to that needed for the numerical values of the matrix, and extra nonnumeric "bookkeeping" operations in addition to the required arithmetic operations. Our goal here is to describe a set of algorithms that can be implemented with little overhead.

We use a particularly robust algorithm designed by Chang [2] and previously used by Gustavson [11]. The computation is broken up into three distinct steps: symbolic factorization (SYMFAC), numeric factorization (NUMFAC), and forward- and back-solution (SOLVE). The SYMFAC step computes the zero structure of U (i.e., the positions of the nonzeros in U) from that of A, disregarding the actual numerical entries of A. The NUMFAC step then uses the structural information generated by SYMFAC to compute the numerical values of U and D. Finally, the SOLVE step uses the information produced by both SYMFAC and NUMFAC to solve the resulting triangular and diagonal systems for x.

The main advantage of splitting up the computation is flexibility. If several linear systems have identical coefficient matrices but different right-hand sides, only one SYMFAC and one NUMFAC step are needed; the different right-hand sides require only separate SOLVE steps. Similarly, a sequence of linear systems whose coefficient matrices have identical zero structures but different numerical entries can be solved by using just one SYMFAC step combined with separate NUMFAC and SOLVE steps for each system.

The algorithms for the SYMFAC, NUMFAC, and SOLVE steps are clearly interdependent and, to some extent, depend also on the data structures used to store A and U. Since experience shows that the NUMFAC step requires substantially more computational effort than the other steps, the development in this paper is driven by the requirements of the NUMFAC algorithm; that is, we first design an efficient NUMFAC algorithm and then tailor the SYMFAC and SOLVE algorithms and data structures accordingly. Thus, in § 2, we construct efficient NUMFAC and SOLVE algorithms; in § 3, we describe data structures for efficiently storing A and U; and in § 4, we develop an efficient SYMFAC algorithm.

As an aside, we note that this paper does not consider the possibility of reordering the equations and unknowns of (1.1) in order to reduce the number of nonzeros in U and the number of arithmetic operations required to factor A and solve for x. In practice, it is quite important to do this (see, for example, [17]), but the effect on the topics we discuss here is minimal, since for positive definite systems, the reordering may be completed as a preprocessing step. In fact, the Yale Sparse Matrix Package (and, for that matter, most software packages available for solving (1.1) with sparse Gaussian elimination) accept as input a permutation corresponding to a reordering of the equations and unknowns and provide at least one subprogram for computing a good permutation.

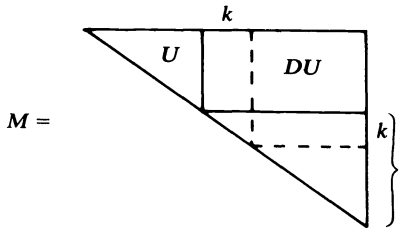
2. Numerical factorization and solution. We begin by examining a factorization algorithm for dense matrices. Such algorithms are well known (see Forsythe and Moler [8]), and Algorithm 2.1 is a row-oriented version. In the algorithm the diagonal entries d_{kk} are stored in a vector D of length N in which $D(k) = d_{kk}$. To avoid notational

confusion, we have presented the algorithm as if the matrix computations were performed on an upper triangular matrix M, although in a standard implementation, all the computations would be performed on the upper triangular portions of A and U.

ALGORITHM 2.1

```
1.  For k = 1 to N do
2.      [D(k) = a_kk;
3.      For j = k+1 to N do
4.          [m_kj = a_kj];
5.      For i = 1 to k-1 do
6.          [t = m_ik;
7.          m_ik = m_ik/D(i);
8.          D(k) = D(k) - t · m_ik;
9.          For j = k+1 to N do
10.             [m_kj = m_kj - m_ik · m_ij]]];
```

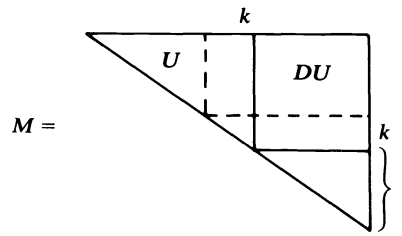
At any time during the execution of Algorithm 2.1, part of M contains entries of U, part contains entries of DU (the matrix product of D and U), and part is unspecified. Figure 2.1a shows the contents of M just prior to the start of the kth step of the factorization. During the kth step, the algorithm computes d_{kk} (line 8), the kth column of U in the kth column of M (line 7), and the kth row of DU in the kth row of M (lines 9–10). Figure 2.1b shows the contents of M at the conclusion of the kth step. At the end of the factorization, M contains exactly the entries of U.



Unspecified

$$m_{ij} = \begin{cases} 0 & j < i \\ 1 & j = i < k \\ u_{ij} & i < j < k \\ (DU)_{ij} & i < k;\ j \geqq k \\ \text{Unspecified} & k \leqq i \leqq j \end{cases}$$

(a) *Prior to kth factorization step*



Unspecified

$$m_{ij} = \begin{cases} 0 & j < i \\ 1 & j = i \leqq k \\ u_{ij} & i < j \leqq k \\ (DU)_{ij} & i \leqq k;\ j > k \\ \text{Unspecified} & k < i \leqq j \end{cases}$$

(b) *After kth factorization step*

FIG. 2.1.

When A is dense, Algorithm 2.1 can be implemented efficiently, since it stores and operates on just the diagonal of D and the upper triangles of A and U. However, when A is sparse, the algorithm fails to exploit the zeros in A and U to reduce the storage and work. The bulk of this section is devoted to the development of an efficient numerical factorization algorithm for this case.

Conceptually, at least, it is possible to avoid arithmetic operations on zeros by explicitly testing the operands prior to using them; unfortunately, there are two serious problems with this simple approach. First, all the entries in the upper triangles of A

and M would have to be stored, since any of them could be tested or used as m_{kj} in line 10 of Algorithm 2.1. Second, there would be more test operations than arithmetic operations, so that the running time of such an algorithm would be asymptotically proportional to the amount of testing rather than to the amount of arithmetic op(A)[1].

To overcome these difficulties, assume that at the beginning of the kth step of the algorithm, we know:

(i) the set R_k^A of columns $j>k$ for which $a_{kj}\neq 0$;
(ii) the set R_k^U of columns $j>k$ for which $u_{kj}\neq 0$;
(iii) the set C_k of rows $i<k$ for which $u_{ik}\neq 0$;
(iv) for each $i<k$, the set $R_{i,k}^U$ of columns $j>k$ for which $u_{ij}\neq 0$.

Then we can modify Algorithm 2.1 to obtain Algorithm 2.2 in which the only entries of M used are those corresponding to nonzeros in A or U.

ALGORITHM 2.2

```
1.  For k=1 to N do
2.      [D(k)=a_kk;
3.      For j∈R_k^U do
4.          [m_kj=0];
5.      For j∈R_k^A do
6.          [m_kj=a_kj];
7.      For i∈C_k do
8.          [t=m_ik;
9.          m_ik=m_ik/D(i);
10.         D(k)=D(k)-t·m_ik;
11.         For j∈R_{i,k}^U do
12.             [m_kj=m_kj-m_ik·m_ij]]];
```

We now turn to the implementation of Algorithm 2.2. This requires that we deal with the problem of "fillin", that is, with the creation of nonzeros in matrix positions of U corresponding to zeros in A. A straightforward means of handling fillin involves list-processing. At the kth step of the algorithm, all of the fillin occurs in the kth row, so, by keeping that row as a linked list, it is possible to easily insert new nonzeros in their proper places. After the kth step no additional fillin will occur in the kth row, so it can be stored in a more efficient manner if desired. Unfortunately, the list-processing approach is not particularly efficient in terms of computation time, since potentially it requires a sweep through the entire linked list for each execution of the innermost loop, even though $|R_{i,k}^U|$ might be quite small.[2]

To avoid list-processing, we use a technique known as "row expansion" that was first suggested by Gustavson [11]. A modification of Algorithm 2.2 using row expansion is given as Algorithm 2.3. During the kth step of Algorithm 2.3, row expansion consists of three phases:

(i) The kth row of A is expanded into a vector V of length N so that $V(j)=a_{kj}$ for $j\in R_k^U\cup\{k\}$. We assume that $R_k^A\subseteq R_k^U$; that is, we do not allow for accidental cancellation in which a nonzero in A becomes zero in U.

(ii) For each $i\in C_k$, a multiple of row i of U is subtracted from V. No special attention need be given to fillin since, if the jth column fills in, then $V(j)$ was set to zero in (i) before the row operations began.

[1] We use op(A) to denote the number of multiplications and divisions required to factor the sparse matrix A, including only operations involving two nonzero operands.

[2] We use $|S|$ to denote the size of the set S.

(iii) The kth row of DU is obtained by storing $V(j)$ in $(DU)_{kj}$ for $j \in R_k^U$. Since $(DU)_{kj} \neq 0$ if and only if $j = k$ or $u_{kj} \neq 0$ (i.e., $j \in R_k^U$), this storage operation stores only nonzeros. Moreover, since all fillin in the kth row occurs during the kth step of the algorithm, this storage operation will implicitly take care of the insertion of fillin entries in the kth row of U.

At this point we have an implementation that is quite time-efficient but apparently requires a great deal of storage for auxiliary information about A and U. Certainly, the sets R_k^A and R_k^U must be stored because they describe the zero structures of A and U, respectively. However, the sets C_k and $R_{i,k}^U$ do not really contain new information about U; instead, they simply present the same structure information in a somewhat different way. To save storage, we compute these sets from the sets R_k^U.

Observe that C_k and $R_{i,k}^U$, $1 \leqq i \leqq k$, are required only during the kth step of Algorithm 2.3. We can exploit this by a scheme in which the computation of these sets need not be completed until the end of the $(k-1)$st step. During previous steps we allow these sets to be only partially computed.

ALGORITHM 2.3

```
1.  For k = 1 to N do
2.      [D(k) = a_kk;
3.      For j ∈ R_k^U do
4.          [V(j) = 0];
5.      For j ∈ R_k^A do
6.          [V(j) = a_kj];
7.      For i ∈ C_k do
8.          [t = m_ik;
9.          m_ik = m_ik/D(i);
10.         D(k) = D(k) − t · m_ik;
11.         For j ∈ R_{i,k}^U do
12.             [V(j) = V(j) − m_ik · m_ij]]];
```

The key to this scheme is to keep the sets R_k^U ordered by increasing column number. Then for $i < k$, $R_{i,k}^U$ contains simply the segment of R_i^U beginning with the first entry larger than k and including all succeeding entries. If each set R_k^U is stored as a sequence of integers in consecutive locations in an array JU, then we can use a vector IL of length N to obtain the sets $R_{i,k}^U$ as required. We just make certain that, at the beginning of the kth step, $IL(i)$ points to the location of the first entry in $R_{i,k}^U$. At the conclusion of the kth step, we must update $IL(i)$ for exactly those $i \in C_k \cup \{k\}$, since $R_{i,k}^U = R_{i,k+1}^U$ for all other i. For $i \in C_k$ we set $IL(i) = IL(i) + 1$, while for $i = k$, we set $IL(k)$ to point to the location of the first entry of R_k^U.

To compute the sets C_k, we employ sets P_k that are constructed in such a way that, during the kth step,

(i) $P_k = C_k$, and

(ii) for $j > k$, P_j contains those row indices $i < k$ such that $i \in C_j$ but $i \notin C_m$, $k \leqq m < j$.

During the kth step we use P_k in place of C_k, and, at the conclusion of the kth step, we update the sets P_j, $j > k$, so that conditions (i) and (ii) are satisfied during the $(k+1)$st step. This update is accomplished by moving the entries of P_k into the appropriate sets P_j, $j > k$. For each $i \in P_k$, if $j > k$ is the smallest entry of $R_{i,k+1}^U$, then i is placed into the set P_j. If $R_{i,k+1}^U$ is empty, i is not placed into any set.

To store the sets P_k we use a special form of linked list in which the data and pointers coincide. All of the information about these sets is stored in a single array JL

of length N. At the kth step each entry $JL(j)$, $k \leqq j \leqq N$, continues the row index that is the first entry of P_j, if any, or zero otherwise. For each nonzero $JL(i)$, $JL(JL(i))$ is either the next row index in the same set as $JL(i)$ or zero if there are no more entries in that set. This scheme works because, at the kth step, the sets P_j, $k \leqq j \leqq N$, are disjoint and contain only row indices $i < k$. As an illustration, Figure 2.2a shows a storage configuration that might ensue during the factorization algorithm.

$$C_k = \{2, 4, 5\} \quad C_{k+1} = \{2, 3\} \quad C_{k+2} = \{1, 2, 4\}$$

$$R_1^U = \{2, \cdots, k-1, k+2, \cdots\}$$

$$R_2^U = \{3, \cdots, k, k+1, k+2, \cdots\}$$

$$R_3^U = \{4, \cdots, k-1, k+1, \cdots\}$$

$$R_4^U = \{5, \cdots, k, k+2, \cdots\}$$

$$R_5^U = \{6, \cdots, k\}$$

(a) At step k:
$$\begin{cases} P_k = \{2, 4, 5\} \\ P_{k+1} = \{3\} \\ P_{k+2} = \{1\} \end{cases}$$

(b) At step $k+1$:
$$\begin{cases} P_k = \emptyset \\ P_{k+1} = \{3, 2\} \\ P_{k+2} = \{1, 4\} \end{cases}$$

FIG. 2.2

To actually update JL to reflect a move of row index i from P_k to P_j, we save temporarily the value of $JL(i)$ (since it is the next row index in P_k), set $JL(i) = JL(j)$, and set $JL(j) = i$. We then go on to the next entry of P_k. Figure 2.2b shows the storage configuration that would result following the update of JL at the kth step, beginning from the configuration shown in Figure 2.2a.

Algorithm 2.4 is a modification of Algorithm 2.3 that incorporates the refinements introduced in this section. In addition, it uses the vector D for both D and V, based on the observation that, at the kth step, the algorithm requires only the first $k-1$ entries of D and the last $N-k+1$ entries of V. From our previous discussion it is clear that the algorithm is efficient in terms of storage, since only about $2N$ storage locations are required in addition to the space for the numerical entries of A, D, and U and the structure descriptions of A and U.

To determine the operational overhead of the algorithm, we examine separately the times required for arithmetic operations and set-updating operations. There are $O(\text{op}(A))$ arithmetic operations (since we avoid operations on zeros), and each one requires constant time, due to the use of row expansion. Thus the NUMFAC algorithm requires a total of $O(\text{op}(A))$ time for arithmetic operations. At the kth step, one set-updating operation is required for each entry of $C_k \cup \{k\}$, so that, overall, $O(nz(U)+N)$ updating operations[3] are performed. Since each set-updating operation costs only constant time, together they require a total of only $O(nz(U)+N)$ time, and the entire algorithm runs in $O(\text{op}(A))$ time since $nz(U)+N \leqq \text{op}(A)$.

We conclude this section by presenting the SOLVE algorithm for the forward- and back-solution of sparse symmetric systems. Algorithm 2.5 successively solves the

[3] We use $nz(M)$ to denote the number of nonzero entries of the array M that must be stored in any sparse representation of M. For symmetric matrices, this includes only nonzero entries in the upper triangle.

ALGORITHM 2.4

```
1.  For k = 1 to N do
2.      [P_k = ∅];
3.  For k = 1 to N do
4.      [D(k) = a_kk;
5.      For j ∈ R_k^U do
6.          [D(j) = 0];
7.      For j ∈ R_k^A do
8.          [D(j) = a_kj];
        For i ∈ P_k do
10.         [t = m_ik;
11.         m_ik = m_ik/D(i);
12.         D(k) = D(k) − t · m_ik;
13.         For j ∈ R_{i,k}^U do
14.             [D(j) = D(j) − m_ik · m_ij];
15.         R_{i,k}^U = R_{i,k}^U − {k + 1};
16.         If R_{i,k+1}^U ≠ ∅ then do
17.           [j = min (R_{i,k+1}^U);
18.           P_j = P_j ∪ {i}]];
19.     If R_k^U ≠ ∅ then do
20.       [j = min (R_k^U);
21.       P_j = P_j ∪ {k}]];
```

ALGORITHM 2.5

```
1.  For k = 1 to N do
2.      [y_k = b_k];
3.  For k = 1 to N − 1 do
4.      [For j ∈ R_k^U do
5.          [y_j = y_j − u_kj · y_k]];
6.  For k = 1 to N do
7.      [z_k = y_k/D(k)];
8.  For k = N to 1 by −1 do
9.      [x_k = z_k;
10.     For j ∈ R_k^U do
11.         [x_k = x_k − u_kj · x_j]];
```

systems (1.3), and it fits in well with the NUMFAC algorithm because it uses only the nonzero entries of A, D, and U and the structure information for A and U. It is quite easy to see that the algorithm requires only $O(nz(U)+N)$ time, since it operates exactly once on each nonzero in U in solving each of the two triangular systems. The cost of solving the diagonal system, of course, is only $O(N)$.

3. Storage of sparse matrices. In the last section we developed an efficient algorithm for the numerical U^TDU factorization of A. That algorithm used as data the nonzeros of A and U and the sets R_k^A and R_k^U, $1 \leqq k \leqq N$, that describe the structures of the rows of A and U, respectively. In this section we describe data structures that can be used to efficiently store all of this information.

All of the storage schemes we consider are row-oriented; that is, information about the matrix is organized row-by-row. This makes it easy to access information about the rows but quite difficult to obtain information about the columns. In general, this might

be a serious drawback in a storage scheme; however, as we have seen, information about the columns is not required in this application.

The first storage scheme is the "uncompressed storage scheme" that is employed in various forms by Gustavson [11], Curtis and Reid [3], Munksgaard [12], and others. The nonzero matrix entries are stored by rows in order of increasing column index. To identify the entries of any row, it is necessary to know where the row begins, how many entries it contains, and in what columns the entries lie. This extra information about the structure of the matrix is the storage overhead mentioned in § 1.

The uncompressed storage scheme for the matrix A uses three arrays (IA, JA, and A) as illustrated in Fig. 3.1. The array A contains the nonzero entries of the upper

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & 0 \\ 0 & a_{32} & a_{33} & 0 & 0 & 0 \\ 0 & a_{42} & 0 & a_{44} & 0 & a_{46} \\ 0 & a_{52} & 0 & 0 & a_{55} & 0 \\ 0 & 0 & 0 & a_{64} & 0 & a_{66} \end{bmatrix}$$

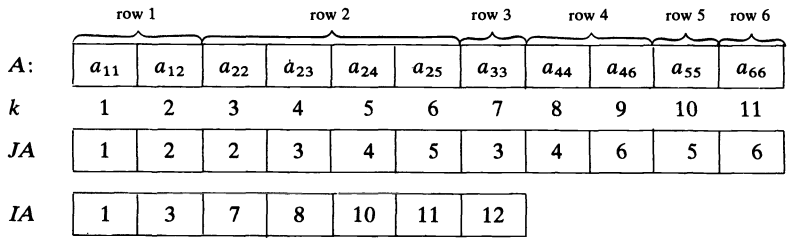


FIG. 3.1

triangle stored row-by-row.[4] The array JA contains the column indices that correspond to the nonzeros in the array A: if $A(k)$ contains a_{ij}, then $JA(k) = j$. Finally, the array IA contains $N+1$ pointers that delimit the rows of nonzeros in A and indices in JA; i.e., $A(IA(i))$ is the first entry stored for the ith row, and $A(IA(i+1)-1)$ is the last. Thus the length of the ith row is given by $IA(i+1)-IA(i)$. Since R_i^A includes all but the first of the column indices stored in JA for row i, we see that the uncompressed storage scheme can be used quite naturally with Algorithms 2.4 and 2.5.

The storage overhead incurred when using the uncompressed storage scheme for A is the storage for the arrays IA and JA. Since IA has $N+1$ entries and JA has one entry per nonzero in the upper triangle of A, the storage overhead is approximately equal to $nz(A)$.

It is possible to use the uncompressed storage scheme for U as well as A, as illustrated in Fig. 3.2a. However, this ignores certain features of U that allow a potentially significant reduction in the space required for column indices. Since storage is often at a premium when solving sparse linear systems, we have chosen to use a more complex "compressed storage scheme" that usually leads to a substantial reduction in storage overhead at the cost of at most a small increase in operational overhead (see [7]).

Figure 3.2a shows the data structures required to store a particular matrix U in the uncompressed storage scheme. It is immediately evident that the diagonal entries

[4] A is symmetric, so only the upper triangle is stored.

$$U = \begin{bmatrix} 1 & u_{12} & 0 & 0 & 0 & 0 \\ 0 & 1 & u_{23} & u_{24} & u_{25} & 0 \\ 0 & 0 & 1 & u_{34} & u_{35} & 0 \\ 0 & 0 & 0 & 1 & u_{45} & u_{46} \\ 0 & 0 & 0 & 0 & 1 & u_{56} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(a)

U:	1	u_{12}	1	u_{23}	u_{24}	u_{25}	1	u_{34}	u_{35}	1	u_{45}	u_{46}	1	u_{56}	1
k:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
JU:	1	2	2	3	4	5	3	4	5	4	5	6	5	6	6

IU:	1	3	7	10	13	15	16

(b)

U:	u_{12}	u_{23}	u_{24}	u_{25}	u_{34}	u_{35}	u_{45}	u_{46}	u_{56}
k:	1	2	3	4	5	6	7	8	9
JU:	2	3	4	5	4	5	5	6	6

IU:	1	2	5	7	9	10	10

(c)

U:	u_{12}	u_{23}	u_{24}	u_{25}	u_{34}	u_{35}	u_{45}	u_{46}	u_{56}
k:	1	2	3	4	5	6	7	8	9
JU:	2	3	4	5	6				

IU:	1	2	5	7	9	10	10

JU:	1	2	3	4	5	5

FIG. 3.2

need not be stored, since they are always equal to one and occur as the first stored entry of each row. Figure 3.2b shows the data structures required when the diagonal entries are omitted.

We now consider ways to "compress" JU. Assume that we are creating the data structure for U by adding one row at a time, and consider the situation as we add row k. Let row j be the highest-numbered row whose indices are stored terminating in the last position in JU used to store column indices for the first $k-1$ rows. There are two main cases in which compression can take place.

First, the indices for row k are the same as the last several indices for some preceding row i. In this case we can use the indices stored for row i to avoid storing a separate set of indices for row k; all that is needed is a pointer to locate the start of the indices for row k within those for row i.[5] In Figure 3.2b the indices for row 3 are 4, 5; while those for row 2 are 3, 4, 5. Instead of storing the indices for row 3 separately, we simply use the last two indices already stored for row 2.

[5] The number of indices for row k can be determined from IU.

Second, the first several indices for row k are the same as the last several for row j. In this case we can overlap the indices for rows j and k. Again all that is needed is a pointer to locate the beginning of the indices for row k. In Figure 3.2b the indices for row 4 are 5, 6; while those for row 3 are 4, 5. Overlapping these two sets of indices allows us to avoid duplicating the index 5 in JU.

In general, the compressed indices in JU do not correspond directly to the nonzeros stored in the array U, so an extra array of pointers (IJU) is required to locate the start of the indices for each row (see Figure 3.2c). Thus the array U contains the nonzero entries of the strict upper triangle of U stored row-by-row; the $N+1$ entries of IU delimit the rows in U as before; JU is the compressed array of column indices; and the N entries of IJU point to the first column index in JU for each row. The nonzeros of the ith row of the strict upper triangle of U are stored in $U(IU(i))$ through $U(IU(i+1)-1)$, and the corresponding column indices are stored in $JU(IJU(i))$ through $JU(IJU(i)+IU(i+1)-IU(i)-1)$. Note that, since the entries of each row are ordered by increasing column number, the column indices stored in JU for row i form precisely the set R_i^U required by Algorithms 2.4 and 2.5.

The storage overhead for the compressed storage of U is the number of locations required for IU, JU, and IJU. Although, for small problems, this overhead can be slightly larger than that for the uncompressed scheme, it is usually substantially smaller. For certain systems arising in the solution of elliptic boundary value problems in a square, for instance, the overhead is actually $O(N)$ storage locations as opposed to the $O(N \log N)$ nonzeros in U (see [16]).

4. Symbolic factorization. In § 2 we presented NUMFAC and SOLVE algorithms that required as input data the sets R_k^A and R_k^U, $1 \leqq k \leqq N$. As we saw in the last section, the sets R_k^A are available directly from the uncompressed storage scheme used to store A. In this section we develop a SYMFAC algorithm to efficiently compute the ordered sets R_k^U. To simplify the presentation, we will assume that the R_k^U are to be computed as unordered sets; at the end of the section, we will discuss how to obtain ordered sets.

At the kth step we will compute R_k^U from R_k^A and the sets R_i^U, $i<k$. An examination of Algorithm 2.3 shows that $u_{kj} \neq 0$, $j>k$, if and only if either

(i) $a_{kj} \neq 0$, or

(ii) $u_{ij} \neq 0$ for some $i \in C_k$.

Thus $j \in R_k^U$ if and only if either

(i) $j \in R_k^A$ or

(ii) $j \in R_{i,k}^U$ for some $i \in C_k$.

This observation leads to Algorithm 4.1, which could be implemented efficiently by replacing references to $R_{i,k}^U$ and C_k as in Algorithm 2.4.

ALGORITHM 4.1

1. **For** $k=1$ **to** N **do**
2. $\quad[R_k^U=\emptyset]$;
3. **For** $k=1$ **to** $N-1$ **do**
4. $\quad$[**For** $j \in R_k^A$ **do**
5. $\quad\quad[R_k^U=R_k^U \cup \{j\}]$;
6. $\quad$**For** $i \in C_k$ **do**
7. $\quad\quad$[**For** $j \in R_{i,k}^U$ **do**
8. $\quad\quad\quad[R_k^U=R_k^U \cup \{j\}]]]$;

However, we can do better. Consider the example in Figure 4.1 in which fillin occurs in u_{24} and u_{34}. In computing R_3^U we use both $R_{1,3}^U$ and $R_{2,3}^U$, even though $R_{1,3}^U$ is redundant in the sense that $R_{1,3}^U \subseteq R_{1,2}^U \subseteq R_2^U$ and hence $R_{1,3}^U \subseteq R_{2,3}^U$.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & 0 & 0 & a_{44} \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$R^U_{1,3} = \{4\}$ $\quad R^U_2 = \{3, 4\}$

$R^U_{1,2} = \{3, 4\}$ $\quad R^U_{2,3} = \{4\}$

FIG. 4.1

In fact, this observation can be generalized. Let l_k be the set of rows $i \in C_k$ for which k is the minimum column index in R^U_i. The following result expresses a type of "transitivity condition" for the fillin in symmetric Gaussian elimination and implies that, in forming R^U_k, it suffices to examine $R^U_{i,k}$ for $i \in l_k$. (Similar results have been developed by others, e.g., [15].)

THEOREM 4.1 (Sherman [16]). *Let $i \in C_k$. Then either $i \in l_k$ or there is an m, $i < m < k$, such that $m \in l_k$ and $R^U_{i,k} \subseteq R^U_{m,k}$.*

Using the theorem we see that $j \in R^U_k$ if and only if either

(i) $j \in R^A_k$, or

(ii) $j \in R^U_{i,k}$ for some $i \in l_k$.

Algorithm 4.2 is a symbolic factorization algorithm based on this observation. The sets l_k are formed by adding each row i to the proper set l_m as soon as R^U_i is computed. In terms of implementation, the sets l_k are similar to the sets P_k introduced in § 2. In particular, only l_k and the partially computed sets l_m, $k < m \leqq N$, are required at the kth step. Since these are disjoint sets containing only row indices less than k, we can store them using a single array of length N, just as we did for the sets P_k.

ALGORITHM 4.2

```
1.  For k = 1 to N do
2.      [R_k^U = ∅;
3.      l_k = ∅];
4.  For k = 1 to N − 1 do
5.      [For j ∈ R_k^A do
6.          [R_k^U = R_k^U ∪ {j}];
7.      For i ∈ l_k do
8.          [For j ∈ R_{i,k}^U do
9.              [If j ∉ R_k^U then do
10.                 [R_k^U = R_k^U ∪ {j}]]];
11.     If R_k^U ≠ ∅ then do
12.         [i = min (R_k^U);
13.         l_i = l_i ∪ {k}]];
```

The cost of Algorithm 4.2 is determined by the costs of its innermost loop (lines 7–10) and of lines 5–6 and 11–13, since the remainder of the algorithm requires only $O(N)$ operations. At the kth step, for each $i \in l_k$ and $j \in R^U_{i,k}$, lines 9–10 are executed once. Since each row i is a member of exactly one set l_k, lines 9–10 are executed at most once for each entry of the combined sets R^U_i, $l \leqq i \leqq N$ (i.e., once for each nonzero entry of U). If the characteristic vector of R^U_k (cf., Aho, Hopcroft, and Ullman [1], p.

49]) is computed along with R_k^U, then the test in line 9 requires constant time, since it can be done by examining one entry of the characteristic vector. Furthermore, the union operation in line 10 also requires only constant time since j is simply appended to the end of R_k^U, and the jth entry of the characteristic vector is set to one. Hence a total of $O(nz(U))$ time is spent in the innermost loop of the algorithm.

At the kth step, line 6 is executed once for each entry in R_k^A. Thus overall, it is executed just once for each entry of the sets R_k^A, $1 \leqq k \leqq N$, combined (i.e., at most once for each nonzero entry of A). Again the union operation requires only constant time, and, since $nz(A) \leqq nz(U)$, the algorithm spends at most $O(nz(U))$ time in lines 5–6.

Finally, note that, for each k, lines 11–13 require $O(|R_k^U|)$ time, since the minimum operation of line 11 requires that much time, while the union operation of line 11 requires constant time. Thus overall, lines 11–13 and the entire algorithm both require only $O(nz(U))$ time.

As discussed earlier, we need each set R_k^U in increasing order. One solution to this difficulty is to modify Algorithm 4.2 to compute ordered sets directly by changing the set union operations to list insertion operations so that each new entry is inserted into its proper place. However, the time required for a list insertion operation depends on the length of the list into which the insertion is made, and, except in special cases, the modified algorithm may require more than $O(nz(U))$ time asymptotically. In practice, the entire loop in lines 8–10 would be replaced with a list merging operation to merge the ordered list R_i^U into R_k^U. However, the same conclusion would follow, since the merging operation could require $O(|R_k^U| + |R_i^U|)$ time.

An alternative to the use of list operations is to first compute the unordered sets and then sort them. Viewing the set $R = \bigcup_{k=1}^{N} R_k^U$ as a set of ordered pairs (k, j) for $j \in R_k^U$, we wish to sort R in lexicographic order; that is, so that (k, j) precedes (k', j') in R if and only if either

(i) $k < k'$ or

(ii) $k = k'$ and $j < j'$.

Since all the entries of the sets are integers between 1 and N, this sorting process can be accomplished efficiently by using a bucket sort (cf. Aho, Hopcroft, and Ullman [1, pp. 77–84]). This requires time proportional to the number of entries in the combined sets, that is, $O(nz(U))$ time, and, combining the sort with Algorithm 4.2, we obtain an algorithm for computing the ordered sets in $O(nz(U))$ time (cf., Rose and Whitten [15], Rose, Tarjan, and Lueker [14], George and Liu [9], and Sherman [16]).

From this discussion it seems clear that asymptotically one should use the second method for obtaining ordered sets. Surprisingly, however, experiments indicate that it is usually faster to compute the ordered sets directly with list operations. This is especially true for linear systems arising in the use of finite difference or finite element methods for the solution of partial differential equations, since it can be proved that computing the ordered sets directly requires only $O(nz(U))$ time for such systems. For this reason, the software based on Algorithm 4.2 (see [5]) uses the list merging version described above.

REFERENCES

[1] A. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] A. CHANG, *Application of sparse matrix methods in electric power system analysis*, in *Sparse Matrix Proceedings*, R. A. Willoughby, ed., Rep. RAI, IBM Research, Yorktown Heights, NY, 1968, pp. 113–122.

[3] A. R. CURTIS AND J. K. REID, *Fortran subroutines for the solution of sparse sets of linear equations*, AERE rep. R.6844, HMSO, London, 1971.

[4] S. C. EISENSTAT, J. A. GEORGE, R. GRIMES, D. R. KINCAID AND A. H. SHERMAN, *Some comparisons of software packages for large sparse linear systems*, in *Advances in Computer Methods for Partial Differential Equations—III*, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1979, pp. 98–106.

[5] S. C. EISENSTAT, M. C. GURSKY, M. H. SCHULTZ AND A. H. SHERMAN, *The Yale sparse matrix package I: Symmetric matrices*, Rep. 112, Dept. of Computer Science, Yale University, New Haven, CT, 1977.

[6] S. C. EISENSTAT, M. H. SCHULTZ AND A. H. SHERMAN, *Application of sparse matrix methods to partial differential equations*, in *Advances in Computer Methods for Partial Differential Equations—I*, R. Vichnevetsky, ed., AICA, New Brunswick, NJ, 1975, pp. 40–45.

[7] S. C. EISENSTAT, M. H. SCHULTZ AND A. H. SHERMAN, *Considerations in the design of software for sparse Gaussian elimination*, in *Sparse Matrix Computations*, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 263–274.

[8] G. E. FORSYTHE AND C. B. MOLER, *Computer Solution of Linear Algebraic Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[9] J. A. GEORGE AND J. W.-H. LIU, *An optimal algorithm for symbolic factorization of symmetric matrices*, Res. rep. CS-78-11, Department of Computer Science, University of Waterloo, Ontario, 1978.

[10] J. A. GEORGE AND J. W.-H. LIU, *User guide for SPARSPAK: Waterloo sparse linear equations package*, Res. rep. CS-78-30, Dept. of Computer Science, University of Waterloo, Ontario, 1978.

[11] 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.

[12] N. MUNSGAARD, *New factorization codes for sparse, symmetric and positive definite matrices*, BIT 19 (1979), pp. 43–52.

[13] D. J. ROSE, A. H. SHERMAN, R. E. TARJAN AND G. F. WHITTEN, *Algorithms and software for in-core factorization of sparse symmetric positive definite matrices*. Comput. & Structures 11 (1980), pp. 597–608.

[14] D. J. ROSE, R. E. TARJAN AND G. S. LUEKER, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput. 5 (1976), pp. 266–283.

[15] D. J. ROSE AND G. F. WHITTEN, private communication.

[16] A. H. SHERMAN, *On the efficient solution of sparse systems of linear and nonlinear equations*, doctoral dissertation, Department of Computer Science, Yale University, New Haven, CT, 1975.

[17] P. T. WOO, S. C. EISENSTAT, M. H. SCHULTZ AND A. H. SHERMAN, *Application of sparse matrix techniques to reservoir simulation*, in *Sparse Matrix Computations*, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 427–438.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 2, June 1981

0196-5204/81/0202-0011 $01.00/0

ON CERTAIN PARALLEL TOEPLITZ LINEAR SYSTEM SOLVERS*

J. GRCAR† AND A. SAMEH†

Abstract. In this paper we describe three algorithms for solving positive-definite banded Toeplitz systems of linear equations on parallel computers. Assuming we have $4mn$ processors, where n is the order of the system and m is the number of super- or subdiagonals, each system may be solved in $O(m \log n)$ time steps. Numerical experiments that compare the behavior of these algorithms in solving pentadiagonal Toeplitz systems are presented.

Key words. circulant matrix, Hurwitz factorization, parallel computers, Toeplitz matrix

1. Introduction. Many problems in mathematical physics and statistics give rise to Toeplitz or block-Toeplitz (or the related Hankel or block-Hankel) systems of linear equations. Some examples are: problems involving convolutions, integral equations with difference kernels, least squares approximations by polynomials, and stationary time series. Several sequential algorithms have been developed for the inversion of Toeplitz and Hankel matrices of order n, all requiring $O(n^2)$ arithmetic operations. See, for example, [Tren64], [Tren65], [Bare69], [Zoha69], [Phil69], [Just74], and [Riss74]. The extension of the above work to block-Toeplitz and block-Hankel matrices of order np with blocks of order p shows that the inverse can be obtained in $O(p^3n^2)$ operations; see [Wats73] and [Riss73]. An excellent survey on inverses of Toeplitz operators is given in [KaVM78]. Recently, some new algorithms have been developed that compute the inverse of a dense Toeplitz matrix even faster, $O(n \log^2 n)$ operations.[1] See [GuYu79], [BrGY80] and [Morf80]. These algorithms, and one of ours, use a doubling or divide-and-conquer strategy, a strategy that is exploited in many parallel algorithms [Hell78]. Our main interest here, however, is the solution of banded Toeplitz linear systems of equations of bandwidth $(m+1)$, where the order n is much larger than m. These systems arise in the numerical solution of certain initial and/or boundary-value problem using finite difference approximation. The fastest sequential algorithm [MoKa77] requires $O(n \log m)$ arithmetic operations for solving one such system of linear equations. For large n, this is rather expensive, especially when one has to solve these Toeplitz systems repeatedly as in time-dependent problems. In this paper we present three algorithms for solving banded Toeplitz systems on parallel computers which require far less time than the classical sequential algorithms. Essentially, we show that a positive-definite banded Toeplitz system may be solved in $O(m \log n)$ time steps, provided we have $4mn$ processors. This time accounts only for the arithmetic operations, where we have assumed that each of the four arithmetic operations consumes one time step. In view of the recent availability of very large-scale integrated circuits, "computer on a chip" microprocessors, and large semiconductor memory chips, parallel computers with a large number of processors may become feasible in the not-too-distant future. The parallel algorithms presented below, however, may be modified for a parallel computer with fewer processors. In our discussion of these algorithms in § 3, we ignore the time consumed in data handling. Assuming, however, that the parallel computer under consideration possesses an alignment network which connects the processors and the memory in such a way as to maximize conflict-free access of data to the various processors, we claim that the time required by the

* Received by the editors February 26, 1980, and in final form January 29, 1981.

† Department of Computer Science, University of Illinois, Urbana, IL 61801. This work was supported in part by the National Science Foundation under Grant No. US NSF MCS79-18394.

[1] Throughout this paper $\log n \equiv \lceil \log_2 n \rceil$.

arithmetic operation is the dominant portion of the whole time necessary for the completion of each of the three algorithms. Such a claim is not unreasonable in view of the fact that each banded Toeplitz matrix can be described by at most $(2m+1)$ numbers.

Throughout the paper we adopt the notational conventions of Householder [Hous64]. So, except for dimensions and indices, lowercase Greek letters represent scalars, lowercase Latin letters represent column vectors, and uppercase Greek or Latin letters represent matrices.

2. Preliminaries. The algorithms presented in the following section rely on several facts concerning Toeplitz matrices and the closely related circulant matrices. In this section, we review some of the main results concerning such matrices. We also present some of the basic parallel algorithms that are used in § 3.

DEFINITION 1. Let $A=[\alpha_{ij}]\in\mathbb{R}^{n\times n}$. Then A is *Toeplitz* if $\alpha_{ij}=\alpha_{i-j}$, $i,j=1,2,\cdots,n$. In other words, the entries of A along each diagonal are the same.

DEFINITION 2 [Zoha69]. Let $A, E_n\in\mathbb{R}^{n\times n}$, where $E_n=[e_n, e_{n-1},\cdots,e_1]$, in which e_i is the ith column of the identity matrix I_n. A is then said to be *persymmetric* if $E_nAE_n=A^T$; in other words, if A is symmetric about the cross-diagonal.

DEFINITION 3 [Aitk39]. Let $A\in\mathbb{R}^{2n\times 2n}$. Hence, A is called *centrosymmetric* if it can be written as

$$A=\begin{bmatrix} B & C \\ C^T & E_nBE_n \end{bmatrix}$$

where $B, C\in\mathbb{R}^{n\times n}$, $B=B^T$, and $C^T=E_nCE_n$. Equivalently, $E_{2n}AE_{2n}=A$.

LEMMA 1 [Zoha69]. *Let $A\in\mathbb{R}^{n\times n}$ be a nonsingular Toeplitz matrix. Then, both A and A^{-1} are persymmetric; i.e., $E_nAE_n=A^T$, and $E_nA^{-1}E_n=A^{-T}$.*

LEMMA 2 [CaBu76]. *Let $A\in\mathbb{R}^{2n\times 2n}$ be a symmetric Toeplitz matrix of the form*

$$A=\begin{bmatrix} B & C \\ C^T & B \end{bmatrix}.$$

Thus, A is also centrosymmetric. Furthermore, if

$$P_{2n}=\frac{1}{\sqrt{2}}\begin{bmatrix} I_n & E_n \\ I_n & -E_n \end{bmatrix}$$

we have

$$P_{2n}AP_{2n}^T=\begin{bmatrix} B+CE_n & 0 \\ 0 & B-CE_n \end{bmatrix}.$$

The proof is a direct consequence of Definition 3 and Lemma 1.

THEOREM 1 [GoFe74], [KaVM78]. *Let $A\in\mathbb{R}^{n\times n}$ be a Toeplitz matrix with all its leading principal submatrices being nonsingular. Let also, $Au=\alpha e_1$ and $Av=\beta e_n$, where*

$$u=(1,\mu_1,\mu_2,\cdots,\mu_{n-1})^T,$$
$$v=(\nu_{n-1},\nu_{n-2},\cdots,\nu_1,1)^T.$$

Since A^{-1} is persymmetric, then $\alpha=\beta$, and A^{-1} can be written as

$$\alpha A^{-1}=UV-\tilde{V}\tilde{U} \tag{2.1}$$

in which the right-hand side consists of the triangular Toeplitz matrices

$$U=\begin{bmatrix} 1 & & & & \\ \mu_1 & 1 & & & \\ \vdots & \ddots & \ddots & & \\ \mu_{n-2} & & \ddots & \ddots & \\ \mu_{n-1} & \mu_{n-2} & \cdots & \mu_1 & 1 \end{bmatrix}, \quad V=\begin{bmatrix} 1 & \nu_1 & \cdots & \nu_{n-2} & \nu_{n-1} \\ & \ddots & & & \nu_{n-2} \\ & & \ddots & & \vdots \\ & & & \ddots & \nu_1 \\ & & & & 1 \end{bmatrix}$$

$$\tilde{V}=\begin{bmatrix} 0 & & & & \\ \nu_{n-1} & \ddots & & & \\ \vdots & \ddots & \ddots & & \\ \nu_2 & & \ddots & \ddots & \\ \nu_1 & \nu_2 & \cdots & \nu_{n-1} & 0 \end{bmatrix}, \quad \tilde{U}=\begin{bmatrix} 0 & \mu_{n-1} & \cdots & \mu_2 & \mu_1 \\ & \ddots & & & \mu_2 \\ & & \ddots & & \vdots \\ & & & \ddots & \mu_{n-1} \\ & & & & 0 \end{bmatrix}$$

This interesting formula was first given by Gohberg and Semencul [GoSe72]. It shows that A^{-1} is completely determined by its first and last columns. Using (2.1), one may compute any selected element of A^{-1}. Cancellation, however, is assured if one attempts to compute an element of A^{-1} below the cross-diagonal. Since A^{-1} is persymmetric, this situation is remedied by computing the corresponding element about the cross-diagonal. Moreover, if A is symmetric, its inverse is both symmetric and persymmetric, and $u = E_n v$ completely determines A^{-1} via

$$\alpha A^{-1} = UU^T - \tilde{U}^T\tilde{U}. \tag{2.2}$$

Finally, if A is a lower triangular Toeplitz matrix, then $A^{-1} = \alpha^{-1}U$.

DEFINITION 4. Let $A \in \mathbb{R}^{n\times n}$ be a banded symmetric Toeplitz matrix with elements $\alpha_{ij} = \alpha_{|i-j|} = \alpha_k$, $k = 0, 1, \cdots, m$, where $m < n$, $\alpha_m \neq 0$ and $\alpha_k = 0$ for $k > m$. Then the complex rational polynomial

$$\phi(\xi) = \alpha_m\xi^{-m} + \cdots + \alpha_1\xi^{-1} + \alpha_0 + \alpha_1\xi + \cdots + \alpha_m\xi^m \tag{2.3}$$

is called the *symbol of* A [GoLe78].

LEMMA 3 [Part 62]. *Let A be as in Definition 4. Then A is positive definite for all $n > m$ if and only if:*

(i) $\phi(e^{i\theta}) = \alpha_0 + 2\sum_{k=1}^{m} \alpha_k \cos k\theta \geqq 0$ *for all θ, and*

(ii) $\phi(e^{i\theta})$ *is not identically zero.*

Note that condition (ii) is readily satisfied since $\alpha_m \neq 0$.

THEOREM 2 [FGHL74]. *Let A in Definition 4 be positive semi-definite for all $n > m$. Then the symbol $\phi(\xi)$ can be factored as*

$$\phi(\xi) = \psi(\xi)\psi(1/\xi) \tag{2.4}$$

where

$$\psi(\xi) = \beta_0 + \beta_1\xi + \cdots + \beta_m\xi^m \tag{2.5}$$

is a real polynomial with $\beta_m \neq 0$, $\beta_0 > 0$, and with no roots strictly inside the unit circle.

This is a theorem due to Fejer and Riesz [RiNa55], see also [Szeg59] and [GrSz58]. The factor $\psi(\xi)$ is called the Hurwitz factor. Such a factorization arises mainly in the time series analysis and the realization of dynamic systems, see [Robi67], [Wils69],

[RiKa72], and [Loew77]. The relevance of the Hurwitz factor to the problem at hand will be apparent from the following theorem.

THEOREM 3 [GoLe78], [Baue55], and [Baue56]. *Let* $\phi(e^{i\theta})>0$, *that is, A is positive definite and the symbol* (2.3) *has no roots on the unit circle. Consider the Cholesky factorization*

$$A = LL^T \tag{2.6}$$

where,

$$L = \begin{bmatrix} \lambda_0^{(1)} & & & & & & \\ \lambda_1^{(2)} & \lambda_0^{(2)} & & & & & \\ \lambda_2^{(3)} & \lambda_1^{(3)} & \lambda_0^{(3)} & & & & \\ \vdots & \vdots & \vdots & & & & \\ \lambda_m^{(m+1)} & \lambda_{m-1}^{(m+1)} & \lambda_{m-2}^{(m+1)} & \cdots & \lambda_0^{(m+1)} & & \\ & \lambda_m^{(m+2)} & \lambda_{m-1}^{(m+2)} & \cdots & \lambda_1^{(m+2)} & \lambda_0^{(m+2)} & \\ & & \ddots & & \ddots & \ddots & \ddots \\ 0 & & \lambda_m^{(k)} & \lambda_{m-1}^{(k)} & & \lambda_1^{(k)} & \lambda_0^{(k)} \end{bmatrix} \tag{2.7}$$

Then,

$$\lim_{k\to\infty} \lambda_j^{(k)} = \beta_j, \qquad 0 \leqq j \leqq m,$$

where β_j *is given by* (2.5). *In fact, if* τ *is that root of* $\psi(\xi)$ *closest to the unit circle* (*note that* $|\tau|>1$) *with multiplicity p, we have*

$$\lambda_j^{(k)} = \beta_j + O[(k-j)^{2(p-1)}/|\tau|^{2(k-j)}]. \tag{2.8}$$

This theorem shows that convergence of the rows of the Cholesky factor L is linear with the asymptotic convergence factor depending on the magnitude of that root of $\psi(\xi)$ closest to the unit circle. The larger this magnitude, the faster is the convergence.

We turn now to circulant matrices that play an important role in one of the algorithms in § 3. In particular, we consider these circulants associated with banded symmetric or nonsymmetric Toeplitz matrices.

Let $A=[\alpha_{-m}, \cdots, \alpha_{-1}, \alpha_0, \alpha_1, \cdots, \alpha_m] \in \mathbb{R}^{n\times n}$ be a banded Toeplitz matrix of bandwidth $(m+1)$ where $2m+1 \leqq n$, and α_m, $\alpha_{-m} \neq 0$. We write this matrix as

$$A = \begin{bmatrix} B & C & & & & & \\ D & B & C & & & & \\ & D & B & C & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & D & B & C \\ & & & & & D & B \end{bmatrix}, \tag{2.9}$$

where B, C and $D \in \mathbb{R}^{m\times m}$ are the Toeplitz matrices

$$B = \begin{bmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{m-1} \\ \alpha_{-1} & \alpha_0 & & \vdots \\ \vdots & & \ddots & \alpha_1 \\ \alpha_{-m+1} & & \alpha_{-1} & \alpha_0 \end{bmatrix}, \quad C = \begin{bmatrix} \alpha_m & & & \\ \alpha_{m-1} & \ddots & & 0 \\ \vdots & \ddots & \ddots & \\ \alpha_1 & \cdots & \alpha_{m-1} & \alpha_m \end{bmatrix},$$

$$\text{and} \quad D = \begin{bmatrix} \alpha_{-m} & \alpha_{-m+1} & \cdots & \alpha_{-1} \\ & \alpha_{-m} & & \\ & \ddots & & \\ & 0 & \ddots & \alpha_{-m+1} \\ & & & \ddots \ \alpha_{-m} \end{bmatrix}.$$

The circulant matrix $\tilde{A}$ corresponding to A is therefore given by

$$\tilde{A} = \begin{bmatrix} B & C & & & & D \\ D & B & C & & 0 & \\ & & \ddots & \ddots & & \\ & 0 & & D & B & C \\ C & & & & D & B \end{bmatrix} \tag{2.10}$$

which may also be written as the matrix polynomial

$$\tilde{A} = \sum_{j=0}^{m} \alpha_j K^j + \sum_{j=1}^{m} \alpha_{-j} K^{n-j}$$

where

$$K = \begin{bmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ 1 & & & & 0 \end{bmatrix}. \tag{2.11}$$

It is of interest to note that $K^T = K^{-1} = K^{n-1}$, and $K^n = I_n$.

LEMMA 4 [Davi79]. *Let W be the unitary Fourier matrix of order n,*

$$W = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{(n-1)} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)^2} \end{bmatrix} \tag{2.12}$$

in which $\omega = e^{i(2\pi/n)}$ is an nth root of unity. Then,

$$\begin{aligned} W^*KW &= \Omega \\ &= \operatorname{diag}(1, \omega, \omega^2, \cdots, \omega^{n-1}), \end{aligned} \tag{2.13}$$

and

$$W^*\tilde{A}W = \Gamma = \sum_{j=0}^{m} \alpha_j \Omega^j + \sum_{j=1}^{m} \alpha_{-j}\Omega^{-j} \tag{2.14}$$

where $\Gamma = \operatorname{diag}(\gamma_0, \gamma_1, \cdots, \gamma_{n-1})$.

In other words, the kth eigenvalue of $\tilde{A}$, $k = 0, 1, \cdots, n-1$, is given by

$$\gamma_k = \tilde{\phi}(\omega^k) \tag{2.15}$$

where

$$\tilde{\phi}(\xi) = \sum_{j=-m}^{m} \alpha_j \xi^j$$

is the symbol of the nonsymmetric Toeplitz matrix (2.9), or

$$\gamma_k = \sqrt{n}\, e_{k+1}^T W a, \tag{2.16}$$

in which $a^T = e_1^T \tilde{A}$.

Most of the fundamental parallel algorithms needed in § 3 may be found in the survey by Heller [Hell78], or in [Same77]. For the sake of self-containment, we present a brief summary of such results.

LEMMA 5. *Let $x, y \in \mathbb{R}^n$, then the inner product $x^T y$ can be evaluated in $1 + \log n$ time steps using n processors.*

LEMMA 6. *Let $L \in \mathbb{R}^{n\times n}$ be lower triangular. The linear system $Lx = f$ can therefore be solved in time $3(n-1)$ using only $(n-1)$ processors, or in $3 + \frac{1}{2}(\log n)(3 + \log n)$ time steps provided that $O(n^3)$ processors are available.*

LEMMA 7. *If L is Toeplitz lower triangular, then $Lx = f$ can be solved in $\log^2 n + O(\log n)$ using only $n^2/4$ processors.*

LEMMA 8. *Let $L \in \mathbb{R}^{n\times n}$ be lower triangular, Toeplitz, and banded with bandwidth $m+1$, i.e., $\lambda_k = 0$ for $k = |i-j| > m$. Then $Lx = f$ and $Ly = e_1$ may be solved in time less than $(3 + 2\log m)\log n$ using $\lceil 3/4\, mn \rceil$ processors.*

THEOREM 4. *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular. Using Gaussian elimination without pivoting, the triangular factorization $A = LU$ may be obtained in time $3(n-1)$ using no more than n^2 processors. If partial pivoting is used, however, the factorization requires time $O(n \log n)$ using the same number of processors.*

THEOREM 5 [Peas68]. *Let $W \in \mathbb{C}^{n\times n}$ be as in Lemma 4, $y \in \mathbb{C}^n$, and n is a power of 2. Then the product Wy may be obtained in $3 \log n$ time steps using only $2n$ processors.*

3. Computational Schemes. In this section we present three algorithms for solving the Toeplitz linear system

$$Ax = f \tag{3.1}$$

where A is given by (2.9) and $n \gg m$. All three algorithms are suitable for parallel computation. We state clearly the conditions under which each algorithm is applicable, and give upper bounds on the time and number of processors required by each algorithm.

The first algorithm requires the least time of all three, $6 \log n + O(1)$. It is applicable only when the corresponding circulant matrix is nonsingular. The second algorithm may be used if A, in (3.1), is positive definite or if all its principal minors are nonzero. It solves (3.1) in $O(m \log n)$ time steps. The third algorithm uses a modification of the second algorithm to compute the Hurwitz factorization of the

symbol $\phi(\xi)$ of a positive definite Toeplitz matrix. The Hurwitz factor, in turn, is then used by an algorithm proposed by Fischer, et al. [FGHL74] to solve (3.1). This last algorithm is applicable only if none of the roots of $\phi(\xi)$ lie on the unit circle. In fact, the root of the factor $\psi(\xi)$ nearest to the unit circle should be far enough away to assure early convergence of the modified Algorithm 2. The third algorithm requires time $O(\log m \log n)$ provided that the Hurwitz factor has already been computed. It also requires the least storage of all three algorithms.

A fourth algorithm that could be included here is the specialization of the block cyclic reduction methods developed by Swarztrauber [Swar74] and Heller [Hell76]. Swarztrauber stabilizes the reduction by the Buneman method [Bune69], while Heller identifies classes of matrices for which the reduction is naturally stable; among these are diagonally dominant matrices. The numerical experiments of § 4 include comparisons with Heller's algorithm.

ALGORITHM 1. Let A be that banded nonsymmetric Toeplitz matrix given by (2.9). We can write the linear system (3.1) as

$$(\tilde{A}-S)x=f \tag{3.2}$$

where $\tilde{A}$ is the circulant matrix (2.10) associated with A, and

$$S=U\begin{bmatrix}0 & D\\ C & 0\end{bmatrix}U^T,$$

in which

$$U^T=\begin{bmatrix}I_m & 0 & \cdots & 0 & 0\\ 0 & 0 & \cdots & 0 & I_m\end{bmatrix}.$$

If $\tilde{A}$ is nonsingular, the Woodbury formula [Hous64] yields the solution

$$x=\tilde{A}^{-1}f-\tilde{A}^{-1}UG^{-1}U^T\tilde{A}^{-1}f, \tag{3.3}$$

where

$$G=U^T\tilde{A}^{-1}U-\begin{bmatrix}0 & D\\ C & 0\end{bmatrix}^{-1}.$$

Now, the computation of x can be performed in four stages.

Stage 1. In this stage we determine whether $\tilde{A}$ is nonsingular and, if so, we determine $\tilde{A}^{-1}$ and $y=\tilde{A}^{-1}f$. Since the inverse of a circulant matrix is also a circulant, $\tilde{A}^{-1}$ is completely determined by solving $\tilde{A}v_1=e_1$. This will be done via (2.14), i.e., $y=W\Gamma^{-1}W^*f$ and $v_1=W\Gamma^{-1}W^*e_1$. Such a computation is organized as follows:

(i) Simultaneously form $\sqrt{n}Wa$ (see (2.16)), and W^*f in 3 log n time steps using $4n$ processors, see Theorem 5 (FFT). This is an inexpensive test for the nonsingularity of $\tilde{A}$. If none of the elements of $\sqrt{n}Wa$ (eigenvalues of $\tilde{A}$) vanish, we proceed to step (ii).

(ii) Using $2n$ processors, obtain $\Gamma^{-1}(W^*e_1)$ and $\Gamma^{-1}(W^*f)$ in one time step. Note that $\sqrt{n}W^*e_1=(1, 1, \cdots, 1)^T$.

(iii) $v_1=W(\Gamma^{-1}W^*e_1)$ and $y=W(\Gamma^{-1}W^*f)$ are then computed via the FFT (Theorem 5) in 3 log n time steps using $4n$ processors.

Therefore, this stage requires time $(1+6\log n)$ employing no more than $4n$ processors.

Stage 2. Here we wish to solve the linear system

$$Gz=U^Ty=\begin{bmatrix}y_1\\ y_v\end{bmatrix} \tag{3.4}$$

where $y_1, y_\nu \in \mathbb{R}^m$ contain the first and last m elements of y, respectively. First, we have to form the matrix $G \in \mathbb{R}^{2m \times 2m}$. From Stage 1, v_1 completely determines $\tilde{A}^{-1}$, in particular we have

$$U^T \tilde{A}^{-1} U = \begin{bmatrix} F & M \\ N & F \end{bmatrix},$$

where F, M, and N are Toeplitz matrices each of order m. Now, G is given by

$$G = \begin{bmatrix} F & M - C^{-1} \\ N - D^{-1} & F \end{bmatrix}.$$

Recalling that C^{-1} and D^{-1} are completely determined by their first columns $C^{-1}e_1$ and $D^{-1}e_1$, the two inverses are simultaneously determined in time $O(\log^2 m)$ with $m^2/2$ processors, see Lemma 7. After one subtraction, (3.4) is solved in $O(m \log m)$ time steps employing $4m^2$ processors, see Theorem 4 and Lemma 6.

Stage 3. The n-vector $u = \tilde{A}^{-1} U z$ is trivially obtained in $O(m)$ time steps using $2n$ processors.

Stage 4. Finally, using n processors, the solution of (3.1) may be obtained in one time step, $x = y - u$.

The time required by this algorithm, which is clearly dominated by the first stage, is $6 \log n + O(m \log m)$ employing no more than $4n$ processors. The corresponding sequential algorithm consumes $O(n \log n)$ time steps, hence we obtain $O(n)$ speedup, indicating very effective usage of our $4n$ processors. It may also be of interest to point out that at no time do we need more than $2n + O(m^2)$ storage locations.

The notion of inverting a Toeplitz matrix via the correction of the corresponding circulant-inverse is certainly not new. It has been used, for example, in some of the sequential algorithms in [FGHL74] and [MoKa77].

Such an algorithm, when applicable, is very attractive on a parallel computer with the appropriate switching network needed for the fast Fourier transforms. Given $4n$ processors, the time required for solving (3.1) is so inexpensive that one is tempted to improve the solution via one step of iterative refinement. Since v_1 is already available, each step of iterative refinement requires only $2n$ processors. The desirability of iterative refinement for this algorithm is discussed in more detail in § 4.

ALGORITHM 2. Here, we assume that the banded Toeplitz matrix A in (2.9) is positive definite. This implies that $D = C^T$, and B is positive definite. For the sake of ease of illustration, we assume that $n = 2^q p$, where p and q are integers with $p = O(\log n) \geqq m$. On a sequential machine, the Cholesky factorization is the preferred algorithm for solving the linear system (3.1). Unless the corresponding circulant matrix is also positive definite, however, the rows of L (the Cholesky factor of A) will not converge, see Theorem 3. This means that one would have to store $O(mn)$ elements, which may not be acceptable for relatively large m. Even if the corresponding circulant matrix is positive definite, Theorem 3 indicates that convergence can indeed be slow if the magnitude of that root of the Hurwitz factor $\psi(\xi)$ (see Theorem 2) nearest to the unit circle is only slightly greater than 1. Implementing the Cholesky factorization on a parallel computer requires $O(s)$ time steps using $O(m^2)$ processors, where s is that row of L at which convergence takes place. If $s \simeq n$, Cholesky factorization is definitely not suitable for a parallel computer, with ample number of processors, with regard to time or storage. In the following, we present an alternative parallel algorithm that solves the same positive definite system in time $O(m \log n)$ with $O(n)$ processors, which requires no more than $2n + O(m^2)$ temporary storage locations. This algorithm is also an

alternative to Algorithm 1 if the circulant matrix associated with A is only positive semidefinite, in which case Algorithm 1 fails.

Recalling our assumption that $n = 2^q p$, the algorithm consists of $(q+1)$ stages which we outline as follows.

Stage 0. Let the pth leading principal submatrix of A be denoted by A_0, and the right-hand side of (3.1) be partitioned as

$$f^T = (f_1^{(0)T}, f_2^{(0)T}, \cdots, f_\eta^{(0)T}),$$

where $f_i^{(0)} \in \mathbb{R}^p$ and $\eta = n/p = 2^q$. In this initial stage we simultaneously solve the $(\eta+1)$ linear systems

$$A_0 z_0 = e_1^{(p)}, \tag{3.5a}$$

$$A_0 y_i^{(0)} = f_i^{(0)}, \qquad i = 1, 2, \cdots, \eta, \tag{3.5b}$$

where $e_1^{(p)} \in \mathbb{R}^p$ is the first column of the identity I_p. From Theorem 4 and Lemma 6 we see that the above systems can be solved in $9(p-1)$ time steps using $m\eta$ processors.

Stage j. $(j = 1, 2, \cdots, q)$. Let

$$A_j = \left[\begin{array}{c|c|c} A_{j-1} & & 0 \\ & C & \\ \hline & C^T & \\ \hline 0 & & A_{j-1} \end{array}\right] \tag{3.6}$$

be the leading $2r \times 2r$ principal submatrix of A, where $r = 2^{j-1}p$. Also, let f be partitioned as

$$f^T = (f_1^{(j)T}, f_2^{(j)T}, \cdots, f_\nu^{(j)T})$$

where $f_i^{(j)} \in \mathbb{R}^{2r}$, and $\nu = n/2r$. Here, we simultaneously solve the $(\nu+1)$ linear systems

$$A_j z_j = e_1^{(2r)}, \tag{3.7a}$$

$$A_j y_i^{(j)} = f_i^{(j)}, \qquad i = 1, 2, \cdots, \nu, \tag{3.7b}$$

where stage $(j-1)$ has already provided us with $z_{j-1} = A_{j-1}^{-1} e_1^{(r)}$ and $y_i^{(j-1)} = A_{j-1}^{-1} f_i^{(j-1)}$, $i = 1, 2, \cdots, 2\nu$. Let us consider the time and number of processors required for solving the ith linear system in (3.7b). Observing that

$$f_i^{(j)} = \begin{bmatrix} f_{2i-1}^{(j-1)} \\ f_{2i}^{(j-1)} \end{bmatrix},$$

and premultiplying both sides of (3.7b) by the positive definite matrix

$$D_j = \operatorname{diag}(A_{j-1}^{-1}, A_{j-1}^{-1})$$

we obtain the linear system

$$\left[\begin{array}{c|c} I_r & G_j \\ \hline 0 \quad H_j & I_r \quad 0 \end{array}\right] [y_i^{(j)}] = \begin{bmatrix} y_{2i-1}^{(j-1)} \\ y_{2i}^{(j-1)} \end{bmatrix} \tag{3.8}$$

where

$$G_j = A_{j-1}^{-1} \begin{bmatrix} 0 \\ C \end{bmatrix} \quad \text{and} \quad H_j = A_{j-1}^{-1} \begin{bmatrix} C^T \\ 0 \end{bmatrix}.$$

From Lemma 1 we see that both A_{j-1}^{-1} and C are persymmetric, i.e., $E_r A_{j-1}^{-1} E_r = A_{j-1}^{-1}$ and $E_m C E_m = C^T$, thus $H_j = E_r G_j E_m$. Using the Gohberg–Semencul formula (2.2), see

Theorem 1, H_j may be expressed as follows. Let

$$u = \alpha z_{j-1}$$
$$= (1, \mu_1, \mu_2, \cdots, \mu_{r-1})^T,$$

and

$$\tilde{u} = J_r E_r u,$$

where $J_r = [e_2^{(r)}, \cdots, e_r^{(r)}, 0]$. Hence, H_j is given by

$$\alpha H_j = (YY_1^T - \tilde{Y}\tilde{Y}_1^T)C^T, \tag{3.9}$$

in which,

$$Y = [u, J_r u, \cdots, J_r^{m-1} u]_{(r\times m)},$$
$$\tilde{Y} = [\tilde{u}, J_r \tilde{u}, \cdots, \tilde{J}_r^{m-1} \tilde{u}]_{(r\times m)},$$

$$Y_1 = (I_m, 0)Y = \begin{bmatrix} 1 & & & \\ \mu_1 & 1 & & 0 \\ \mu_2 & \mu_1 & 1 & \\ \vdots & \cdot & \cdot & \cdot \\ \mu_{m-1} & \cdots & \mu_2 & \mu_1 \quad 1 \end{bmatrix},$$

and

$$\tilde{Y}_1 = (I_m, 0)\tilde{Y} = \begin{bmatrix} 0 & & & \\ \mu_{r-1} & 0 & & 0 \\ \mu_{r-2} & \mu_{r-1} & 0 & \\ \vdots & \cdot & \cdot & \cdot \\ \mu_{r-m+1} & \cdots & \mu_{r-2} & \mu_{r-1} \quad 0 \end{bmatrix}.$$

The matrix of coefficients in (3.8), $D_j^{-1}A_j$, may be written as

$$\begin{bmatrix} I_{r-m} & N_1^{(j)} & 0 \\ 0 & N_2^{(j)} & 0 \\ 0 & N_3^{(j)} & I_{r-m} \end{bmatrix}.$$

$N_2^{(j)}$ is clearly nonsingular since A_j is invertible. Note also that the eigenvalues of the center block $N_2^{(j)} \in \mathbb{R}^{2m\times 2m}$ are those eigenvalues of $(D_j^{-1}A_j)$ that are different from 1. Since $D_j^{-1}A_j$ is similar to the psoitive definite matrix $D_j^{-1/2}A_jD_j^{-1/2}$, $N_2^{(j)}$ is not only nonsingular but also has all its eigenvalues positive. Hence, the solution of (3.8) is trivially obtained if we first solve the middle $2m$ equations,

$$N_2^{(j)}h = g, \tag{3.10a}$$

or

$$\begin{bmatrix} I_m & E_m M_j E_m \\ M_j & I_m \end{bmatrix}\begin{bmatrix} h_{1,i} \\ h_{2,i} \end{bmatrix} = \begin{bmatrix} g_{1,i} \\ g_{2,i} \end{bmatrix}, \tag{3.10b}$$

where

$$M_j = (I_m, 0)H_j = \alpha^{-1}(Y_1Y_1^T - \tilde{Y}_1\tilde{Y}_1^T)C^T$$

and $g_{k,i}$, $h_{k,i}$, $k = 1, 2$, are the corresponding partitions of $f_i^{(j)}$ and $y_i^{(j)}$, respectively.

Observing that $N_2^{(j)}$ is centrosymmetric (see Definition 3), then from Lemma 2 we reduce (3.10b) to the two independent linear systems

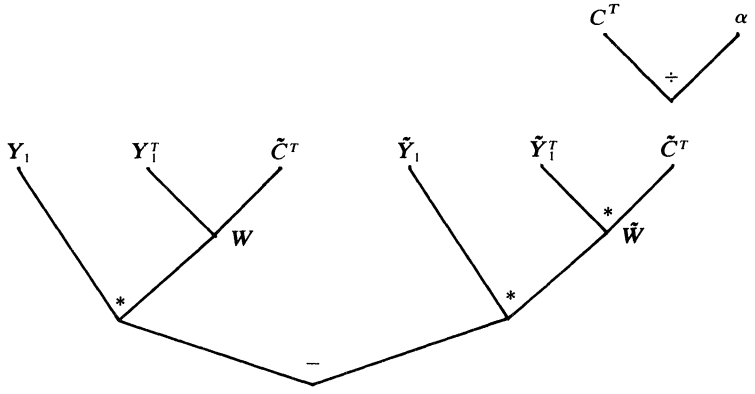
$$\begin{aligned}(I_m + E_m M_j)(h_{1,i} + E_m h_{2,i}) &= (g_{1,i} + E_m g_{2,i}),\\ (I_m - E_m M_j)(h_{1,i} - E_m h_{2,i}) &= (g_{1,i} - E_m g_{2,i}).\end{aligned} \tag{3.10c}$$

Since $P_{2m} N_2^{(j)} P_{2m}^T$ has all its leading principal minors positive, these two systems may be simultaneously solved using Gaussian elimination without partial pivoting. Once h_1 and $E_m h_2$ are obtained, $y_i^{(j)}$ is readily available.

$$y_i^{(j)} = \begin{bmatrix} y_{2i-1}^{(j)} - E_r H_j E_m h_{2,i} \\ y_{2i}^{(j-1)} - H_j h_{1,i} \end{bmatrix}. \tag{3.11}$$

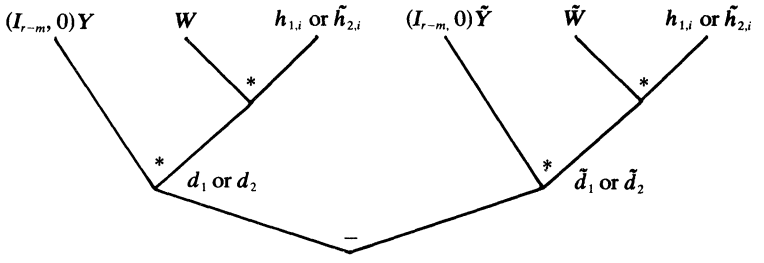
The computations to be performed in this stage may be organized as follows:

(a) Since u and $\tilde{u}$ are readily available from stage $j-1$, M_j may be computed in time $4+2\log m$ (see Lemma 5), using no more than m^3 processors (see Fig. 1(a)).



Level	1	2	3	4
Time	1	$1+\log m$	$1+\log m$	1
Processors	$<m^2$	$<m^3$	$<m^3$	m^2

FIG. 1(a)



Level	1	2	3
Time	$1+\log m$	$1+\log m$	1
Processors	$m(m+1)$	$2m(r-m)$	$r-m$

FIG. 1(b)

(b) The two linear systems (3.10c) may be solved for $h_{1,i}$ and $\tilde{h}_{2,i} = E_m h_{2,i}$ in time $3m + \frac{1}{2}\log^2 m + O(\log m)$ using less than m^3 processors (see Theorem 4 and Lemma 6).

(c) From (3.9), one can simultaneously obtain the top $(r-m)$ elements of the two r-vectors $H_j h_{1,i}$ and $H_j \tilde{h}_{2,i}$ in time $3+2\log m$ using $4m(r-m)$ processors (see Fig. 1(b)).

(d) Finally, the rest of the elements of $y_i^{(j)}$ are obtained via (3.11) in one time step using $2(r-m)$ processors.

Thus, using no more than $4mr$ processors, one system in (3.7b) may be solved in $3m + \frac{1}{2}\log^2 m + O(\log m)$ time steps. Since we have $(\nu+1)$ such systems in (3.7a) and (3.7b), the jth stage requires the same time above, provided we have $4m(r-m)\times(\nu+1) < 2mn + 4mr$ processors. Consequently, the whole algorithm consumes time $3m \log n + O(\log^2 m \log n)$ employing no more than $4mn$ processors. In fact, the number of processors may be reduced to $O(n)$ with the time remaining $O(m \log n)$. This can be attained if we assign only one processor to each inner product at level 2 of Figure 1(b) rather than m processors. In this case d_i and $\tilde{d}_i$, $i = 1, 2$, are computed in $(2m-1)$ time steps employing $4(r-m)$ processors. Hence, using only $4n$ processors, the above algorithm requires time $5m \log n + O(\log^2 m \log n)$. The storage required in both versions of Algorithm 2 is only $2n + O(m^2)$. Faced with another right-hand side in (3.1), we can reduce the number of processors required by Algorithm 2 by one-half, provided that we store the vectors z_j, $j = 1, 2, \cdots, q-1$ (see (3.5a) and (3.7a)). This, however, will increase the required storage to $3n + O(m^2)$.

We would like to point out that Algorithm 2 is not restricted to positive definite systems. It may be used to solve any nonsymmetric matrix whose leading principal submatrices of order $2^k p$, $k = 0, 1, \cdots, q$, are nonsingular. In such a case, however, we need not only compute the first column of the inverse of each A_j, but also the last column of the inverse. This will allow us to make use of the Gohberg–Semencul formula to conveniently express the matrices G_j and H_j. The time and the number of processors required by the algorithm will remain $O(m \log n)$ and $O(n)$, respectively.

Such nonsymmetric Toeplitz systems arise, for example, from the finite difference approximation of the Korteweg–de Vries equation

$$u_t + uu_x + \delta^2 u_{xxx} = 0$$

using the scheme proposed by Kreiss and Widlund [KrWi67], see also [Buck74] and [Buck77]. In this case, one has to solve nonsymmetric Toeplitz systems at each time step, Δt, where the banded matrix of coefficients is given by

$$A = I + S$$

in which $m = 2$ and S is skew symmetric. Since all the eigenvalues of S are pure imaginary, all the leading principal submatrices of A are nonsingular and we can use Algorithm 2. Since m is small, we can further reduce the time required for the repeated solution of such systems by forming and storing the matrices H_j, $j = 1, 2, \cdots, q-1$. Note that $G_j = -E_r H_j E_m$. It is not difficult to show that, by computing the matrices H_j explicitly, the time required to solve the linear system for the first time is $14 \log n$ provided we use $6n$ processors. Now, allocating $4n + O(\log n)$ storage locations for the matrices H_j, the first and last columns of A_j^{-1}, and the LU-factorization of M_j, $j = 1, 2, \cdots, q-1$, each subsequent linear system is solved in time $6 \log n$ using only $2n$ processors. Algorithm 1, which may also be used in solving these systems, requires essentially the same time, but double the number of processors.

ALGORITHM 3. Here, we present an algorithm that is applicable only to those banded Toeplitz positive definite matrices that have positive definite associated circulant matrices. In other words, we assume that both A in (2.9) and A in (2.10) are positive definite. Hence, from Lemma 4 and Theorem 3, row $\hat{\imath}$ of the Cholesky factor of A will converge to the coefficients of $\psi(\xi)$, where $\hat{\imath} < n$. Consider the Cholesky factorization (2.6), $A = LL^T$, where

$$L = \begin{bmatrix} S_1 & & & & & & & \\ V_1 & S_2 & & & & & & \\ & V_2 & S_3 & & & & & \\ & & \ddots & \ddots & & & & \\ & & & \ddots & \ddots & & & \\ & & & & V_{i-1} & S_i & & \\ & & & & & V & S & \\ & & & & & & V & S \\ & & & & & & & \ddots \end{bmatrix} \tag{3.12}$$

in which $\hat{\imath} = mi + 1$, S_j and $V_j \in \mathbb{R}^{m \times m}$ are lower and upper triangular, respectively, and S, V are Toeplitz and given by

$$S = \begin{bmatrix} \beta_0 & & & \\ \beta_1 & & 0 & \\ \vdots & \ddots & \ddots & \\ \beta_{m-1} & \cdots & \beta_1 & \beta_0 \end{bmatrix},$$

and

$$V = \begin{bmatrix} \beta_m & \beta_{m-1} & \cdots & \beta_1 \\ & \beta_m & \ddots & \vdots \\ & & \ddots & \beta_{m-1} \\ 0 & & & \beta_m \end{bmatrix}.$$

Equating both sides of (2.6), we get

$$B = SS^T + VV^T,$$

and

$$C = SV^T,$$

where A, B, and C are as given in (2.9). Now, A can be expressed as

$$A = R^T R + \begin{pmatrix} I_m \\ 0 \end{pmatrix} VV^T (I_m, 0) \tag{3.13}$$

where $R^T \in \mathbb{R}^{n \times n}$ is the Toeplitz lower triangular matrix

$$R^T = \begin{bmatrix} S & & & & \\ V & S & & & \\ & \ddots & \ddots & & \\ & & \ddots & \ddots & \\ & & & V & S \end{bmatrix} \tag{3.14}$$

Assuming that an approximate Hurwitz factor, $\tilde{\psi}(\xi) = \sum_{j=0}^{m} \tilde{\beta}_j \xi^j$, is obtained from a floating-point Cholesky factorization of A, i.e.,

$$\tilde{R}^T = \begin{bmatrix} \tilde{S} & & & & \\ \tilde{V} & \tilde{S} & & & \\ & \cdot & \cdot & & \\ & & \cdot & \cdot & \\ & & & \tilde{V} & \tilde{S} \end{bmatrix}$$

is available, Fischer et al. [FGHL74] have proposed to compute the solution of the linear system (3.1) via the Woodbury formula [Hous64] as

$$x = F^{-1}f - F^{-1}\begin{bmatrix} \tilde{V} \\ 0 \end{bmatrix} Q^{-1}[\tilde{V}^T, 0]F^{-1}f \tag{3.15}$$

where

$$F = \tilde{R}^T\tilde{R},$$

and

$$Q = I_m + [\tilde{V}^T, 0]F^{-1}\begin{bmatrix} \tilde{V} \\ 0 \end{bmatrix}.$$

Computing x in (3.15) may then be organized as follows:

(a) Solve the linear system $Fv = f$ via the forward and backward sweeps $\tilde{R}^T w = f$ and $\tilde{R}v = w$. Note that in Lemma 8 we obtain also $\tilde{R}^{-1}e_1$ which completely determines $\tilde{R}^{-1}$.

(b) Let $\tilde{L}_1 = \tilde{R}^{-T}\binom{I_m}{0}$ and $v_1 = (I_m, 0)v$, and simultaneously form $T = \tilde{L}_1\tilde{V}$ and $u = \tilde{V}^T v_1$. Now $Q = I_m + T^T T$ is easily computed.

(c) Solve the linear system $Qa = u$, see Theorem 4 and Lemma 6, and obtain the column vector $b = Ta$.

(d) Finally, solve the Toeplitz triangular system $\tilde{R}c = b$, and obtain the solution $x = v - c$.

From the relevant basic lemmas and theorems outlined in § 2 we see that, given the approximation to the Hurwitz factor $\tilde{R}$, (3.15) requires a time of $(10 + 6 \log m) \log n + O(m)$ steps using no more than mn processors, and only $n + O(m^2)$ storage locations, the lowest among all three algorithms.

Since the best known time for obtaining $\tilde{S}$ and $\tilde{V}$ via the Cholesky factorization of A on a parallel computer with no less than m^2 processors is $O(\hat{\imath})$, the cost of computing the Hurwitz factorization can indeed be very high unless $\hat{\imath} \ll n$. This will happen only if the Hurwitz factor $\psi(\xi)$ has its nearest root to the unit circle of magnitude well above 1.

In what follows, we present an attractive alternative to the Cholesky factorization. A simple modification of Algorithm 2 yields a very efficient method for computing the Hurwitz factor. Let $\tilde{r} = 2^k p \geqq m(i+1) = \hat{\imath} + m - 1$. Then, from (2.6), (3.8)–(3.12) we see that

$$E_m M_k E_m = S^{-T}S^{-1}C. \tag{3.16}$$

In other words, the matrices M_j in (3.10b) converge to a matrix M (say), and the elements $\tilde{\beta}_j$, $0 \leqq j \leqq m-1$, of $\tilde{S}$ are obtained by computing the Cholesky factorization of

$$E_m C^T M^{-1} E_m = \left[(0, I_m)A_{j-1}^{-1}\begin{pmatrix} 0 \\ I_m \end{pmatrix}\right]^{-1}.$$

Now, the only remaining element of $\tilde{V}$, namely $\tilde{\beta}_m$, is easily obtained as $\alpha_m/\tilde{\beta}_0$, which can be verified from the relation $C = SV^T$.

Observing that we need only to compute the matrices M_j, Algorithm 2 may be modified so that in any stage j, we solve only the linear systems $A_j z_j = e_1^{(2r)}$, where $j = 0, 1, 2, \cdots$, and $2r = 2^{j+1}p$. Since we assume that convergence takes place in the kth stage, where $\tilde{r} = 2^k p \leqq n/2$, the time required for computing the Hurwitz factor is approximately $6mk \leqq 6m \log (n/2p)$ using $2m\tilde{r} \leqq mn$ processors.

4. Numerical experiments. The above parallel algorithms have been tested using positive definite Toeplitz systems of linear equations with bandwidths of 3, that is, with $m = 2$. Lemma 3 states that all sufficiently large Toeplitz matrices having diagonal values $(1, \sigma, \delta, \sigma, 1)$ are positive definite provided that the symbol function $\phi(e^{i\theta}) = \delta + 2\sigma \cos\theta + 2\cos 2\theta$ has no negative values. For positive δ, this condition is satisfied when (σ, δ) corresponds to a point on or above the lower curve in Figure 2. The matrix is

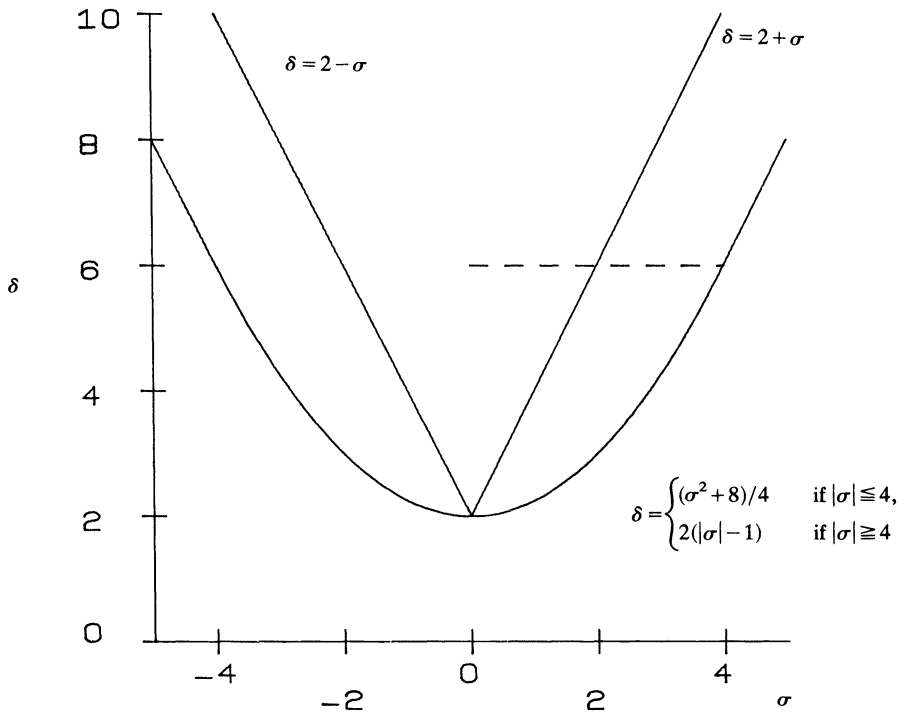


FIG. 2

diagonally dominant when the point lies above the upper curve. The test matrices have $\delta = 6$ and $0 \leqq \sigma \leqq 4$. The points corresponding to these matrices lie on the dashed line in Figure 2. Examination of the proof of Lemma 3 reveals that as the order of the positive definite Toeplitz matrix increases, the convex hull of its eigenvalues approaches the set of all values of the symbol function. Thus, the formula $8/(4-\sigma)$ closely bounds the 2-norm condition numbers of the matrices used in the tests.

The order of the test matrices is 4096, this favors no one algorithm over the others. For example, the fast Fourier transform used in Algorithm 1 is genuinely fast only when applied to vectors whose length is a power of two. Algorithm 2 also prefers an order

divisible by a large power of two. However, it can be modified to accept any size problem by partitioning the solution vector into pieces that are not of one uniform size.

The above algorithms are also compared with the block cyclic reduction algorithm, a method which adapts well to parallel computation. Heller [Hell76] formulates the algorithm in such a way as to allow submatrix blocks of varying dimension. In this way, the method adapts to systems of linear equations whose blocked order is not divisible by one less than a large power of two. Such a variable dimension algorithm requires a storage scheme for the blocks that is more complex than the in-place partitioning of vectors used to extend Algorithm 2 to problems of arbitrary order. For this reason, the test matrices for the block cyclic reduction algorithm have order 4094.

Table 1 presents the parallel time, processor requirements, and the number of arithmetic operations of the various algorithms for the pentadiagonal test matrices of

TABLE 1
Time, processor, and operation counts.

	Time	Processors	Operations	Storage
Algorithm 1	$6 \log n$	$2n$	$10n \log n$	$2n$
* Algorithm 2	$18 \log n$	$4n$	$4n \log n$	$2n$
† Algorithm 3	$16 \log n$	$2n$	$12n \log n$	n
‡ Block cyclic reduction	$25 \log n$	$4n$	$19n$	n

* The linear systems (3.10c) are solved via Cramer's rule in 3 time steps.
† Algorithm 3 does not include the computation of the Hurwitz factors.
‡ The block cyclic reduction algorithm does not check for early convergence of the solution.

order n. The various entries show only the leading term. Algorithm 1 is faster than the others due to its use of the fast Fourier transform. Algorithm 3 uses half the number of processors of any other algorithm.

The results of the tests appear in Figure 3. There, the base ten logarithms of the relative errors for each algorithm's solutions appear as functions of the parameter σ that determines the coefficient matrices. The matrices become more ill-conditioned with increasing values of σ. Higher curves correspond to larger relative errors. The tests have been performed on the Control Data Cyber 175 computer at the University of Illinois for which the arithmetic precision is 14.35 decimal digits.

The highest curve of all belongs to Algorithm 1. Its smoothness indicates the presence of strong rounding errors. The circulant systems of linear equations, which are the algorithm's basis, may be the source of these errors. Indeed, the Toeplitz matrix obtained when $\sigma = 4$ has a related circulant matrix which is actually singular, and the algorithm fails. One step of iterative refinement, however, remedies this situation, as shown by the lowest curve in Figure 3. Figure 4 reproduces the error curves for the two versions of Algorithm 1, together with the relative errors in the solution obtained from the Cholesky factorization of the Toeplitz matrix.

The second highest curve, in Figure 3, is that of Algorithm 2. Once again the curve's smoothness indicates persistent rounding errors. This cannot be caused by the small systems of pivot equations (3.10a) which occur once at each step of the algorithm (their coefficient matrices were observed to have condition numbers much less than those of the original Toeplitz matrix). From the linear convergence of the rows of the Cholesky factor of the Toeplitz matrix (see Theorem 3), it can be inferred that these pivot matrices converge quadratically. This rate of convergence may be explained by the fact that each succeeding pivot matrix is associated with a principal submatrix that

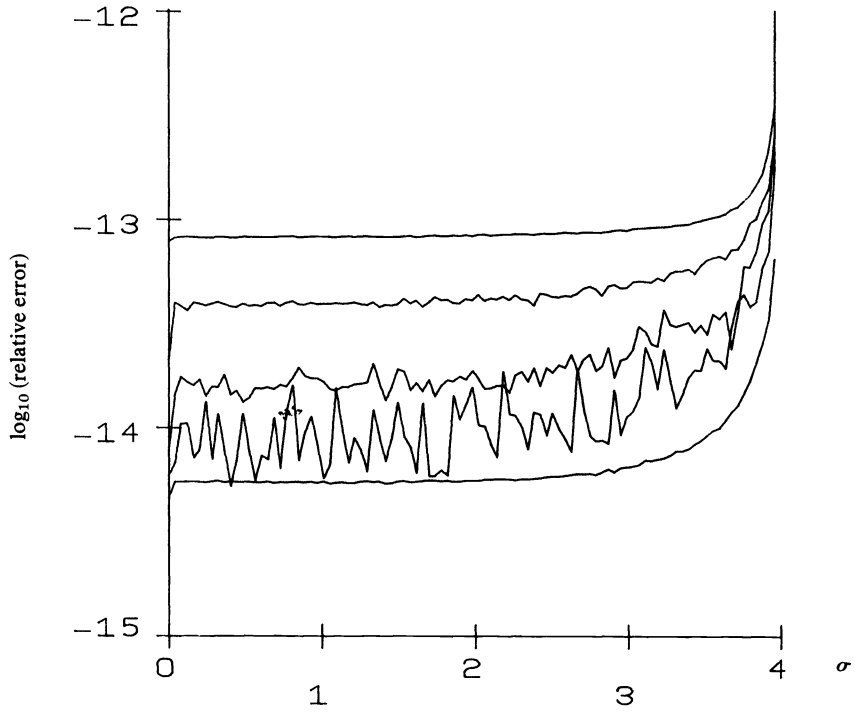


FIG. 3. *From top to bottom the curves represent Algorithm* 1, *Algorithm* 2, *block cyclic reduction, Algorithm* 3, *and Algorithm* 1 *followed by one step of iterative refinement.*

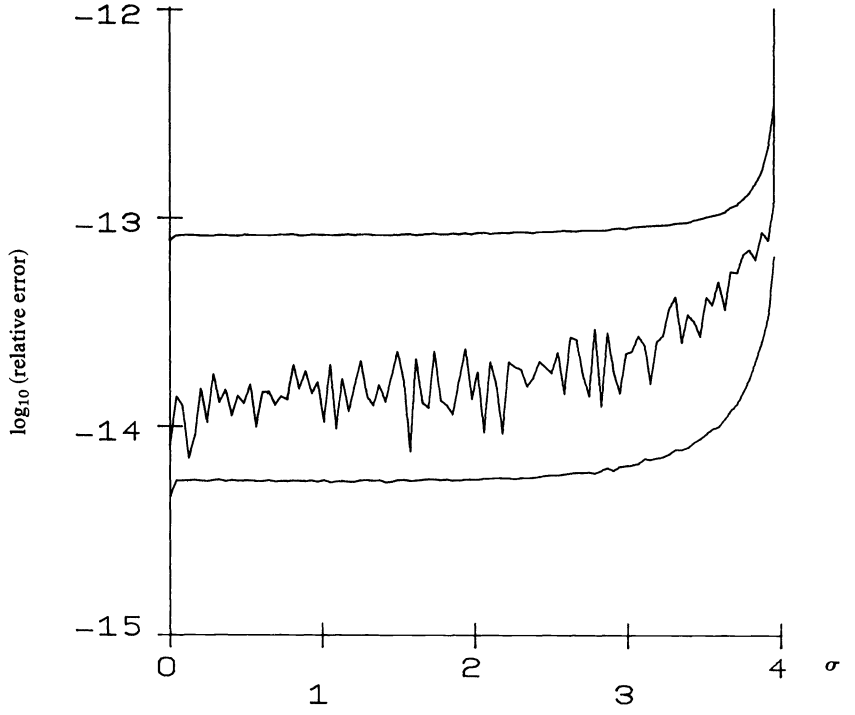


FIG. 4. *The top and bottom curves are reproduced from Fig.* 3. *The center curve represents the solution obtained by Cholesky factorization.*

has double the order of the preceeding submatrix. Even for Toeplitz matrices corresponding to values of σ close to 4, this convergence is remarkably fast. For σ equal to 4 the rows of the Cholesky factor do converge, but slower than linearly. The condition numbers of such Toeplitz matrices grow with their order and are not bounded. In this case, the condition numbers of the pivot matrices were observed to grow exponentially. Only for this one particular system of linear equations the solution found by Algorithm 2 has no accuracy whatsoever.

The center curve displays the relative errors in the solutions produced by the block cyclic reduction algorithm. This algorithm and the Cholesky factorization algorithm are the only two that succeed in solving the system of linear equations corresponding to $\sigma = 4$. Both algorithms, however, attain only 3-digit accuracy for this problem.

The next lower curve represents the errors of Algorithm 3. They are close to and often slightly smaller than those of the Cholesky factorization algorithm (see Fig. 4). The similarity can be explained by the generally fast convergence of the rows of the Cholesky factor to the coefficients of the Hurwitz factor (2.5) used by Algorithm 3. This convergence slows dramatically for values of σ near 4, in which case the Cholesky factorization algorithm has smaller errors. For the case $\sigma = 4$, Algorithm 3 fails to obtain the solution.

Acknowledgments. We are grateful to Gene Golub for sharing his thoughts on this subject. We also wish to thank Paul Swarztrauber for providing us with his Fast Fourier Transform programs, M. Morf for providing us with his translation of the third reference—[Baue55]—and the referees for their helpful comments.

REFERENCES

[Aitk39] A. C. AITKEN, *Determinants and Matrices*, Oliver Boyd, London 1939.

[Bare69] E. H. BAREISS, *Numerical solutions of linear equations with Toeplitz and vector Toeplitz matrices*, Numer. Math., 13 (1969), pp. 404–424.

[Baue55] F. L. BAUER, *Ein direktes Iterationsverfahren zur Hurwitz-Zerlegung eines Polynoms*, Arch. Elektrischen Ubertragung, 9 (1955), pp. 285–290.

[Baue56] ———, *Beitrage zur Entwicklung numerische Verfahren fur programmgesteurete Rechenanlagen: II. Direkte Faktorisierung eines Polynoms*, Bayerische Akademie der Wissenschaften, Mathematisch-Naturwissenschaftliche Klasse, Sitzungsberichte, 1956.

[BrGY80] R. BRENT, F. GUSTAVSON AND D. YUN, RC 8173, Mathematical Sciences Dept., IBM T. J. Watson Res. Ctr., Yorktown Heights, NY, 1980.

[Buck74] A. BUCKLEY, *A note on matrices $A = I + H$, H skew symmetric*, Z. Angew. Math. Mech., 54 (1974), pp. 125–126.

[Buck77] ———, *On the solution of certain skew symmetric linear systems*, SIAM J. Numer. Anal., 14 (1977), pp. 566–570.

[Bune69] O. BUNEMAN, *A Compact Non-iterative Poisson Solver*, Rep. 294, Stanford University Institute for Plasma Research, Stanford, California, 1969.

[CaBu76] A. CANTONI AND P. BUTLER, *Eigenvalues and eigenvectors of symmetric centrosymmetric matrices*, Linear Algebra Appl., 13 (1976), pp. 275–288.

[Davi79] P. J. DAVIS, *Circulant Matrices*, John Wiley, New York, 1979.

[FGHL74] D. FISCHER, G. H. GOLUB, O. HALD, C. LEIVA AND O. WIDLUND, *On Fourier-Toeplitz methods for separable elliptic problems*, Math. Comp., 18 (1974), pp. 349–368.

[GoFe74] I. C. GOHBERG AND I. A. FELDMAN, *Convolution Equations and Projection Methods for Their Solution*, Translations of Mathematical Monographs, Vol. 41, American Mathematical Society, Providence, RI, 1974.

[GoLe78] I. GOHBERG AND S. LEVIN, *Asymptotic properties of Toeplitz matrix factorization*, Integral Equations Operator Theory, 1 (1978), pp. 519–538.

[GoSe72] I. C. GOHBERG AND A. A. SEMENCUL, *On the inversion of finite Toeplitz matrices and their continuous analogues*, Mat. Issled., 2 (1972), pp. 201–233.

[GrSz58] U. GRENANDER AND G. SZEGO, *Toeplitz Forms and Their Applications*, Univ. of California Press, Berkeley, CA, 1958.

[GuYu79] F. G. GUSTAVSON AND D. Y. YUN, *Fast Computation of Padé Approximants and Toeplitz Systems of Equations Via the Extended Euclidean Algorithm*, RC 7551, Mathematical Sciences Dept., IBM T. J. Watson Res. Ctr., Yorktown Heights, NY, 1979.

[Hell76] D. HELLER, *Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems*, SIAM J. Numer. Anal., 13 (1976), pp. 484–496.

[Hell78] ———, *A Survey of Parallel Algorithms in Numerical Linear Algebra*, SIAM Rev. 20 (1978), pp. 740–777.

[Hous64] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Blaisdell, Waltham, MA, 1964.

[Just74] J. H. JUSTICE, *The Szego recurrence relation and inverses of positive definite Toeplitz matrices*, SIAM J. Math. Anal., 5 (1974), pp. 503–508.

[KaVM78] T. KAILATH, A. VIEIRA AND M. MORF, *Inverses of Toeplitz operators, innovations, and orthogonal polynomials*, SIAM Rev., 20 (1978), pp. 106–119.

[KrWi67] H. O. KRIESS AND O. WIDLUND, *Difference Approximations for Initial Value Problems for Partial Differential Equations*, Dept. of Computer Sci. Rpt. NR7, Uppsala Univ., Uppsala, Sweden, 1967.

[Loew77] D. LOEWENTHAL, *Numerical computation of the roots of polynomials by spectral factorization*, in *Topics in Numerical Analysis III*: Proc. Royal Irish Acad. Conf. on Numerical Analysis, 1976, J. J. H. Miller, ed., Academic Press, New York, 1977, pp. 237–255.

[Morf80] M. MORF, *Doubling algorithms for Toeplitz and related equations*, Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, April 9–11, 1980, pp. 954–959.

[MoKa77] M. MORF AND T. KAILATH, *Recent results in least-squares estimation theory*, Ann. Economic Social Measurement, (1977), pp. 261–274.

[Part62] S. V. PARTER, *An observation on the numerical solution of difference equations and a theorem of Szego*, Numer. Math., 4 (1962), pp. 293–295.

[Peas68] M. PEASE, *An adaptation of the fast Fourier transform for parallel processing*, J. Assoc. Comput. Mach., 15 (1968), pp. 252–264.

[Phil71] J. L. PHILLIPS, *The triangular decomposition of Hankel matrices*, Math. Comp., 25 (1971), pp. 599–602.

[RiKa72] J. RISSANEN AND T. KAILATH, *Partial realization of random systems*, Automatica, 8 (1972), pp. 389–396.

[RiNa55] F. RIESZ AND B. SZ.-NAGY, *Functional Analysis*, trans. from 2nd French ed. by L. F. Boron, Frederick Ungar, New York, 1956.

[Riss73] J. RISSANEN, *Algorithms for triangular decomposition of block Hankel and Toeplitz matrices with applications to factoring positive matrix polynomials*, Math. Comp., 27 (1973), pp. 147–154.

[Riss74] J. RISSANEN, *Solution of linear equations with Hankel and Toeplitz matrices*, Numer. Math., 22 (1974), pp. 361–366.

[Robi67] E. A. ROBINSON, *Multichannel Time Series Analysis with Digital Computer Programs*, Holden-Day Series in Time Series Analysis, Holden-Day, San Francisco, 1967.

[Same77] A. H. SAMEH, *Numerical parallel algorithms—A survey*, in *High Speed Computer and Algorithm Organization*, Academic Press, New York, 1977, pp. 207–228.

[Swar74] P. N. SWARZTRAUBER, *A direct method for the discrete solution of separable elliptic equations*, SIAM J. Numer. Anal., 11 (1974), pp. 1136–1150.

[Szeg59] G. SZEGO, *Orthogonal Polynomials*, Rev. Ed., American Math. Society Colloquium Publications, Vol. 23, American Mathematical Society, Providence, RI, 1959.

[Tren64] W. F. TRENCH, *An algorithm for the inversion of finite Toeplitz matrices*, J. Soc. Indust. Appl. Math., 12 (1964), pp. 515–522.

[Tren65] ———, *An algorithm for the inversion of finite Hankel matrices*, J. Soc. Indust. Appl. Math., 13 (1965), pp. 1102–1107.

[Wats73] G. A. WATSON, *An algorithm for the inversion of block matrices of Toeplitz form*, J. Assoc. Comput. Mach., 20 (1973), pp. 409–415.

[Wils69] G. WILSON, *Factorization of the covariance generating function of a pure moving average process*, SIAM J. Numer. Anal., 6 (1969), pp. 1–7.

[Zoha69] S. ZOHAR, *Toeplitz matrix inversion: The algorithm of W. F. Trench*, J. Assoc. Comput. Mach., 16 (1969), pp. 592–601.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0001 $01.00/0

A NONLINEAR PROGRAMMING PROBLEM IN STATISTICS (EDUCATIONAL TESTING)*

ROGER FLETCHER†

Abstract. The educational testing problem is reviewed; it concerns a reliability coefficient which measures how reliable are the student's total scores in an examination consisting of a number of subtests. A lower bound on this coefficient is obtained by solving a certain nonlinear programming problem. Expressions for both first and second derivatives are given, and a scaling feature is described which enables the search for a solution to take place along feasible arcs on the boundary of the feasible region. The SOLVER method is used to generate the directions of search and numerical results are described. While an improvement over previous methods is obtained, some difficulties over slow convergence are observed in some cases and a possible explanation is given.

Key words. educational testing, reliability coefficient, nonlinear programming, SOLVER method

1. Introduction. An interesting problem is to determine how much can be subtracted from the diagonal of a given positive definite matrix S subject to S remaining positive semidefinite, $S \geqq 0$. More precisely, if T is a diagonal matrix $T = \operatorname{diag} \theta_i$, then this problem can be posed as the optimization problem

$$\begin{aligned} &\underset{T}{\text{maximize}} \ \operatorname{trace}(T) \\ &\text{subject to } S - T \geqq 0, \qquad T \geqq 0. \end{aligned} \tag{1.1}$$

This is a convex programming problem, since the objective $(\sum_i u_i)$ is linear and hence convex, and the feasible region is a convex set. (Clearly, if T_0, T_1 are feasible then so is $T_\lambda = (1-\lambda)T_0 + \lambda T_1$, $\lambda \in (0, 1)$; consider $z^T(S - T_\lambda)z$ for any z. Necessarily $T \leqq \operatorname{diag} S_{ii}$, so the feasible region is closed and bounded and a solution always exists. This problem has been found to arise in statistics in the area of educational testing and I am very grateful to Paul H. Jackson (University of Wales, Aberystwyth, UK) for bringing it to my attention. The statistical background is described in some detail in § 2 and is associated with determining how reliable is an examination consisting of a number of subtests. A coefficient ρ is defined which measures this reliability and it is shown that the solution of (1.1) gives a lower bound on ρ. In § 3 it is shown that the problem (1.1) can be reduced to a standard nonlinear programming problem via the solution of an eigenvalue problem. Expressions for both first and second derivatives are shown to be readily calculated. A "scaling" feature is described which enables the search for the solution to take place on the boundary of the feasible region and obviates the need to use a penalty function. The SOLVER method is used to solve the nonlinear programming problem and numerical results with the method are described in § 4. While an improvement over previous methods is obtained, some difficulties over slow convergence are observed. A possible explanation of these difficulties and suggestions for how they can be avoided in future research are described.

* Received by the editors May 8, 1980, and in revised form February 1, 1981. This paper was presented as an invited lecture at the Summer Research Conference on Numerical and Statistical Analysis, Newark, Delaware, June 1980.

† Mathematics Department, University of Dundee, Dundee DD1 4HN, Scotland, UK. This paper was prepared while the author was a visitor at Mathematics Department, University of Kentucky, Lexington, KY 40506.

2. Educational testing. This section explains the statistical background of the educational testing problem which gives rise to the nonlinear programming problem (1.1). The educational testing problem arises when N students take a test or examination consisting of n subtests. It is required to find how reliable is the student's total score in the sense of being able to reproduce this total on two independent occasions. Guttman (1945) has defined a certain *reliability coefficient* ρ ($\leqq 1$) such that $\rho = 1$ corresponds to a completely reliable score. Since in practice it is only possible to carry out the test on one occasion only, it is important to consider what information about reliability can be determined from a single examination. In this case a lower bound on ρ can be determined by solving the nonlinear programming problem (1.1).

The development is largely of Guttman (1945) although many of the ideas may originate from Spearman (1904). The notation of Jackson and Agunwamba (1977) is used. The given data for the problem is an $N \times m$ table of scores $[X_{ij}]$ (*e.g.*, Table 1) such that X_{ij} gives the score of student i on subject j. Columns in the table are referred to by $\underset{\sim}{X}_j$. The student's total score is $X_i = \sum_j X_{ij}$ and $\underset{\sim}{X}$ denotes the column vector of total scores. The mean score for each test is $\bar{X}_j = (\sum_i X_{ij})/N$ and the mean total score

TABLE 1
Data for the Woodhouse (1976) *problems.*

	SUBJECTS																		
STUDENTS	15	25	20	28	35	50	21	18	22	28	28	12	15	40	18	23	14	16	15	10
	21	27	32	32	41	42	30	35	33	32	64	16	24	38	34	13	15	17	28	18
	23	35	40	22	55	48	36	40	46	18	38	18	26	37	24	24	17	20	19	26
	23	29	50	36	42	52	44	32	24	19	32	24	20	46	32	23	11	12	40	38
	34	37	42	19	36	46	17	26	35	28	39	54	21	47	29	42	18	18	30	20
	36	60	70	45	55	54	32	30	32	29	41	28	20	47	36	28	20	20	18	24
	36	35	46	27	50	40	60	34	39	46	48	63	20	48	40	19	21	24	40	22
	38	70	44	50	45	42	20	28	29	16	55	40	22	49	42	25	23	26	30	28
	39	46	52	24	37	60	53	30	46	43	54	54	23	46	44	22	35	22	48	30
	40	74	65	60	72	41	33	36	24	52	64	36	28	50	46	26	25	23	30	30
	40	48	32	23	58	52	23	40	37	24	58	38	29	54	44	28	11	27	41	34
	41	12	24	50	47	48	41	42	37	28	56	57	32	51	43	25	17	24	32	39
	46	52	76	48	70	58	20	50	28	42	76	58	28	58	45	34	27	35	18	56
	46	73	84	63	38	57	33	56	42	18	72	77	21	52	48	32	35	32	32	19
	47	42	74	28	60	57	36	42	48	52	63	46	36	53	49	53	29	30	42	40
	47	82	72	70	39	64	21	25	44	26	44	44	37	51	46	33	38	35	37	42
	47	40	42	50	48	61	40	40	26	29	61	44	30	56	47	52	46	27	48	40
	48	70	65	48	42	57	35	58	50	46	60	32	34	58	54	35	36	31	16	18
	49	65	60	55	62	56	52	50	52	28	50	48	34	58	53	41	45	40	38	52
	50	30	35	28	62	54	41	46	50	21	65	33	32	58	54	38	50	44	43	50
	52	42	54	33	42	64	40	40	56	44	64	38	34	60	44	34	38	30	44	38
	52	72	70	65	72	68	62	38	56	44	58	46	56	58	46	36	55	20	48	47
	52	44	64	72	44	62	35	44	56	46	62	39	30	61	46	38	40	42	24	80
	53	25	42	28	68	52	41	45	44	26	28	43	51	62	47	35	42	48	50	40
	54	48	60	58	36	51	63	41	64	29	63	49	32	58	47	39	43	58	48	49
	55	64	62	30	42	57	34	47	52	34	57	37	43	63	48	38	47	20	54	65
	58	30	24	62	51	51	44	36	43	25	36	54	41	65	48	43	40	35	50	42
	58	16	40	45	42	58	44	42	58	36	58	52	40	64	49	36	18	45	53	28
	58	44	56	51	68	68	46	48	72	38	62	34	32	68	49	42	47	47	28	70
	39	58	58	50	74	52	36	58	60	28	44	56	34	72	51	33	48	58	54	58
	60	32	35	48	40	56	52	32	40	37	72	57	36	61	52	51	42	46	17	51
	60	78	80	62	52	54	58	47	80	32	64	39	45	66	53	42	70	40	50	18
	60	38	55	66	42	52	30	54	62	42	90	38	38	63	56	46	62	48	55	44
	61	48	64	68	70	53	42	40	38	45	73	56	50	64	54	46	45	42	50	22
	62	86	94	50	49	62	48	56	74	33	84	36	52	67	52	47	41	70	57	53

TABLE 1—*continued*

	63	35	38	55	38	58	46	59	63	48	62	58	38	68	53	47	48	75	44	25
	64	79	65	76	68	57	32	33	52	46	72	62	52	54	54	48	55	35	60	54
	64	50	52	35	60	56	52	64	76	36	63	44	48	56	52	49	63	38	48	46
	65	37	42	70	50	58	58	62	66	34	53	64	62	72	53	50	74	45	62	57
	65	82	74	63	36	62	60	58	60	38	57	63	58	70	51	51	55	55	64	58
	67	44	46	54	52	58	55	68	80	40	28	65	50	71	54	52	65	70	74	68
	67	48	56	80	44	64	62	35	70	62	58	72	56	72	60	41	78	38	75	68
	68	62	78	56	50	53	62	54	80	64	62	48	54	74	62	76	58	45	80	36
	69	39	30	42	38	62	40	32	68	56	68	71	58	73	56	43	79	47	73	70
	70	52	20	76	69	61	64	56	69	54	64	66	58	78	52	72	32	47	71	71
	72	54	72	38	54	51	66	65	76	72	54	49	60	74	57	42	68	62	22	46
	72	42	48	70	70	57	42	40	68	53	62	74	60	78	58	52	55	48	72	75
STUDENTS	74	64	66	70	42	60	40	78	53	48	69	67	76	77	59	68	58	55	16	60
	78	68	62	63	35	56	63	80	74	43	71	78	62	76	59	68	70	75	72	75
	79	37	42	28	64	52	40	38	72	72	56	52	58	82	60	61	75	66	58	58
	80	62	30	65	59	51	68	57	74	48	78	71	42	54	61	62	78	69	78	58
	82	85	80	52	44	57	70	69	64	71	85	76	64	56	63	67	44	70	60	26
	82	40	74	52	52	64	74	64	76	46	64	46	51	80	63	68	65	55	70	67
	84	42	76	70	55	61	90	80	56	41	58	82	72	72	67	73	85	60	76	78
	84	42	54	60	42	58	42	60	52	70	77	68	68	74	59	71	79	65	44	75
	86	85	88	80	37	63	80	72	79	42	73	42	68	82	65	73	85	62	80	82
	87	53	51	62	68	56	60	42	78	42	62	74	62	84	65	75	63	75	68	83
	89	41	60	40	60	54	88	88	83	57	84	64	64	80	62	65	90	78	88	52
	90	73	78	77	52	42	56	50	58	59	72	84	70	84	62	63	82	85	78	87
	90	81	74	64	48	38	86	52	80	63	66	68	60	62	63	74	75	81	84	94
	96	85	88	90	72	44	58	62	70	74	64	74	72	64	64	84	85	78	88	54
	97	56	55	35	68	70	78	76	56	72	83	69	65	86	65	76	82	89	92	90
	99	75	65	88	54	42	80	90	88	58	78	88	70	88	63	82	72	71	98	80
	100	65	75	70	70	60	83	85	70	62	72	90	72	84	64	78	88	80	80	72

regarded as having been sampled from a universe of tests, and $\mathscr{E}[\cdot]$ denotes the expected value on this universe. Then it is assumed that

$$X_{ij} = T_{ij} + E_{ij} \quad \forall i, j \tag{2.1}$$

where

$$\mathscr{E}[E_{ij}] = 0. \tag{2.2}$$

The quantities $[T_{ij}]$ represent hypothetical unknown "true scores" and true total scores T, mean scores $\bar{T}_j$ and mean total score $\bar{T}$ are defined as for the observed scores, and are the expected values of the corresponding observed scores from (2.2).

To define the reliability of the total scores in the test, the variance of the total scores from the expected mean score

$$\sigma_x^2 = \frac{1}{N-1} \sum_i \mathscr{E}[(X_i - \mathscr{E}[\bar{X}])^2]$$

$$= \frac{1}{N-1} \sum_i \mathscr{E}[(X_i - \bar{T})^2] \tag{2.3}$$

is defined. Division by $N-1$ and not N expresses the fact that one degree of freedom is lost by taking the mean total score $\bar{T}$. Reliability of the test can be regarded as being the correlation in the student's total scores from two independent tests selected from

the hypothetical universe. Let $[{}_1X_{ij}]$ and $[{}_2X_{ij}]$ denote from two such tests and ${}_1\underset{\sim}{X}$ and ${}_2\underset{\sim}{X}$ the corresponding total scores. Then Guttman (1945) defines the *reliability coefficient* ρ by

$$\rho^2 = \frac{1}{N-1}\frac{\sum_i \mathscr{E}[({}_1X_i - \bar{T})({}_2X_i - \bar{T})]}{\sigma_X^2}. \tag{2.4}$$

This is seen to be a correlation between the observed total scores; in a perfectly reliable test ${}_1X_i = {}_2X_i$, and it follows from (2.3) that $\rho = 1$. Assuming that the errors E_i in the total scores are uncorrelated, so that

$$\mathscr{E}[{}_1E_i \cdot {}_2E_i] = \mathscr{E}[{}_1E_i]\mathscr{E}[{}_2E_i], \tag{2.5}$$

it follows that

$$\rho^2 = \frac{\sigma_T^2}{\sigma_X^2} \tag{2.6}$$

where

$$\sigma_T^2 = \frac{1}{N-1}\sum_i (T_i - \bar{T})^2. \tag{2.7}$$

Clearly a reliability coefficient close to 1 indicates that observed total scores are very reliable. One practical question which is of interest is precisely what value of ρ indicates that a test is sufficiently reliable; that is to say, should one aim for $\rho \geqq 0.9$, or $\rho \geqq 0.8$, or some other target. Possibly the criterion should depend upon n also. Some practical observations are made at the end of § 4.

Unfortunately it is not practical to estimate ρ by making two or more independent tests unless some way of inducing complete amnesia in the students after each test can be found! Therefore it is desirable to determine what information can be deduced from a single test. This can be done by relating σ_X^2 and σ_T^2 to certain variance-covariance matrices. The variance-covariance matrix Σ_X of observed scores from the expected mean observed scores on the subtests is defined as

$$\begin{aligned}[\Sigma_X]_{jk} &= \frac{1}{N-1}\sum_i \mathscr{E}[(X_{ij} - \mathscr{E}[\bar{X}_j])(X_{ik} - \mathscr{E}[\bar{X}_k])] \\ &= \frac{1}{N-1}\sum_i \mathscr{E}[(X_{ij} - \bar{T}_j)(X_{ik} - \bar{T}_k)].\end{aligned} \tag{2.8}$$

Likewise matrices Σ_T and Σ_E are defined by

$$[\Sigma_T]_{jk} = \frac{1}{N-1}\sum_i (T_{ij} - \bar{T}_j)(T_{ik} - \bar{T}_k), \tag{2.9}$$

$$[\Sigma_E]_{jk} = \frac{1}{N-1}\sum_i \mathscr{E}[E_{ij}E_{ik}]. \tag{2.10}$$

If we assume uncorrelated errors in the sense that

$$\mathscr{E}[E_{ij}T_{ik}] = \mathscr{E}[E_{ij}]T_{ik} \quad \forall i, j, k \tag{2.11}$$

and

$$\mathscr{E}[E_{ij}E_{ik}] = \mathscr{E}[E_{ij}]\mathscr{E}[E_{ik}] \quad \forall i, j, k, \qquad j \neq k, \tag{2.12}$$

then it follows from (2.2) that Σ_E is a diagonal matrix and from (2.9) and (2.1) that

$$\Sigma_X = \Sigma_T + \Sigma_E. \tag{2.13}$$

It also follows simply from the definitions (2.3) and (2.9) that

$$\sigma_x^2 = \sum_{j,k} [\Sigma_E]_{jk} = \underline{e}^T \Sigma_X \underline{e}, \tag{2.14}$$

where $\underline{e} = (1, 1, \ldots, 1)^T$, and likewise that

$$\sigma_T^2 = \underline{e}^T \Sigma_T \underline{e}. \tag{2.15}$$

Hence, writing the diagonal elements of Σ_E for convenience as $\theta_i (= [\Sigma_E]_{ii})$, we can express (2.6) as

$$\rho^2 = 1 - \frac{\sum_i \theta_i}{\sigma_X^2}. \tag{2.16}$$

The situation can be summarized in the following way. The matrix Σ_X can be estimated by

$$[\Sigma_X]_{jk} \simeq \frac{1}{N-1} \sum_i (X_{ij} - \bar{X}_j)(X_{ik} - \bar{X}_k), \tag{2.17}$$

which is acceptable if N is large (see Guttman (1945)), and can be regarded as being known. The matrices Σ_T and Σ_E are unknown but being variance-covariance matrices they are necessarily positive semi-definite. These conditions can be written using (2.13) as

$$\Sigma_X - \Sigma_E \geqq 0, \qquad \Sigma_E \geqq 0 \tag{2.18}$$

and can be regarded as constraints on the values of the θ_i. Clearly, then, the θ_i satisfy

$$\begin{aligned} \sum_i \theta_i &\leqq \max_{\theta_i} \sum_i \theta_i \qquad \text{subject to (2.18)} \\ &= \phi, \end{aligned} \tag{2.19}$$

say, so that from (2.16)

$$\rho^2 \geqq 1 - \frac{\phi}{\sigma_X^2}. \tag{2.20}$$

Thus, by solving the optimization problem in (2.19) which defines ϕ, a lower bound on ρ is obtained. This is the best that can be done on the basis of a single test. If solving for ϕ indicates that $\rho \geqq 0.9$, or whatever value is considered suitable, then it is possible to conclude that the test is sufficiently reliable.

3. Solving the optimization problem. It is convenient to simplify the notation here and write $T = \Sigma_E = \text{diag}\ \theta_i$ and $S \simeq \Sigma_X$ as the matrix calculated in (2.17). Then (2.19) becomes the optimization problem (1.1). An earlier approach to solving (1.1) is due to Bentler (1972) who writes $S - T = FF^T$, where F is unknown and minimizes trace (FF^T) subject to certain conditions. This is clumsy in that there are a large number of variables, and it also does not account for the condition $T \geqq 0$. Furthermore, some difficulties over convergence are experienced. Woodhouse and Jackson (1977) attempt to solve problem (1.1) by a search in the space of the θ_i, but it seems that their method does not work well on other than small problems.

In this paper it is shown that (1.1) can be written as a nonlinear programming problem in conventional form. What is possibly the most obvious way of doing this did not occur to me initially but may be worth pursuing and is described at the end of § 4.

The approach given here is to consider the generalized eigenproblem

$$TX = SXU, \tag{3.1}$$

with an eigenvalue matrix $U = \operatorname{diag} \mu_i$ and an eigenvector matrix $X = [\underset{\sim}{x}_1, \underset{\sim}{x}_2, \ldots, \underset{\sim}{x}_n]$ which collect eigenvalues and vectors μ_i and $\underset{\sim}{x}_i$ for $i = 1, 2, \ldots, n$. Two properties of the eigensystem are

$$X^T S X = I \tag{3.2}$$

and

$$X^T T X = U. \tag{3.3}$$

From (3.2) X^{-1} exists so the columns of X are independent. A simple result which uses these results is the following.

LEMMA 1. $S - T \geqq 0$ *iff* $\mu_i \leqq 1$, $i = 1, 2, \ldots, n$.

Proof. $S - T \geqq 0$ implies $\underset{\sim}{x}_i^T S \underset{\sim}{x}_i \geqq \underset{\sim}{x}_i^T T \underset{\sim}{x}_i = \underset{\sim}{x}_i^T S \underset{\sim}{x}_i \mu_i$ from which $\mu_i \leqq 1$ follows. Conversely any vector $\underset{\sim}{y}$ can be expanded as $\underset{\sim}{y} = X\underset{\sim}{\alpha}$ and

$$\begin{aligned} \underset{\sim}{y}^T (S - T)\underset{\sim}{y} &= \underset{\sim}{\alpha}^T (X^T S X - X^T T X)\underset{\sim}{\alpha} \\ &= \underset{\sim}{\alpha}^T (I - U)\underset{\sim}{\alpha} \geqq 0 \end{aligned}$$

if $\mu_i \leqq 1$ $i = 1, 2, \ldots, n$, which shows that $S - T \geqq 0$. □

Thus, to solve (1.1) the unknowns are regarded as $\theta_1, \theta_2, \ldots, \theta_n$ (or T). The eigenproblem (3.1) is solved for any such T and the eigenvalues μ_i are regarded as nonlinear functions $\mu_i(T)$. Then problem (1.1) can be restated as

$$\begin{aligned} &\underset{\theta_i}{\text{minimize}} \; -\sum_i \theta_i \\ &\text{subject to } \mu_i(\theta_1, \theta_2, \ldots, \theta_n) \leqq 1, \quad \theta_i \geqq 0, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{3.4}$$

This is a nonlinear programming problem in standard form. The objective function is linear and so the location of the solution is usually determined by curvature in the constraints $\mu_i(T) \leq 1$. Thus it is important to use a numerical method for solving (3.4) which uses second derivative information and such a method is described later in the section. It is also convenient to order the eigenvalues as

$$\mu_1 \geqq \mu_2 \geqq \ldots \geqq \mu_n, \tag{3.5}$$

in which case a possible alternative statement of the nonlinear constraints in (3.4) is to have just $\mu_1(T) \leqq 1$. However $\mu_1(T)$ is a nonsmooth function when μ_1 is a multiple eigenvalue. When this situation arises, or nearly so, it can be advantageous in method (3.15) below to have present linearizations from all the constraints rather than just $\mu_1(T) \leqq 1$. Thus the possibility of having a single nonlinear constraint is not explored.

It is important first of all to determine formulae for derivatives to be used in the numerical method. Writing (3.1) as

$$T\underset{\sim}{x}_i = S\underset{\sim}{x}_i \mu_i, \qquad i = 1, 2, \ldots, n \tag{3.6}$$

and differentiating with respect to θ_i yields

$$\left(\frac{\partial \mu_i}{\partial \theta_j} S - \frac{\partial T}{\partial \theta_j}\right) \underset{\sim}{x}_i = (T - \mu_i S) \frac{\partial \underset{\sim}{x}_i}{\partial \theta_j}. \tag{3.7}$$

Premultiplying by $\underset{\sim}{x}_i^T$ and using (3.6), (3.2) and $T = \text{diag}\ \theta_i$ gives

$$\frac{\partial \mu_i}{\partial \theta_j} = \underset{\sim}{x}_i^T \frac{\partial T}{\partial \theta_j} \underset{\sim}{x}_i = x_{ji}^2. \tag{3.8}$$

This gives the required expression for the first derivative of μ_i with respect to θ_j for any i, j. In passing it is noted that a more complicated dependence of T on the θ_i could be handled readily. To obtain second derivatives, (3.7) is differentiated with respect to θ_k giving

$$\left(\frac{\partial^2 \mu_i}{\partial \theta_j\, \partial \theta_k} S - \frac{\partial^2 T}{\partial \theta_j\, \partial \theta_k}\right) \underset{\sim}{x}_i - \left(\frac{\partial T}{\partial \theta_j} - \frac{\partial \mu_i}{\partial \theta_j} S\right) \frac{\partial \underset{\sim}{x}_i}{\partial \theta_k}$$

$$= \left(\frac{\partial T}{\partial \theta_k} - \frac{\partial \mu_i}{\partial \theta_k} S\right) \frac{\partial \underset{\sim}{x}_i}{\partial \theta_j} + (T - \mu_i S) \frac{\partial^2 \underset{\sim}{x}_i}{\partial \theta_j\, \partial \theta_k}.$$

Using $\partial^2 T / \partial \theta_k = 0$, premultiplying by $\underset{\sim}{x}_i^T$ and using (3.6) and (3.2) yields

$$\begin{aligned} \frac{\partial^2 \mu_i}{\partial \theta_j\, \partial \theta_k} &= \underset{\sim}{x}_i^T \left(\frac{\partial T}{\partial \theta_j} - \frac{\partial \mu_i}{\partial \theta_j} S\right) \frac{\partial \underset{\sim}{x}_i}{\partial \theta_k} + \underset{\sim}{x}_i^T \left(\frac{\partial T}{\partial \theta_k} - \frac{\partial \mu_i}{\partial \theta_k} S\right) \frac{\partial \underset{\sim}{x}_i}{\partial \theta_j} \\ &= -2 \frac{\partial \underset{\sim}{x}_i}{\partial \theta_j} (T - \mu_i S) \frac{\partial \underset{\sim}{x}_i}{\partial \theta_k} \end{aligned} \tag{3.9}$$

from (3.7). The matrix $T - \mu_i S$ is singular, and the generalized inverse matrix $(T - \mu_i S)^+$ satisfies

$$T - \mu_i S = (T - \mu_i S)(T - \mu_i S)^+ (T - \mu_i S);$$

so using this in (3.9) together with (3.7) gives

$$\frac{\partial^2 \mu_i}{\partial \theta_j\, \partial \theta_k} = -2 \underset{\sim}{x}_i^T \left(\frac{\partial T}{\partial \theta_k} - \frac{\partial \mu_i}{\partial \theta_k} S\right) (T - \mu_i S)^+ \left(\frac{\partial T}{\partial \theta_j} - \frac{\partial \mu_i}{\partial \theta_j} S\right) \underset{\sim}{x}_i. \tag{3.10}$$

To simplify this expression the following result is used.

LEMMA 2.

$$(T - \mu_i S)^+ = \sum_{p \in I_i} \frac{\underset{\sim}{x}_p \underset{\sim}{x}_p^T}{\mu_p - \mu_i}, \tag{3.11}$$

where

$$I_i = \{p : \mu_p \neq \mu_i\}. \tag{3.12}$$

Proof. From (3.2) and (3.3)

$$T - \mu_i S = X^{-T} (U - \mu_i I) X^{-1}.$$

Now $U - \mu_i I$ is diagonal with zero elements when $\mu_p = \mu_i$, so $(U - \mu_i I)^+$ is diagonal with corresponding zero elements and elements $1/(\mu_p - \mu_i)$ otherwise. Thus

$$(T - \mu_i S) = X (U - \mu_i I)^+ X^T,$$

from which (3.11) follows. □

Substituting (3.11) into (3.10) and using the form of $\partial T / \partial \theta_j$ gives

$$\frac{\partial^2 \mu_i}{\partial \theta_j\, \partial \theta_k} = 2 \sum_{p \in I_i} \frac{x_{kp} x_{ki} x_{jp} x_{ji}}{\mu_i - \mu_p} \tag{3.13}$$

which is the required expression for second derivatives. The resulting matrix of second derivatives of μ_i is referred to as $\nabla^2 \mu_i$. If I_i has c_i elements, and defining $y_{jp}^i = x_{jp} x_{ji}$

and $d_i = 1/(\mu_i - \mu_p)$ as elements of the $n \times c_i$ matrix Y^i and the $c_i \times c_i$ diagonal matrix D^i respectively, then (3.13) can be written

$$\nabla^2 \mu_i = Y^i D^i Y^{iT}. \tag{3.14}$$

This shows that $\nabla^2 \mu_i$ has the same rank as Y^i which is at most c_i, and $c_i \leqq n-1$. It follows that $\nabla^2 \mu_i$ is singular and the reason for this is seen when ideas of scaling are described below. It also follows under the ordering (3.5) that $\nabla^2 \mu_1$ is positive semidefinite which is consistent with the convexity result in § 1. Equation (3.13) also shows that $\nabla^2 \mu_i$ is a discontinuous function of T if the multiplicity of an eigenvalue changes. This fact may be the cause of some numerical difficulties when near multiplicity occurs at the solution to (3.4).

A feature of problem (3.4) which has implications for the choice of method is the possibility of *scaling*. It is a property of (3.1) that scaling $T \to kT$ causes $\mu_i \to k\mu_i$ for all i and leaves x_i unchanged. Consider any point $T = \text{diag}\, \theta_i$ which is strictly feasible in (3.4) and therefore has $\mu_1(T) < 1$ (see Fig. 1). It is possible to replace T by another

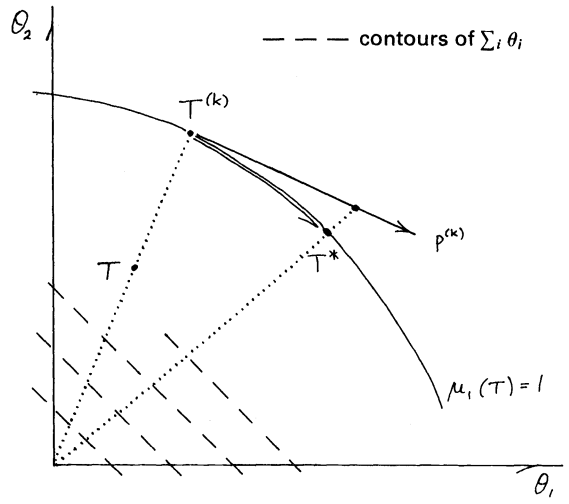


FIG. 1. *Scaling and the feasible arc search.*

point ($T^{(k)}$ in Fig. 1) by scaling T by $k = 1/\mu_1$ so that $\mu_1(T) = 1$ at the new point which is therefore on the boundary of the feasible region. Also trace(T) is increased so the value of the objective function is improved. Since the eigenvectors are unchanged, derivatives at the new point are available at no extra cost. Also if $T^{(k)}$is an iterate in some method and $\underset{\sim}{p}^{(k)}$ is a search direction in the tangent plane (a usual feature) then it is possible to search along the *feasible arc* which is obtained by scaling points on the line $\underset{\sim}{T}^{(k)} + \alpha \underset{\sim}{p}^{(k)}$ to lie on the boundary of the feasible region as illustrated in Figure 1. This feature implies that there is no need to use a penalty function approach to solving (3.4).

A numerical method which is likely to be suitable for solving (3.4) is the SOLVER method (see Fletcher (1974)). In this case the quadratic programming subproblem

$$\begin{aligned} &\underset{\underset{\sim}{p}}{\text{minimize}}\ \tfrac{1}{2}\, \underset{\sim}{p}^T W^{(k)} \underset{\sim}{p} - \underset{\sim}{e}^T \underset{\sim}{p} \\ &\text{subject to } \underset{\sim}{p}^T \nabla \mu_i^{(k)} \leqq 1 - \mu_i^{(k)}, \qquad p_i \geqq -\theta_i, \quad i = 1, 2, \cdots, n \end{aligned} \tag{3.15}$$

is solved on the kth iteration. The constraints in (3.15) are the constraints in (3.4)

linearized about $T^{(k)}$, and $\mu_i^{(k)}$ etc. refers to $\mu_i(T^{(k)})$. The matrix $W^{(k)}$ is defined by

$$W^{(k)} = \sum_i \lambda_i^{(k)} \nabla^2 \mu_i^{(k)}, \tag{3.16}$$

where the $\lambda_i^{(k)}$ are Lagrange multiplier estimates taken as the multipliers of the linear constraints in (3.15) for iteration $k-1$ (if $k>1$), or as $\lambda_1^{(1)} = \varrho^T \nabla\mu_1 / (\nabla\mu_1^T \nabla\mu_1)$ and $\lambda_i^{(1)} = 0$, $i>1$, if $k=1$.

The method has been programmed as described here in the M.Sc. thesis of Jayarajan (1979). The quadratic programming problem is solved by subroutine VEO2A from the Harwell Subroutine Library. A simple line search is used starting with a unit step $\alpha^{(k)} = 1$ and using repeated quadratic interpolation (but limited to reducing $\alpha^{(k)}$ by $\frac{1}{10}$ at each step) until a better value of the objective function is obtained. The eigenvalue problem is solved by the technique of calculating Choleski factors $S = LL^T$ (once and for all) and rewriting (3.1) as the standard eigenvalue problem

$$(L^{-1}TL^{-T})Y = YU \tag{3.17}$$

where $Y = L^T X$. This is implemented using a subroutine from the NAG library. In fact, from (2.17) S can be written as $S = M^T M$ where

$$M_{ij} = \frac{1}{\sqrt{N-1}} (X_{ij} - \bar{X}_j) \tag{3.18}$$

and L can be calculated in a more stable way numerically from QR factors of the matrix M.

4. Numerical results and discussion. Jayarajan (1979) gives numerical results which check out the method on some small problems and illustrate the rapid (second order) rate of convergence which is to be expected from the SOLVER method. Some larger problems given by Woodhouse (1976) have also been solved. These are obtained from the data given in Table 1 which corresponds to 64 students and 20 subtests. Different selections from the set of subtests are used to generate different problems. Jayarajan uses the six problems generated by the following columns of Table 1: (1, 2, 5, 6), (1, 3, 4, 5), (1, 2, 3, 6, 8, 10), (1–8), (1–12) and (1–18). The full set of 20 subtests is in fact not used. The results are given in Table 2. Each problem is solved from two different initial values; the first from Woodhouse's (1976) best solution and the second from $T^{(1)} = I$, suitably scaled. The number of iterations required is given in the respective columns of Table 2. A feature of the SOLVER method is that a number

TABLE 2

Numerical results (Jayarajan (1979)).

n	Number of iterations		m	Best function value ϕ in (2.19)		Lower bound on ρ
				SOLVER	Woodhouse	
4	3	8	1	542.7736	542.7	.796
4	6	25$^+$	2	633.1441	633.2*	.854
6	3	10	1	305.4817	305.6*	.968
8	15$^+$	25$^+$	3	743.6962	741.6	.978
12	15$^+$	25$^+$	3	641.4196	628.4	.957
18	15$^+$	25$^+$	3	750.7517	608.0	.993

$^+$—terminated due to slow convergence; *—not a feasible point.

(m, say) of the constraints which are linearizations of $\mu_i(T) \leqq 1$ are active on each iteration. After a few iterations this number is observed to settle down to a single value, and this is the value of m which is tabulated. A main result which the table shows is that the method obtains a better solution than that which Woodhouse (1976) gives, especially when n is large. Therefore the method appears to be an improvement on others which have been tried. However, there is one adverse feature; slow convergence is observed when the value of m is greater than one. This is obviously related to the situation in which nearly multiple eigenvalues occur. In fact a modified version of the method was tried in which the set $I_1 = \{m+1, \cdots, n\}$ is used to replace the set $I_1 = \{2, \cdots, n\}$ which arises in the definition (3.13) of $\nabla^2 \mu_1$ when $\mu_1 > \mu_2$. The purpose of this is to try to anticipate nearly multiple eigenvalues and in fact the results in Table 2 are computed using this modification. However, Jayarajan (1979) shows that neither version exhibits rapid convergence. Exhaustive checks on differences and first order conditions have been carried out but no fault in the computer code has become apparent.

One feature which these checks do show up is that near the solution when $m > 1$, small perturbations to T make large changes to the eigenvectors. This ill-conditioning suggests that to transform using (3.17) might be numerically unstable. Therefore an orthogonal (eigenvector) decomposition of $S = Q\Lambda^2 Q^T$ might improve matters. This is best achieved by making a singular value decomposition $M = V\Lambda Q^T$ of the matrix M in (3.18). However a referee makes the pertinent point that eigenvectors corresponding to close eigenvalues are inherently badly determined, irrespective of the numerical method used to solve the eigenvalue problem. Since the first and second derivatives (3.8) and (3.9) involve these eigenvectors, it is likely that ill-conditioning will continue to cause difficulties.

A likely explanation of the ill-conditioning is the following. As functions of the parameter T the trajectories $\mu_i(T) = 1$ are unlikely to cross but behave as illustrated in Fig. 2. It can be seen that constraint linearizations remote from the solution are likely to predict a value of $m > 1$, although the value $m = 1$ actually holds at the solution. In the neighborhood of the near crossing point, the eigenvectors swing round rapidly from one vector to another, so that the eigenvectors corresponding to different eigenvalues appear to swap over. This accounts for the ill-conditioning and also suggests that the

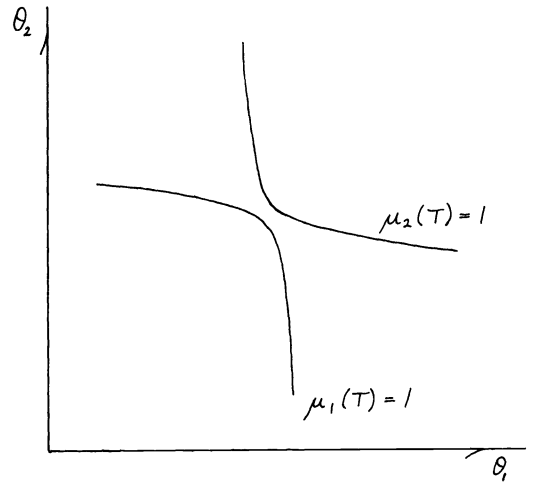


FIG. 2. *Eigenvalue trajectories.*

idea of replacing the set I_1 as described above may not be good. We hope to investigate these possibilities in future work.

Another possibility is to set up the nonlinear constraints corresponding to the condition $S-T \geqq 0$ in a simpler way. If the eigenproblem

$$(S-T)X = XU \tag{4.1}$$

defines eigenvalues μ_i as functions $\mu_i(T)$, then $S-T \geqq 0$ is also equivalent to the nonlinear constraints that $\mu_i(T) \geqq 0$ $i = 1,2,\cdots, n$. Furthermore, an analogue of the scaling idea in § 3 is a *shifting* process. That is, if T is feasible and $\mu_1 \geqq \mu_2 \geqq \cdots \geqq \mu_n > 0$, then a shift $T \to T + \mu_n I$ causes the eigenvalues to change to $\mu_i \to \mu_i - \mu_n$. As in § 3, T is moved to the boundary of the feasible region and the objective function value is improved. In a similar way a line search along a feasible arc can be determined by shifting the points $\underset{\sim}{T}^{(k)} + \alpha \underset{\sim}{p}^{(k)}$. We hope to investigate whether this method or that described in § 3 is best able to handle the ill-conditioned nature of the problem.

The above comments are all concerned with the numerical properties of the optimization method. The results of Table 2 also throw a little light on the statistical question of how large ρ should be for a reliable test. The lower bound in (2.20) for each test problem in Table 2 has been calculated from Jayarajan's solution. A study of all subjects in Table 1 indicates a definite trend that the students are listed in order of increasing scores (approximately). Thus it is likely that these tests are reliable, and therefore that the lower bound on ρ of about 0.8 or better in the table should be an indication of a reliable test.

REFERENCES

P. M. BENTLER, (1972), *A lower-bound method for the dimension-free measurement of internal consistency*, Social Sci. Res., 1, pp. 343–357.

R. FLETCHER, (1974), *Methods related to Lagrangian functions*, in *Numerical Methods for Constrained Optimization*, P. E. Gill and W. Murray, eds., Academic Press, London.

L. GUTTMAN, (1945), *A basis for analyzing test-retest reliability*, Psychometrika, 10, pp. 255–282.

P. H. JACKSON, AND C. C. AGUNWAMBA, (1977), *Lower bounds for the reliability of the total score on a test composed of non-homogeneous items: I. Algebraic lower bounds*, Psychometrika, 42, pp. 567–578.

S. JAYARAJAN, (1979), *A nonlinear optimization problem in educational testing*, M.Sc. Thesis, Dept. of Mathematics, University of Dundee, Scotland.

C. SPEARMAN, (1904), *The proof and measurement of association between two things*, Amer. J. Psychol., 15, pp. 72–101.

B. WOODHOUSE, (1976), *Lower bounds for the reliability of a test*, M.Sc. Thesis, Dept. of Statistics, University of Wales, Aberystwyth.

B. WOODHOUSE, AND P. H. JACKSON, (1977), *Lower bounds for the reliability of the total score on a test composed of non-homogeneous items: II. A search procedure to locate the greatest lower bound*, Psychometrika, 42, pp. 579–591.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0002 $01.00/0

FAMILIES OF HIGH ORDER ACCURATE DISCRETIZATIONS OF SOME ELLIPTIC PROBLEMS*

RONALD F. BOISVERT†

Abstract. In this paper we construct and analyze high order finite difference discretizations of a class of elliptic partial differential equations. In particular, two one-parameter families of fourth order HODIE discretizations of the Helmholtz equation are derived and a discretization optimal with respect to a certain norm of the truncation error is identified. The use of compact nine-point formulas of positive type admits both fast direct methods and standard iterative methods for the solution of the resulting systems of linear equations. Extensions yielding sixth order accuracy for the Helmholtz equation and fourth order accuracy for a more general operator are given. Finally, numerical results demonstrating the effectiveness of the discretizations for a wide range of problems are presented.

Key words. Finite difference methods, elliptic partial differential equations, high order accuracy, Poisson equation

1. Introduction.

We consider the numerical solution to the problem

$$\begin{aligned} Lu \equiv au_{xx} + cu_{yy} + fu = g \quad &\text{on } R, \\ u = \phi \quad &\text{on } \partial R \end{aligned} \tag{1.1}$$

where $a > 0$, $c > 0$, $f \leqq 0$ are constants and R is a rectangular domain with boundary ∂R. We will extend our methods to slightly more general operators L in the sequel. Problems with mixed boundary conditions and some non-rectangular domains are treated in [4], [5] and [6].

We construct and analyse finite difference discretizations of the HODIE type [23]. The method yields high order accurate formulas that are compact as possible, require multiple evaluations of the source function g, yet do not involve derivatives. For the class of problems considered here, it produces coefficients of positive type. The associated linear systems of difference equations have the same block tridiagonal structure required for the straightforward application of both fast direct solution methods and standard iterative methods; in fact, the coefficients are $O(h^2)$ perturbations of the standard nine-point compact discretization of the Laplacian.

We summarize the HODIE method in § 2. Details are given in [4], [23] and [24]. In § 3 we derive two one-parameter families of fourth order discretizations based upon the location of certain auxiliary points where the source function g is evaluated. An error analysis is given in § 4 in which a discretization optimal with respect to a certain norm on the truncation error is given. Properties of the linear systems of difference equations arising in the methods are outlined in § 5. In § 6 the results of numerical experiments undertaken to evaluate the performance of the algorithms for problems of varying degree of difficulty are summarized. Finally, extensions to slightly more general differential operators and a technique for combining members of the fourth order families to obtain sixth order accuracy are described in §§ 7 and 8.

2. The HODIE method.

Let G denote the intersections of a rectangular grid with mesh sizes $h_x \times h_y$ placed over R. For each point $p \in G \cap (R - \partial R)$ we denote the closed grid rectangle centered at p by R_p. A HODIE discretization of the general linear second

* Received by the editors April 21, 1980, and in revised form, February 5, 1981.

† Scientific Computing Division, National Bureau of Standards, Washington, DC, 20234. This work is a part of the author's Ph.D. thesis prepared at Purdue University, W. Lafayette, Indiana, under the supervision of Prof. John R. Rice.

order partial differential equation (PDE) $Lu = g$ takes the form

$$L_p U = I_p g, \qquad p \in G, \tag{2.1a}$$

$$L_p U = \sum_{q \in D_p} \alpha_{pq} U_q, \qquad I_p g = \sum_{q \in E_p} \beta_{pq} g_q. \tag{2.1b}$$

Here U_q is the value of the unknown function u at one of the nine grid points in R_p which we call the set of *discretization points* and denote by D_p. The symbol g_q denotes the evaluation of g at one of another set of points in R_p called auxiliary or *evaluation points* and denoted by E_p. The number of points and their location depends upon the desired order of accuracy of the scheme. The acronym HODIE stands for High Order Differences with Identity Expansion.

The values of the coefficients α_{pq} and β_{pq} are determined by requiring that the difference approximation be exact on some finite dimensional linear space S; that is, we require that $L_p s = I_p L s$ for all $s \in S$. Let $\{s_l : l = 1, \cdots, \dim(S)\}$ be a basis for the space S, where $\dim(S)$ denotes its dimension; then the difference equation at p must satisfy $L_p s_l - I_p L s_l = 0$, $l = 1, \cdots, \dim(S)$, or more explicitly,

$$\sum_{q \in D_p} \alpha_{pq} (s_l)_q - \sum_{q \in E_p} \beta_{pq} (L s_l)_q = 0 \qquad l = 1, \cdots, \dim(S). \tag{2.2}$$

We call this homogeneous linear system the *HODIE equations*. Usually the number of evaluation points is $\dim(S) - 8$ (one β is used for normalization), but a propitious choice of E_p can often reduce this number (equivalently, some β's are zero).

The existence of HODIE discretizations exact on polynomial spaces $\mathcal{P}_n$ (the space of all polynomials of degree n or less) is studied in [23] and [6]. It is known, for instance, that nontrivial HODIE discretizations of the Laplacian exist that are exact on $\mathcal{P}_8$, although several classes of such discretizations exact on $\mathcal{P}_7$ are known (these are given in this paper). This severe restriction does not occur in general, and HODIE discretizations exact on $\mathcal{P}_n$ with $n > 7$ have been found for equations with lower order derivative terms. When a HODIE discretization exact on $\mathcal{P}_n$ exists, then the order of its *truncation error*, $T_p u \equiv L_p u - I_p L u$, is usually $O(h^{n-1})$ for $u \in C^{n+1}$.

The use of nonpolynomial basis elements to model behavior such as singularities is also possible. The compact discretization of El-Mistikawy and Werle [12], for instance, which is based upon the representation of U as a piecewise exponential function, has proven successful in approximating solutions of two-point boundary value problems with boundary layers.

It is clear from (2.2) that if the coefficients of the PDE operator vary on R, then the difference equation coefficients α_{pq} and β_{pq} will vary with p. The computation of α_{pq} and β_{pq} may be done by a computer program for solving (2.2), and proper choice of basis functions s_l can reduce the cost of this operation. The computational complexity of the method for $S = \mathcal{P}_n$ is analysed in [24]. In that study, the number of arithmetic operations required to discretize and solve (by band elimination) a general linear problem on a rectangle with an $m \times m$ grid was found to be $2m^4 + 1079m^2$ for a fourth order HODIE method and $64(m+1)^4 + 320(m+1)^2$ for bicubic Hermite collocation, another fourth order method. Such operation counts only give a very rough comparison between the methods. For example, they neglect the costs of evaluating the coefficient functions of L as well as the right side g, which can become significant when these functions are even only moderately complicated. Also, it is assumed that appropriate locations for evaluation points in the HODIE method are known a priori, which is not always so. Recently however, Leventhal [20] has derived explicit formulas for the

coefficients of a fourth order compact discretization of $Lu = g$ with evaluation points the same as discretization points. For constant coefficient problems the operation counts reduce to $2m^4 + 13m^2$ for HODIE and $64(m+1)^4$ for collocation.

Finally, we note that the method is also applicable to problems with nonrectangular domains [6], general linear boundary conditions [5], and more than two space variables [21]. Similar techniques have been studied for two-point boundary value problems [22], and parabolic problems [1], [8].

3. Families of fourth order discretizations. We now return to problem (1.1). A simple change of variable yields the equation

$$\nabla^2 u + fu = g \tag{3.1}$$

on a rescaled domain R, and hence we assume $a = c$ without loss of generality. To get an $O(h^4)$ HODIE method we need $T_p s = 0$ for all $s \in \mathcal{P}_5$. Since the dimension of $\mathcal{P}_5$ is 21 we need 13 β's, although most will be zero by symmetry in our case. To simplify notation we let $h_x = h$, $h_y = \theta h$, $p = (0, 0)$ and drop the subscript p where no confusion results.

Define the HODIE discretization $P(h, \mu, \theta, f)$ by

$$L^{P,\mu}U = I^{P,\mu}g, \tag{3.2}$$

$$\begin{aligned} L^{P,\mu}U = (1/6h^2)\{&\alpha_0 U(0,0) + \alpha_1[U(h,0) + U(-h,0)] \\ &+ \alpha_2[U(0,\theta h) + U(0,-\theta h)] \\ &+ \alpha_3[U(h,\theta h) + U(-h,\theta h) + U(-h,-\theta h) + U(h,-\theta h)]\}, \\ I^{P,\mu}g = \beta_0 g(0,0) &+ \beta_1[g(\mu h,0) + g(-\mu h,0)] \\ &+ \beta_2[g(0,\mu\theta h) + g(0,-\mu\theta h)], \end{aligned}$$

where $0 < \mu \leqq 1$ is a parameter. Similarly we define $Q(h, \mu, \theta, f)$ by

$$L^{Q,\mu}U = I^{Q,\mu}g, \tag{3.3}$$

$$\begin{aligned} L^{Q,\mu}U = (1/6h^2)\{&\alpha_0 U(0,0) + \alpha_1[U(h,0) + U(-h,0)] \\ &+ \alpha_2[U(0,\theta h) + U(0,-\theta h)] \\ &+ \alpha_3[U(h,\theta h) + U(-h,\theta h) + U(-h,-\theta h) + U(h,-\theta h)]\}, \\ I^{Q,\mu}g = \beta_0 g(0,0) &+ \beta_1[g(\mu h,\mu\theta h) + g(-\mu h,\mu\theta h) \\ &+ g(-\mu h,-\mu\theta h) + g(\mu h,-\mu\theta h)]. \end{aligned}$$

The coefficients α_i and β_i may be different in the two cases. Their derivation is greatly simplified by the following lemma whose proof follows easily from symmetry.

LEMMA 3.1. *The* HODIE *discretizations $P(h, \mu, \theta, f)$ and $Q(h, \mu, \theta, f)$ of* (3.1) *are exact for all monomials $x^i y^j$ with at least one of i and j odd.*

THEOREM 3.2. *The discretization $P(h, \mu, \theta, f)$* of (3.1) *is exact on $\mathcal{P}_5$ for all $0 < \mu \leqq 1$, $\theta > 0$ if and only if*

$$\begin{aligned} \alpha_0 &= -10(\rho + \theta^2)/\theta^2 + 2(3 - \mu^2)F + (\mu^2 - 1)F^2/2, \\ \alpha_1 &= (5\theta^2 - \rho)/\theta^2 + \mu^2 F/2, \\ \alpha_2 &= (5\rho - \theta^2)/\theta^2 + \rho\mu^2 F/2, \\ \alpha_3 &= (\rho + \theta^2)/(2\theta^2), \end{aligned}$$

$$\beta_0 = [3\mu^2 - (\rho+1)/2]/(12\mu^2) + (\mu^2-1)F/12,$$
$$\beta_1 = 1/(12\mu^2),$$
$$\beta_2 = \rho/(12\mu^2),$$

where $F = h^2 f$ *and* $\rho = [12 + (\mu^2-1)F]/[12 + (\mu^2-1)\theta^2 F]$.

Proof. By Lemma 3.1 we need only satisfy $L^{P,\mu}s = I^{P,\mu}Ls$ for the functions 1, x^2-h^2, $y^2-(\theta h)^2$, $x^2(x^2-h^2)$, $y^2(y^2-(\theta h)^2)$ and $(x^2-h^2)(y^2-(\theta h)^2)$, plus a normalization. This leads to a system of seven linear equations, nonsingular for all $0 < \mu \leqq 1$, whose straightforward solution leads to the coefficients given above. QED

THEOREM 3.3. *The discretization* $Q(h, \mu, \theta, f)$ *of* (3.1) *is exact on* $\mathcal{P}_5$ *if and only if at least one of* $\mu = 1$, $\theta = 1$ *or* $f = 0$. *In these cases the coefficients are given uniquely by*

$Q(h, \mu, \theta, 0)$:
$$\alpha_0 = -10(1+\theta^2)/\theta^2,$$
$$\alpha_1 = (5\theta^2-1)/\theta^2,$$
$$\alpha_2 = (5-\theta^2)/\theta^2,$$
$$\alpha_3 = (1+\theta^2)/(2\theta^2),$$
$$\beta_0 = (6\mu^2-1)/(6\mu^2),$$
$$\beta_1 = 1/(24\mu^2),$$

$Q(h, \mu, 1, f)$:
$$\alpha_0 = -20 + (6-\mu^2)F - (1-\mu^2)F^2/2,$$
$$\alpha_1 = \alpha_2 = 4,$$
$$\alpha_3 = 1 + \mu^2 F/4,$$
$$\beta_0 = (6\mu^2-1)/(6\mu^2) - (1-\mu^2)F/12,$$
$$\beta_1 = 1/(24\mu^2), \text{ and}$$

$Q(h, 1, \theta, f)$:
$$\alpha_0 = -20 + 20\rho F,$$
$$\alpha_1 = (10\theta^2-2)/(1+\theta^2),$$
$$\alpha_2 = (10-2\theta^2)/(1+\theta^2),$$
$$\alpha_3 = 1 + \rho F,$$
$$\beta_0 = 10\rho/3,$$

where $F = h^2 f$ *and* $\rho = \theta^2/[2(1+\theta^2)]$.

Proof. By Lemma 3.1 we need only satisfy $L^{Q,\mu}s = I^{Q,\mu}Ls$ for $s = 1$, x^2-h^2, $y^2-(\theta h)^2$, $x^2(x^2-h^2)$, $(x^2-h^2)(y^2-(\theta h)^2)$, plus a normalization. (The function $y^2(y^2-(\theta h)^2)$ leads to the same HODIE equation as $x^2(x^2-h^2)$ in this case.) This leads to a system of six linear equations in the unknowns α_0, α_1, α_2, α_3, β_0 and β_1. If $f \neq 0$, $\theta \neq 1$ and $\mu \neq 1$ this system is inconsistent; otherwise, it is nonsingular and its straightforward solution yields the coefficients given above. QED

Several specific discretizations in these families have been considered previously for the case of the Poisson equation. For example, $P(h, 1, 1, 0)$ is given in [9] and $P(h, 1, \theta, 1)$ is given in [29]. Similar families were derived by Esch [13] using Fourier series methods.

When $\mu = 1$, no more than one evaluation of the right side g per grid point is needed to construct the system of finite difference equations. When $\mu = \frac{1}{2}$, an average of two evaluations per grid point are needed and for arbitrary μ up to five evaluations per grid point are required. However, in some cases a HODIE discretization with $0 < \mu < 1$ can produce an error small enough to warrant its use in spite of an increase in the number of evaluations of g. Some stencils of particular interest are given in Fig. 1.

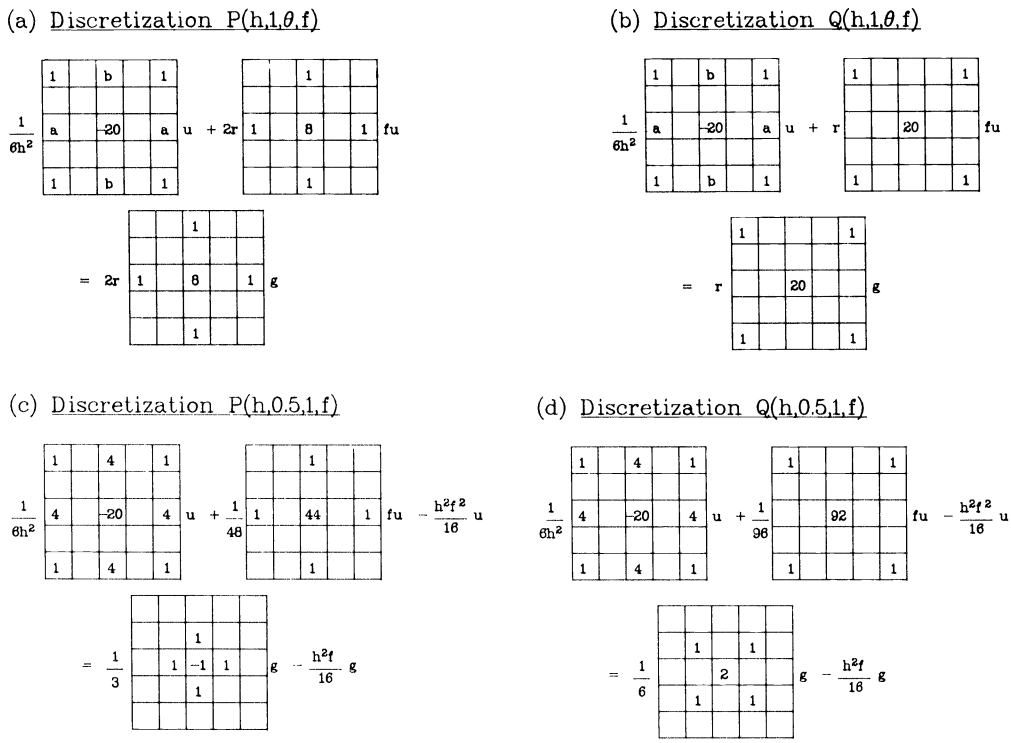


FIG. 1. Some discretizations in the fourth order families $P(h, \mu, \theta, f)$ and $Q(h, \mu, \theta, f)$, where $a = (10\theta^2 - 2)/(1 + \theta^2)$, $b = (10 - 2\theta^2)/(1 + \theta^2)$, and $r = \theta^2/(12(1 + \theta^2))$. Stencils are given at half-grid points. The discretizations in (a) and (b) also apply in the case of variable f; see § 7.

4. Error analysis. The first nonzero terms in the truncation error expansions of the schemes of Theorems 3.2 and 3.3 can be determined by the usual Taylor series analysis; details are given in [4]. Let $X^n Y^m$ denote the differentiation operator of order n in x and m in y (hence $X^n Y^m u = \partial^{n+m} u / \partial x^n \, \partial y^m$). We then have the following theorem.

THEOREM 4.1. *Suppose* $u \in C^8$. *The truncation errors of the* HODIE *discretizations* (3.2) *and* (3.3) *of* (3.1) *are then*

$$P(h, \mu, \theta, f): \quad (h^4/720)\{[(5\mu^2 - 2) + (\mu^4 - 1)h^2 f/6]X^6 + [(5\rho\mu^2 - 2) + \rho(\mu^4 - 1)\theta^2 h^2 f/6]Y^2 + 5[(\mu^2 - 1) - \theta^2\rho]X^4 Y^2 + 5[(\mu^2 - 1)\theta^2\rho - 1]X^2 Y^4\}u + O(h^6),$$

$$Q(h, \mu, \theta, 0): \quad (h^4/720)\{[5\mu^2 - 2](X^6 + Y^6) + 5[(6\theta^2 + 1)\mu^2 - (\theta^2 + 1)]X^4 Y^2 + 5\theta^2[(6 + \theta^2)\mu^2 - (\theta^2 + 1)]X^2 Y^4\}u + O(h^6),$$

$$Q(h,1,\theta,f): \quad (h^4/720)\{3(X^6+\theta^4Y^6)-25\theta^2(X^4Y^2+X^2Y^4)\}u+O(h^6),$$

$$Q(h,\mu,1,f): \quad (h^4/720)\{[5\mu^2-2](X^6+Y^6)+5[7\mu^2-2](X^4Y^2+X^2Y^4)\}u+O(h^6),$$

$$P(h,\mu,1,f): \quad (h^4/720)\{[5\mu^2-2](X^6+Y^6)+5[\mu^2-2](X^4Y^2+X^2Y^4)\}u+O(h^6).$$

It is easy to see from Theorems 3.2 and 3.3 that, for h sufficiently small, these discretizations are of *positive type* [7]; hence, the discretization error, $\max\{|U_p-u_p|, p\in G\}$, is of the same order as the truncation error. Thus, each method in the families presented here has fourth order accuracy for all $u\in C^6$.

For Laplace's equation ($f=g=0$), the discretizations $P(h,\mu,1,0)$ and $Q(h,\mu,1,0)$ reduce to the well-known "-20" formula, which has order of accuracy six for $u\in C^8$. The discretization $Q(h,\sqrt{0.2},1,f)$ also has increased accuracy in some cases. From Theorem 4.1 the truncation error of this discretization is $(-h^4/720)\nabla^6u+O(h^6)$. When $\nabla^2u=g$ this is $(-h^4/720)\nabla^4g+O(h^6)$ and so this discretization has sixth order accuracy for Poisson's equation whenever g is harmonic.

We now consider the value of $0<\mu\leqq 1$ which produces the smallest truncation error in the case of square grids ($\theta=1$). In each case the error takes the form

$$T_pu=(h^4/720)[r(\mu)(X^6+Y^6)+s(\mu)(X^4Y^2+X^2Y^4)]u+O(h^6).$$

Since we do not know anything about the derivatives of u in general, it is natural to seek that value of μ that minimizes the max norm of the 2-vector $[r(\mu), s(\mu)]$. This then determines a set of evaluation points which is optimal with respect to this choice of norm. A straightforward analysis shows that the minima occur at $\mu=\sqrt{0.3}$ for $Q(h,\mu,1,f)$ and at $\mu=1$ for $P(h,\mu,1,f)$. Table 1 gives values of the error norm for various choices of the point set, and indicates the relative accuracy we might expect from each discretization. The numerical experiments of § 4 support these conjectures.

TABLE 1

Values of an error norm for various choices of evaluation points.

Evaluation point set	$\max[\lvert r(\mu)\rvert, \lvert s(\mu)\rvert]$
$Q(h,\sqrt{0.3},1,f)$	0.5
$Q(h,1/2,1,f)$	2.5
$P(h,1,1,f)$	5.0
$P(h,1/2,1,f)$	8.75
$Q(h,1,1,f)$	25.0

5. Properties of the systems of difference equations. The following theorem summarizes the properties of the matrices generated by the HODIE discretizations of the previous section. We assume that an $(n+2)\times(m+2)$ grid is used and the nodes are numbered from left to right and bottom to top (the natural ordering).

THEOREM 5.1. *Let $-A$ be the $nm\times nm$ matrix generated by any of the* HODIE *discretizations of* § 3 *on a rectangle with Dirichlet boundary conditions. Then, for h sufficiently small, A is real and*

a) *symmetric,*

b) *an $m\times m$ block tridiagonal matrix, each block being itself an $n\times n$ symmetric tridiagonal matrix,*

c) *irreducible*[1],

d) *an L-matrix*[1],

[1] Definitions of these standard terms can be found in [30].

e) *weakly diagonally dominant*[1], *and*

f) *positive definite.*

For a matrix with these properties, much of the standard theory of iterative solution methods applies. In [2], numerical experiments demonstrate the effectiveness of the adaptive iterative algorithms of Hageman and Young [14], [15] for linear systems resulting from HODIE discretizations.

In addition, the symmetric block tridiagonal nature of these linear systems show that fast direct methods such as block cyclic reduction, fast Fourier transform and tensor product methods [11] may also be applied to HODIE discretizations. This combination has lead to extremely fast and highly accurate software for this problem [17], [18], [19].

6. Numerical experiments. We now report the results of substantial numerical experiments undertaken to support the claims made about the fourth order HODIE methods described above. In particular, we wish to assess the effect on accuracy of the placement of the evaluation points. The methodology used in the tests is based upon the framework presented in [25] and [27] and implemented in the PDE software evaluation system [3]. A complete list of all generated data is given in [4].

We select a set of ten Poisson and Helmholtz problems from the PDE population [28]. Each problem is defined on the unit square $[0, 1]\times[0, 1]$ and all boundary conditions are Dirichlet type. Some problems depend on parameters and counting each parameter value used we have 14 test cases. The problems are listed below.

Problem 3.

Operator:	Poisson
Solution:	$c(x^{a/2}-x)(y^{a/2}-y)$, $c = 1/[a^{a/(1-a)}-a^{1/(1-a)}]^2$
Case 3-1	$(a = 1.5)$
Features:	Solution has singular first derivatives along the lines $x = 0$ and $y = 0$.
Case 3-2	$(a = 2.5)$
Features:	Solution has singular second derivatives along the lines $x = 0$ and $y = 0$.

Problem 4.

Operator:	Poisson
Solution:	$3 \exp(x+y)xy(1-x)(1-y)$
Case 4-1	
Features:	Entire, slowly varying solution.

Problem 7.

Operator:	Poisson, $\nabla^2 u = 1$, $u = 0$ on ∂R
Solution:	Series solution computed with $\simeq 10^{-9}$ accuracy.
Case 7-1	
Features:	Singular second derivatives in solution at four corners.

Problem 8.

Operator:	Poisson
Solution:	$\phi(x)\phi(y)$, where $\phi(x) = 1$ for $x < 0.35$, $\phi(x) = p(x)$ for $0.35 \leqq x \leqq 0.65$, and $\phi(x) = 0$ for $x \geqq 0.65$. $p(x)$ is a quintic polynomial determined so that $\phi(x)$ has two continuous derivatives.
Case 8-2	
Features:	Solution has a wavefront with a discontinuous third derivative along a right angle through the center of the domain connecting two flat areas.

Problem 9.
Operator: Helmholtz, $\nabla^2 u - 100u = g$
Solution: $(\cosh(10x)/\cosh(10) + \cosh(ay)/\cosh(a))/2$
Case 9-2 $(a = 20)$
Features: Mild boundary layer in solution along $x = 1$ and $y = 1$.
Case 9-3 $(a = 50)$
Features: More pronounced boundary layer in solution along $x = 1$ and $y = 1$.

Problem 10.
Operator: Poisson
Solution: $xy(1-x)(1-y)\exp(-a[(x-0.5)^2 + (y-b)^2])$
Case 10-7 $(a = 50, b = 0.117)$
Features: Solution has sharp peak at (0.5, 0.117).

Problem 17.
Operator: Poisson
Solution: $\exp(-y^2 - [a(bx)^3/(1+(bx)^3)]^2) + \sin(x - y + 0.5)$
Case 17-2 $(a = 5, b = 3)$
Features: Very mild boundary layer in solution near $x = 0$.

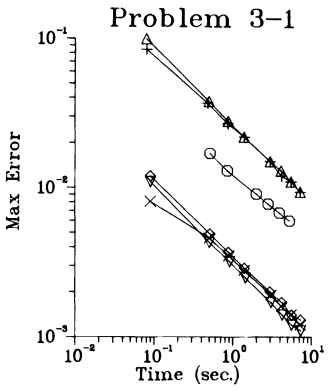
Problem 41.
Operator: Helmholtz, $\nabla^2 u - au = g$
Solution: $x(\pi - x)/2 - (4/\pi)$
$\times \sum_{k=1}^{b} \{[\sin((2k-1)x)\cosh((2k-1)(y-\pi/2))]$
$/[(2k-1)^3 \cosh((2k-1)\pi/2)]\}$
Case 41-3 $(a = 10.0, b = 25)$
Features: Solution has derivative singularities.

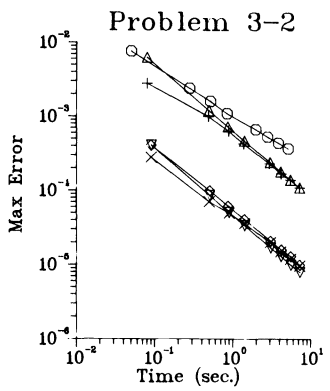
Problem 47.
Operator: Poisson
Solution: $(xy)^{a/2}$
Case 47-2 $(a = 5)$
Features: Solution has singular third derivatives along $x = 0$ and $y = 0$.

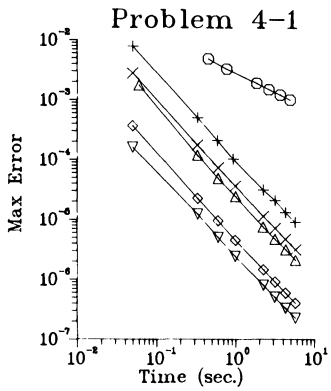
Problem 53.
Operator: Helmholtz, $\nabla^2 u - au = g$
Solution: $\cos(by) + \sin(b(x-y))$
Case 53-3 $(a = 10, b = 10)$
Features: Entire, oscillatory solution.
Case 53-4 $(a = 30, b = 10)$
Features: Entire, oscillatory solution.
Case 53-2 $(a = 10, b = \pi)$
Features: Entire, slowly varying solution.

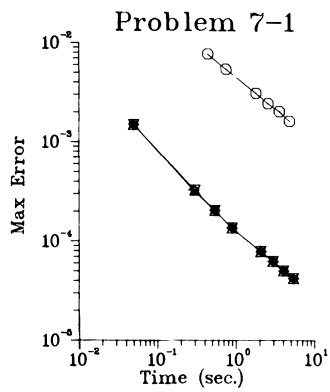
Each problem was solved using five different discretizations: $Q(h, \sqrt{0.3}, 1, f)$, $Q(h, \frac{1}{2}, 1, f)$, $P(h, 1, 1, f)$, $P(h, \frac{1}{2}, 1, f)$, $Q(h, 1, 1, f)$ and the standard five-point star. In each case the resulting linear system of difference equations was solved by the LINPACK symmetric band solver [10]. All codes tested are part of the March 1979 version of the ELLPACK system [26].

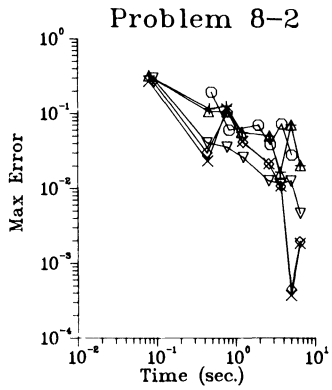
Each problem was run on a sequence of grids so that convergence behavior could be studied. All calculations were performed on a CDC 6500 computer in single precision arithmetic (14.3 decimal digits of precision). The programs used were written in ANSI standard FORTRAN (1966) and were compiled using the Minnesota FORTRAN compiler (MNF). We summarize the results in three parts and in Figure 2 display performance curves for each case. The graphs display the base 10 log of maximum error on the grid divided by the maximum value of the solution on the grid versus the base

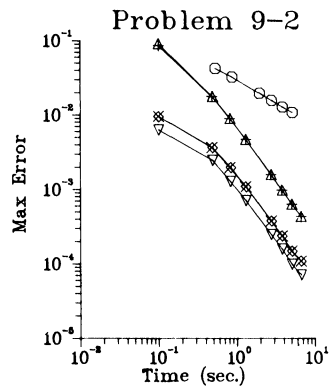

Problem 3-1
Max Error
Time (sec.)

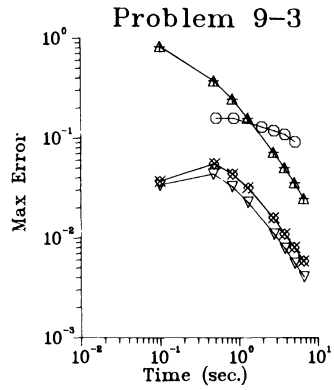

Problem 3-2
Max Error
Time (sec.)

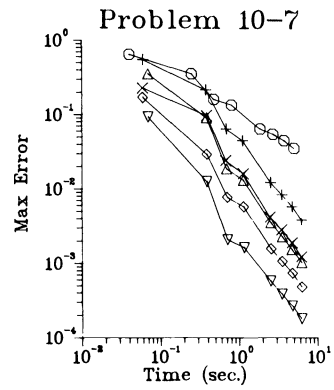

Problem 4-1
Max Error
Time (sec.)


Problem 7-1
Max Error
Time (sec.)


Problem 8-2
Max Error
Time (sec.)


Problem 9-2
Max Error
Time (sec.)


Problem 9-3
Max Error
Time (sec.)


Problem 10-7
Max Error
Time (sec.)

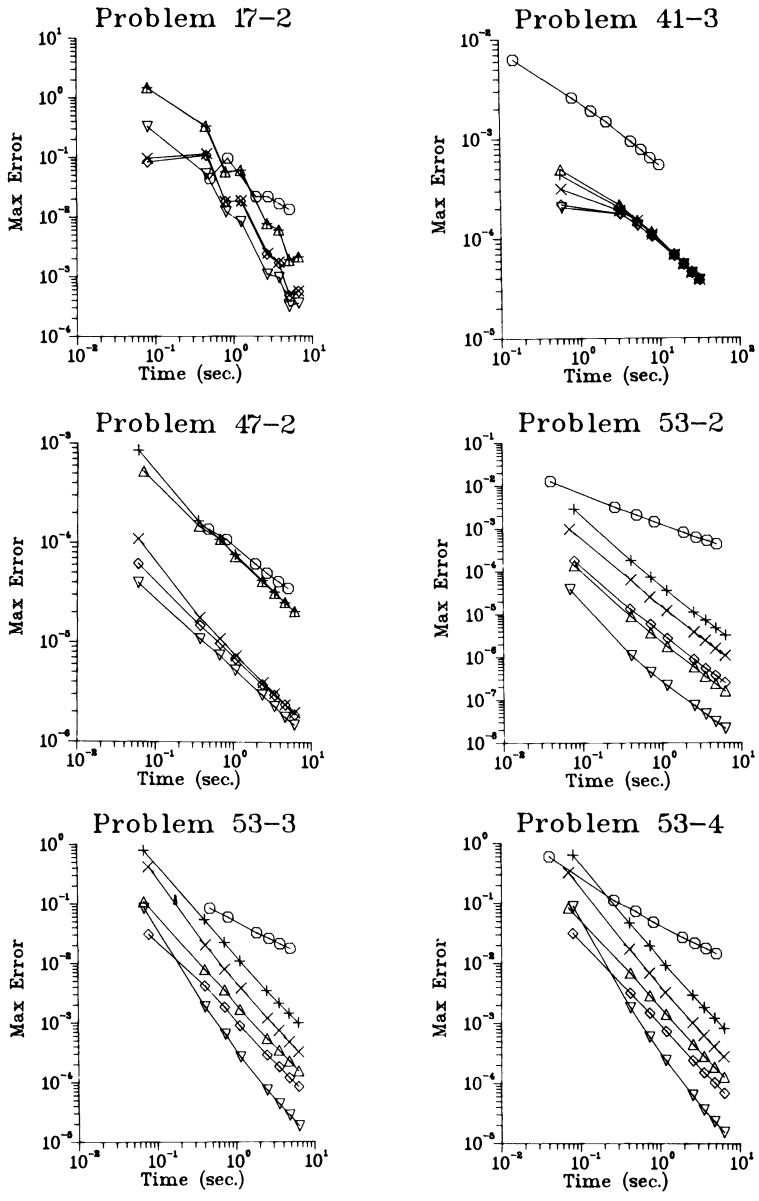


FIG. 2. Graphs of max-error on grid divided by max-solution on grid versus time in seconds on a CDC 6500 (log-log scale). Each point represents the solution of a problem on a uniform grid. The fourteen problems may be classified as those with smooth solutions (4-1, 53-2, 53-3, 53-4), those with smooth but difficult to compute solutions (9-2, 9-3, 10-7, 17-2), and those with discontinuous low order derivatives (3-1, 3-2, 7-1, 8-2, 41-3, 47-2). Each problem is solved using six separate discretizations: ○ 5-point star; △ HODIE $P(h, 1, 1, f)$; + HODIE $Q(h, 1, 1, f)$; × HODIE $P(h, 1/2, 1, f)$; ◇ HODIE $Q(h, \frac{1}{2}, 1, f)$; ▽ HODIE $Q(h, \sqrt{0.3}, 1, f)$.

10 log of the computation time in seconds. From these graphs we can easily assess which programs are the most efficient, that is, for a given amount of computation time, which code produces the smallest error. All problems fit in main memory and hence I/O time is not a factor.

Study 1. *Performance for smooth problems.* Problems 4 and 53 each have solutions in C^6 and hence the analysis of § 4 applies. For each problem the relative behavior of the HODIE methods was exactly as the error analysis indicated, with the optimal method producing the smallest error. $Q(h, \frac{1}{2}, 1, f)$ proved to be a good sub-optimal method while $Q(h, 1, 1, f)$ was the least efficient in each case. The difference between the good and poor choices of evaluation points was quite striking, consistently larger than an order of magnitude. As expected, each HODIE discretization performed better than the five-point star.

Study 2. *Performance for smooth but difficult problems.* Problems 9, 10 and 17 each have solutions that are in C^6, but also have troublesome behavior such as boundary layers or sharp peaks. For such problems the error analysis of § 4 still holds asymptotically, but one would expect the numerical methods to have trouble resolving the sharp gradients in the solution using the coarse uniform grids we used. This is indeed the case. In general, the relative efficiency of the eight methods tested is the same as in Study 1, although the accuracy is lower for each case and the differences between methods is much less as the difficulty of the problems increases.

Study 3. *Performance for nonsmooth problems.* The solutions of problems 3, 7, 8, 41, and 47 all possess singular or discontinuous derivatives of orders less than six and hence the error analysis of § 4 no longer applies. In the case of Poisson's equation, the HODIE methods tested here were also derived by Esch [13] using Fourier series methods. Esch's analysis indicates that the HODIE methods should still be effective for these problems. In each case we find that the order of convergence reduces to that of the five-point star; however, the absolute error of most of the HODIE methods is better and hence they are more efficient. We also observe that the HODIE methods fall into two groups of almost identical performance, the set $Q(h, \sqrt{0.3}, 1, f)$, $Q(h, \frac{1}{2}, 1, f)$, $P(h, \frac{1}{2}, 1, f)$ performing much better than $P(h, 1, 1, f)$ and $Q(h, 1, 1, f)$. See the results for cases 3-1, 3-2 and 47-2, which have solutions with discontinuous first, second and third derivatives, respectively.

We now rank the codes tested over the population of 14 cases based upon one performance indicator: time required to attain 0.05 per cent error. Since few programs attained this accuracy exactly on one of the grids employed, we fit a least squares line through the data for each method and problem and obtain the required value by interpolation. Table 2 gives the average rank and median time in seconds for each method. Applying the Friedman, Kendall and Babington-Smith test to this data [16, Ch. 7], we find that the average rank differences are significant with 95 per cent certainty if greater than 2.02. Conclusions at 80, 90, 95, 97.5 and 99 per cent confidence levels are given in Table 3.

TABLE 2
Time to attain 0.05 *per cent accuracy—average rankings of* 6 *methods for* 14 *problems.*

Method name	Average rank	Median time (sec.)
$Q(h, \sqrt{0.3}, 1, f)$	1.29	0.93
$Q(h, 1/2, 1, f)$	2.43	1.6
$P(h, 1/2, 1, f)$	2.79	2.1
$P(h, 1, 1, f)$	4.07	2.5
$Q(h, 1, 1, f)$	4.71	6.6
5-point star	5.71	100.0

TABLE 3
Ranking of methods based on time to achieve 0.05% *error.*

		WORSE				
		5-point star	$Q(h, 1, 1, f)$	$P(h, 1, 1, f)$	$P(h, 1/2, 1, f)$	$Q(h, 1/2, 1, f)$
B	$Q(h, \sqrt{0.3}, 1, f)$	99/4.42	99/3.42	99/2.78	/1.50	/1.14
E	$Q(h, 1/2, 1, f)$	99/3.28	97.5/2.28	80/1.64	/0.36	
T	$P(h, 1/2, 1, f)$	99/2.92	90/1.92	/1.28		
T	$P(h, 1, 1, f)$	80/1.64	/0.64			
E	$Q(h, 1, 1, f)$	/1.00				
R						

Entries are of the form top/bottom, where the top entry is the percent of confidence in the relative rankings of the two methods based upon the Friedman, Kendall and Babington–Smith test and the bottom entry is the average rank difference between the two methods. This difference is significant to 80%, 90%, 95%, 97.5% and 99% if greater than 1.62, 1.83, 2.02, 2.18 and 2.38 respectively.

7. Extensions to variable *f*. A simple extension of several discretizations of § 3 to the case of variable f is obtained from the following theorem.

THEOREM 7.1. *Let K_p and I_p be finite difference operators such that $K_p s = I_p \nabla^2 s$ for all $s \in \mathcal{P}_n$. Then the discretization*

$$L_p u \equiv K_p u + I_p(fu) = I_p g$$

of the equation $Lu \equiv \nabla^2 u + fu = g$ with variable f is also exact on $\mathcal{P}_n$. Moreover, both discretizations produce the same truncation and discretization errors for all $u \in C^{n+1}$.

Proof. Let $u \in C^{n+1}$. Then the truncation error of $L_p u = I_p g$ is $I_p Lu - L_p u = I_p(\nabla^2 u + fu) - [K_p u + I_p(fu)] = I_p \nabla^2 u - K_p u$ which is the truncation error of the discretization $K_p u = I_p \nabla^2 u$. Since the truncation errors are the same they are both exact on $\mathcal{P}_n$. In fact, they both produce the same estimates U of u and hence they must both have the same discretization errors. QED

COROLLARY 7.2. *Let $L^{P,1}u = I^{P,1}\nabla^2 u$ and $L^{Q,1}u = I^{Q,1}\nabla^2 u$ denote the discretizations $P(h, 1, \theta, 0)$ and $Q(h, 1, \theta, 0)$ of the Poisson equation as given in Theorems* 3.2 *and* 3.3. *Then the discretizations*

$$L^{P,1}u + I^{P,1}(fu) = I^{P,1}g,$$
$$L^{Q,1}u + I^{Q,1}(fu) = I^{Q,1}g$$

of the equation $\nabla^2 u + fu = g$ with variable f have truncation errors $O(h^4)$ and order of accuracy four for all $u \in C^6$.

As in § 3, a suitable change of variables yields discretizations of $au_{xx} + cu_{yy} + f(x, y)u = g$. Neither of these discretizations requires more evaluations of the right side function g than usual low order methods.

Inspecting the coefficients of these discretizations it is easy to see that the resulting system of linear equations is, for h sufficiently small and $f \leqq 0$, weakly diagonally dominant and an L-matrix. Unfortunately, unless f is constant, the matrix is not symmetric, although it is an $O(h^2)$ perturbation of one. It seems unlikely that a symmetric matrix could be generated without a substantial increase in the number of evaluation points.

We present one numerical example. Consider the Dirichlet problem for the equation $\nabla^2 u + fu = g$ on the unit square, where

$$f(x, y) = -[100 + \cos(2\pi x) + \sin(2\pi y)]$$

and

$$u = 0.31[\Psi(x)\Psi(y)\sin(\pi x)(y^2 - y)(1/(1+\phi^4)) - 0.5],$$

$$\phi = 4[(x-0.5)^2 + (y-0.5)^2], \qquad \Psi(z) = 5.4 - \cos(4\pi z).$$

The maximum value of the solution is about 3.3. The results obtained by solving this problem using the discretizations $P(h, 1, 1, f)$ and $Q(h, 1, 1, f)$ for various values of h are given in Table 4. Note that the error analysis of § 4 applies in this case and, as predicted there, the method using evaluation point set $P(h, 1, 1, f)$ has a smaller error than the method using the set $Q(h, 1, 1, f)$.

TABLE 4
Max error and observed convergence rate for a Helmholtz problem with variable f using discretizations $P(h, 1, 1, f)$ and $Q(h, 1, 1, f)$.

	$P(h, 1, 1, f)$		$Q(h, 1, 1, f)$	
h	max-error	rate	max-error	rate
.2500	7.1E-01	—	8.8E-01	—
.1667	1.2E-01	4.4	2.7E-01	3.0
.1250	3.4E-02	4.4	9.6E-02	3.0
.1000	1.3E-02	4.3	4.1E-02	3.9
.0833	7.1E-03	3.3	2.6E-02	2.5
.0625	2.3E-03	3.9	7.8E-03	4.2

8. Extensions—sixth order methods. With the aid of the fourth order families of § 3 it is easy to obtain discretizations of the Helmholtz equation on square computational cells ($\theta = 1$) with order of accuracy six.

THEOREM 8.1. *The* HODIE *discretization $L_p U = I_p g$ of the Dirichlet problem for the equation $\nabla^2 u + fu = g$ with constant $f \leqq 0$ defined by*

$$60L_p = 48L^{Q,1/2} + 8L^{P,1} + 4L^{Q,1}$$
$$60I_p = 48I^{Q,1/2} + 8I^{P,1} + 4I^{Q,1}$$

has order of accuracy six for $u \in C^8$.

Proof. The truncation error of this discretization is $[48T^{Q,1/2} + 8T^{P,1} + 4T^{Q,1}]u/60$. From Theorem 4.1 we see that this linear combination of truncation errors is $O(h^6)$. Since the resulting discretization is a convex combination of discretizations of positive type it also is of this type, and hence it has discretization error $O(h^6)$ for $u \in C^8$. QED

In stencil notation this scheme is written as

$$(6h^2)^{-1}$$

1		4		1
4		−20		4
1		4		1

$$U+(f/90)$$

1		1		1
1		82		1
1		1		1

$$U-h^2f^2U/20$$

$$=(1/360)$$

1		4		1
	48		48	
4		148		4
	48		48	
1		4		1

$$g-h^2f^2g/20,$$

where each box is of distance $h/2$ from its neighbors. In [18] the use of a sixth order HODIE discretization of the Helmholtz problem due to R. E. Lynch is evaluated for use with a fast direct equation solver. Although this discretization uses the same auxiliary points as the one above (they are identical when $f=0$), the coefficients of the scheme we present here are slightly simpler.

In general, a sixth order HODIE method requires 28 auxiliary points. The discretization that we have just presented for the Helmholtz equation uses only 13 (that is, 15 of the β's are zero). We next show that this number can be reduced further. We use the same superposition principle as in Theorem 8.1, looking for parameters ω, μ, ν such that

$$\omega T^{Q,\mu}+T^{P,\nu}=O(h^6). \tag{8.1}$$

where $T^{Q,\mu}$ and $T^{P,\nu}$ are the truncation operators of the discretizations $Q(h,\mu,1,f)$ and $P(h,\nu,1,f)$ respectively. The corresponding discretization,

$$[\omega L^{Q,\mu}+L^{P,\nu}]U=[\omega I^{Q,\mu}+I^{P,\nu}]g$$

has order of accuracy six for $u\in C^8$ while using only nine evaluation points. Applying the truncation error expansions of Theorem 4.1 we find that (8.1) is satisfied if and only if

$$\mu^2=4(1+\omega)/(15\omega), \qquad \nu^2=2(1+\omega)/15.$$

The requirement that $\mu\leqq 1$ and $\nu\leqq 1$ imposes the restriction $4/11\leqq\omega\leqq 13/2$. We have shown the following.

THEOREM 8.2. *The* HODIE *discretization $L_pU=I_pg$ of the Helmholtz equation on square computational cells ($\theta=1$) defined by*

$$[\omega L^{Q,\mu}+L^{P,\nu}]U=[\omega I^{Q,\mu}+I^{P,\nu}]g,$$

TABLE 5
Max error and observed convergence rate for the solution of a Helmholtz problem by four sixth order discretizations.

	Thm 8.1		Thm 8.2, $\omega = 4/11$	
h	max-error	rate	max-error	rate
.2500	3.4E-05	—	6.1E-05	—
.1667	2.8E-06	6.2	5.1E-06	6.1
.1250	5.2E-07	5.9	9.5E-07	5.8
.1000	1.4E-07	5.9	2.5E-07	6.1
.0833	4.5E-08	6.2	8.3E-08	6.1
.0625	8.1E-09	6.0	1.5E-08	6.0
	Thm 8.2, $\omega = 13/2$		Thm 8.2, $\omega = 2$	
h	max-error	rate	max-error	rate
.2500	5.7E-05	—	4.4E-05	—
.1667	4.7E-06	6.2	3.7E-06	6.1
.1250	8.7E-07	5.9	6.8E-07	5.9
.1000	2.3E-07	6.0	1.8E-07	6.0
.0833	7.6E-08	6.1	6.0E-08	6.0
.0625	1.4E-08	5.9	1.1E-08	5.9

where $\mu^2 = 4(1+\omega)/(15\omega)$, $\nu^2 = 2(1+\omega)/15$, $4/11 \leqq \omega \leqq 13/2$, *each have order of accuracy six for* $u \in C^8$.

Although these methods require only nine evaluation points, they do not appear to be as efficient as that of Theorem 8.1, which requires only an average of two evaluations of g per grid point. Table 5 displays the maximum error and observed convergence rate for the solution of the problem $\nabla^2 u - 5u = g$, with $u = \cos(2y) + \sin(2(x-y))$, using the discretization of Theorem 8.1 and three discretizations from the one-parameter family of Theorem 8.2.

9. Concluding remarks. We have used the HODIE approach with success in solving a class of elliptic boundary value problems on rectangles with Dirichlet boundary conditions, demonstrating that this technique provides an important tool for obtaining highly accurate solutions to problems of varying difficulty with only modest computational effort. A separate paper shows similar results in the presence of constant coefficient mixed boundary conditions. The practical applicability of the HODIE method in the general variable coefficient case has yet to be determined, since even the formulation of the HODIE equations (2.2) at each point can be a formidable computation.

Some nonlinear problems of the form $\nabla^2 u = g(u)$ have been solved using HODIE discretizations and fixed point iteration, although the straightforward applicability of the method is restricted to the case where the evaluation points are also discretization points. A similar restriction applies in the parabolic case.

Finally, the ability of the HODIE method to retain its high order accuracy on some simple nonrectangular domains has been demonstrated, and further investigation of this aspect of the method seems warranted.

REFERENCES

[1] A. E. BERGER, J. M. SOLOMON, AND M. CIMENT, *High order accurate tridiagonal difference methods for diffusion convection equations*, in Advances in Computer Methods for Partial Differential Equations—III, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1979, pp. 322–330.

[2] R. F. BOISVERT, *High order discretizations of the Helmholtz problem that admit iterative solution techniques*, in Advances in Computer Methods for Partial Differential Equations—III, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1979, pp. 1–7.

[3] R. F. BOISVERT, E. N. HOUSTIS AND J. R. RICE, *A system for performance evaluation of partial differential equations software*, IEEE Trans. Soft. Eng., SE-5 (1979), pp. 418–425.

[4] R. F. BOISVERT, *High Order Finite Difference Methods for Elliptic Boundary Value Problems*, Doctoral thesis, Purdue Univ., Indiana, 1979.

[5] ———, *High order compact difference formulas for elliptic problems with mixed boundary conditions*, in Advances in Computer Methods for Partial Differential Equations—IV, IMACS, New Brunswick, NJ, 1981.

[6] ———, *Attainable accuracy of compact discretizations of the Poisson equation*, in Elliptic Problem Solvers, M. Schultz, ed., Academic Press, New York, 1980.

[7] J. H. BRAMBLE AND B. E. HUBBARD, *A theorem on error estimation for finite difference analogues of the Dirichlet problem for elliptic equations*, in Contributions to Differential Equations, vol. 2, John Wiley, New York, 1963, pp. 319–340.

[8] M. CIMENT, S. H. LEVENTHAL AND B. C. WEINBERG, *The operator compact implicit methods for parabolic equations*, J. Comp. Phys., 28 (1978), pp. 135–166.

[9] L. COLLATZ, *The Numerical Treatment of Differential Equations*, Springer-Verlag, Berlin, 1960.

[10] J. J. DONGARRA, J. R. BUNCH, C. B. MOLER AND G. W. STEWART, LINPACK *User's Guide*, Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1979.

[11] F. W. DORR, *The direct solution of the discrete Poisson equation on a rectangle*, SIAM Rev., 12 (1970), pp. 248–263.

[12] T. M. EL-MISTIKAWY AND M. J. WERLE, *Numerical method for boundary layers with blowing—the exponential box scheme*, AIAA J., 16 (1978), pp. 749–751.

[13] R. ESCH, *High order difference approximations to Poisson's equation*, in Proceedings of a Harvard Symposium on Digital Computers and their Application—Annals Comp. Lab. Harvard Univ., vol. 31, 1961, pp. 81–102.

[14] R. G. GRIMES, D. R. KINCAID AND D. M. YOUNG, ITPACK 2.0 *User's Guide*, Center for Numerical Analysis, Univ. Texas at Austin, 1979.

[15] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1980.

[16] M. HOLLANDER AND D. A. WOLFE, *Nonparametric Statistical Methods*, John Wiley, New York, 1973.

[17] E. N. HOUSTIS AND T. S. PAPATHEODOROU, *Comparison of fast direct methods for elliptic problems*, in Advances in Computer Methods for Partial Differential Equations—II, R. Vichnevetsky, ed., IMACS, New Brunswick, NJ, 1977, pp. 46–52.

[18] ———, *A sixth order fast Helmholtz equation solver and its performance*, in Advances in Computer Methods for Partial Differential Equations—III, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, 1979, pp. 13–17.

[19] ———, *Algorithm FFT9: Fast solution of Helmholtz type partial differential equations*, ACM Trans. Math. Soft., 5 (1979), pp. 431–441.

[20] S. H. LEVENTHAL, *Two dimensional OCI methods*, in preparation.

[21] R. E. LYNCH, *$O(h^6)$ accurate finite difference approximation to solutions of Poisson's equation in three variables*, CSD-TR 230, Dept. of Comp. Sci., Purdue Univ., Indiana, April 19, 1977.

[22] R. E. LYNCH AND J. R. RICE, *A high order difference method for differential equations*, Math. Comp., 34 (1980), pp. 333–372.

[23] ———, *High accuracy finite difference approximations to solutions of elliptic partial differential equations*, Proc. Natl. Acad. Sci., 75 (1978), pp. 2541–2544.

[24] ———, *The* HODIE *method and its performance for solving elliptic partial differential equations*, in Recent Advances in Numerical Analysis, C. deBoor and G. Golub, eds., Academic Press, New York, 1978, pp. 143–175.

[25] J. R. RICE, *The algorithm selection problem*, in Advances in Computers, vol. 15, Rubicoff and Yovits, eds., Academic Press, New York, 1976, pp. 65–118.

[26] ———, ELLPACK: *A research tool for elliptic partial differential equations software*, in Mathematical Software—III, J. R. Rice, ed., Academic Press, New York, 1977, pp. 65–118.

[27] ———, *Methodology for the algorithm selection problem*, in Proc. IFIP TC2.5 Working Conference on Performance Evaluation of Numerical Software, L. D. Fosdick, ed., North-Holland, Amsterdam, 1979, pp. 301–307.

[28] J. R. RICE, E. N. HOUSTIS AND W. R. DYKSEN, *A population of linear, second order, elliptic partial differential equations on rectangular domains*, MRC Tech. Summary Rep. 2078–9, Mathematics Research Center, Univ. of Wisconsin, Madison, 1980.

[29] J. B. ROSSER, *Finite difference solutions of Poisson's equation in rectangles of arbitrary shape*, MRC Tech. Summary Rep. 1404, Mathematics Research Center, Univ. of Wisconsin, Madison, 1974.

[30] D. M. YOUNG, *Iterative Solutions of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0003 $01.00/0

PROPERTIES OF A VORTEX STREET OF FINITE VORTICES*

P. G. SAFFMAN† AND J. C. SCHATZMAN†

Abstract. Steady solutions of the Euler equations are calculated for an infinite array of vortices, consisting of two staggered parallel rows of identical vortices of finite area and uniform vorticity. These models are similar to the "vortex streets" studied theoretically by von Kármán and others, except that here vortices of finite rather than infinitesimal area are employed.

Key words. Kármán vortex street, laminar wake

1. Introduction. For a certain range of Reynolds number, a regular pattern of vortices is observed in the wake of a two-dimensional blunt body placed in a uniform stream. In his classic work, von Kármán modeled the problem with an infinite street of point vortices (see Kármán [4], [5], Kármán and Rubach [6]). This approximate approach has several limitations, among which are the infinite kinetic energy and difficulty in fitting the model to flow past a body.

To improve the model, the vortices are herein allowed to be of finite area, but uniform vorticity. An integrodifferential equation is then solved to obtain the steady shapes of the vortices. This paper is a report of the first part of a study of the wake flow problem, and describes only properties of the steady solutions for the infinite vortex array. Subsequent papers will report on a stability analysis of the steady states, and on relating this model to the wake flow problem.

2. Formulation. Consider an infinite array of uniform two-dimensional vortices, consisting of one row of identical vortices of area A and strength $-\Gamma$ with centroids at positions $x = 0, \pm l, \pm 2l, \pm 3l, \cdots, y = 0$, and of a second row of identical vortices of area A and strength $+\Gamma$ with centroids at $x = d, d \pm l, d \pm 2l, d \pm 3l, \cdots, y = -h$. Let there be superimposed a uniform flow U_s in the x direction, at infinity, as in Fig. 1. It is assumed

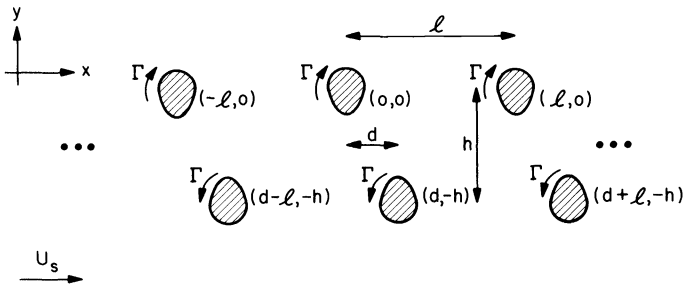


FIG. 1. *The configuration of the fully infinite vortex street with arbitrary stagger.*

that the flow is inviscid, incompressible, two-dimensional and, outside the vortices, irrotational. This paper deals with steady flows of this kind, principally with $d/l = 0.5$ (for values other than 0 and 0.5 and for the symmetric periodic cases considered here, the street does not move parallel to itself; see Rosenhead [9]).

* Received by the editors September 10, 1980, and in revised form February 18, 1981.

† Applied Mathematics Department, California Institute of Technology, Pasadena, California 91125.
This work was supported by the U.S. Department of Energy (Office of Basic Energy Sciences).

The complex potential outside the vortices can be written (with obvious notation)

$$(2.1)\quad w(z)=\frac{i\Gamma}{2\pi A}\left\{\iint_{\Sigma_1}\log\sin\frac{\pi}{l}(z-z')\,dx'\,dy'-\iint_{\Sigma_2}\log\sin\frac{\pi}{l}(z-z')\,dx'\,dy'\right\}+U_s z,$$

where Σ_1, Σ_2 refer to the cross-sections of one vortex in the upper and lower rows, respectively. By applying a Green's theorem, the complex velocity can be written as a line integral around the boundaries of the vortices:

$$(2.2)\quad u+iv=\frac{\Gamma}{2\pi A}\left\{\int_{\Sigma_1}\log\left|\sin\frac{\pi}{l}(z-z')\right|dz'-\int_{\Sigma_2}\log\left|\sin\frac{\pi}{l}(z-z')\right|dz'\right\}+U_s.$$

The requirement that the velocity field be tangent to the boundary of the vortices then determines the steady shapes of the vortices, apart from the scaling, as a function of the three dimensionless parameters $d/l\equiv\mu$, $h/l\equiv\kappa$, and $A/l^2\equiv\alpha$.

To simplify the calculations, the vortices in the two rows are assumed to have identical shapes, differing only in position and orientation. There are two reasonable choices of symmetry: invariance to reflection about the line $y=-h/2$ (the streamwise axis centered between the two rows) and a suitable x translation, and similarly with an *additional* reflection about $x=0$ (the vertical axis of one of the vortices). In both cases, it is sufficient to satisfy (2.2) along the boundary of a vortex in either row. For vortices of streamwise symmetry, these two cases are equivalent.

The second choice was picked, giving in place of (2.2)

$$(2.3)\quad u+iv=\frac{\Gamma}{2\pi A}\left\{\int_{\Sigma_1}\log\left|\sin\frac{\pi}{l}(z-z')\sin\frac{\pi}{l}(z+z'-\mu l+i\kappa l-2\bar{z})\right|dz'\right\}+U_s,$$

where $\bar{z}$ denotes the centroid of Σ_1 and is for now assumed to be arbitrary. The first symmetry could also have been considered, but since only streamwise symmetric solutions were found using (2.3), this was not attempted.

Three quantities of interest are the propagation velocity U_s of the array, the kinetic energy of the fluid, and the momentum transport. They are needed to relate this model to the wake flow problem. Specifically, the following quantities are defined:

$$(2.4)\quad T=\frac{1}{2l}\int_{-\infty}^{+\infty}\int_{-l/2}^{l/2}(u'^2+v^2)\,dx\,dy,\qquad D'=-\frac{1}{2}\operatorname{Im}\int_{-i\infty}^{+i\infty}(u'-iv)^2\,dz,$$

where the contour integral in the expression for D' is along any contour from $y=-\infty$ to $y=+\infty$ which does not pass through a vortex. Here T is the kinetic energy of the fluid per unit length (streamwise), D' is essentially the momentum flux of the fluid in the streamwise direction with the contribution from the vortices themselves omitted, and $u'=u-U_s$ is the x velocity relative to the free stream. Dimensionless values of these quantities are defined as follows:

$$(2.5)\quad \hat{U}_s(\alpha,\kappa)=\frac{l}{\Gamma}U_s,\quad \hat{T}(\alpha,\kappa)=\frac{l}{\Gamma^2}T,\quad \hat{D}'(\alpha,\kappa)=\frac{l}{\Gamma^2}D'.$$

3. Numerical method. Two successful numerical schemes were employed to calculate the steady vortex shapes, one using Newton's method in a straightforward manner and the other using an ad hoc iterative scheme (Pierrehumbert and Widnall [8]). Only solutions for vortices symmetric in the streamwise direction were computed. The first numerical scheme allowed solutions lacking this symmetry, but none were found (although an exhaustive search was not conducted). For purposes of the calculations, Σ_1 was taken to be the vortex with centroid at the origin.

The condition that the vortex boundary be a streamline may be written:

$$\text{Im}\left\{\frac{\partial z^*}{\partial s}(u+iv)\right\}=0 \tag{3.1}$$

where the derivative is taken along the boundary. The boundary of Σ_1 is parameterized using polar coordinates:

$$\begin{aligned} &z=R(\tilde{\vartheta})\,e^{i\vartheta(\tilde{\vartheta})}, && 0\leqq\tilde{\vartheta}\leqq 2\pi,\\ &\vartheta(\tilde{\vartheta})=\vartheta_0+\tilde{\vartheta}-\delta\sin 2\tilde{\vartheta}, && 0\leqq\delta<\tfrac{1}{2},\\ &R(\tilde{\vartheta})=\tfrac{1}{2}a_0+\sum_{j=1}^{N}(a_j\cos j\tilde{\vartheta}+b_j\sin j\tilde{\vartheta}). \end{aligned} \tag{3.2}$$

Here ϑ_0, δ are parameters which permit limited adjustment of the scaling of ϑ in regions of high curvature of the boundary, so as to improve the rate of convergence of the Fourier series for R.

Equation (3.1) is evaluated at uniformly spaced values of $\tilde{\vartheta}$:

$$\tilde{\vartheta}=\tilde{\vartheta}_j\equiv\frac{2\pi j}{2N+1},\qquad j=0,1,2,\cdots,2N. \tag{3.3}$$

This gives $2N+1$ equations for the $2N+2$ unknowns $a_0,\cdots,a_N, b_1,\cdots,b_N, U_s$. An additional equation comes from fixing the size of the vortex, e.g.

$$R(\varphi)=\text{fixed}, \tag{3.4}$$

where φ is some fixed angle. However, the resulting system is singular, because (3.1) is invariant to a translation of z. The specification is completed by fixing the centroid of the vortex at the origin, i.e.

$$\bar{x}+i\bar{y}\equiv\frac{1}{3A}\int_{\Sigma_1}R^3\,e^{i\vartheta}\,d\vartheta=0. \tag{3.5}$$

The resulting $2N+4$ real equations for the $2N+2$ unknowns are not independent and the problem is handled by using the trick (Chen and Saffman [1]) of solving (3.4) combined with the $2N+1$ equations which arise from the discretization of

$$\text{Im}\left\{\frac{\partial z^*}{\partial s}(u+iv)\right\}+f_1(\tilde{\vartheta})\bar{x}+f_2(\tilde{\vartheta})\bar{y}=0, \tag{3.6}$$

where f_1 and f_2 are more or less arbitrary nontrivial functions chosen to ensure that the Jacobian of the system is nonzero for the solutions that satisfy (3.5). The choice of $\tilde{\vartheta}$ and $\tilde{\vartheta}^2$ respectively for f_1 and f_2 was found to be satisfactory. Also, the $\bar{z}$ appearing in (2.3) was dropped to simplify the equations slightly.

If $\delta=0$ then, as the limiting case of touching vortices (in each row) is approached, the curvature of R with respect to ϑ becomes large near $\vartheta=0$ and $\vartheta=\pi$, and hence convergence of the series for R becomes slow. For large κ, $\vartheta_0=0$, $\delta=0.4999$ were used, which concentrates mesh points in these regions of large curvature and hence smooths out R as a function of $\tilde{\vartheta}$. This procedure works well for roughly $\kappa>0.36$; below this point, the vortices are too irregularly shaped for this simple technique to be useful. To speed up some computations, $\vartheta_0=-\pi/2$, $\delta=0$ were used, and the vortices were assumed to be streamwise symmetric. In this case, the Fourier series for R contains only the cosine terms, and hence the number of unknowns is reduced. Specifically, (3.6) is evaluated at the mesh points (3.3) for $j=1,2,\cdots,N$, and the

unknowns are $a_0, a_1, \cdots, a_{N-1}, U_s$; the remaining Fourier coefficients are taken to be zero.

Integrals for velocity, centroid, and area were evaluated using the trapezoidal rule, with care taken, in the former case, to preserve formal infinite order accuracy (see appendix A). Initial guesses for Newton's method were provided by using one Euler step to advance from the previous converged solution, starting with small vortices and gradually increasing the size by continuation in the parameter (3.4). However, convergence was observed to be insensitive to the initial guess. Accuracy was ensured by requiring that the highest order Fourier coefficients be sufficiently small, and by checking that increasing N had sufficiently small effect on the results. Values of N from 50 to 400 were found to be adequate for 5-digit precision in the final results. Each iteration required roughly from 1 to 25 seconds using a CDC Cyber 203 computer (64 bit floating point).

The second numerical scheme is essentially a scalar approximation to Newton's method. Consider the variation of the stream-function ψ along the boundary of the vortex at some intermediary stage in the calculation. It is assumed that most of the change in ψ on the boundary due to a perturbation of the boundary comes from the fact that it is computed at a different point, rather than the fact that the flow field is changed. Again, a polar coordinate representation for the boundary is employed:

$$z(\tilde{\vartheta}) = R(\tilde{\vartheta})\, e^{i\vartheta(\tilde{\vartheta})}, \qquad \vartheta(\tilde{\vartheta}) = -\frac{\pi}{2} + \tilde{\vartheta} - \delta \sin 2\tilde{\vartheta}, \tag{3.7}$$

but here streamwise symmetry of each vortex is assumed ab initio, and the unknowns are taken to be U_s and the values of R at

$$\tilde{\vartheta} = \tilde{\vartheta}_j \equiv \frac{\pi j}{N}, \qquad j = 0, 1, 2, \cdots, N. \tag{3.8}$$

The iteration performed can be written

$$R_j^{(n+1)} = R_j^{(n)} - \rho \frac{\psi_j^{(n)} - \psi_0^{(n)}}{\psi_{R,j}^{(n)}}, \qquad j = 1, 2, \cdots, N. \tag{3.9}$$

The numerator of the quotient is obtained by integration of the velocity using the trapezoidal rule:

$$\psi_j^{(n)} - \psi_{j-1}^{(n)} \tag{3.10}$$
$$\approx \frac{\pi}{2N}\left\{\left(\frac{\partial\psi}{\partial R}\right)_j^{(n)}\left(\frac{\partial R}{\partial\tilde{\vartheta}}\right)_j^{(n)} + \left(\frac{\partial\psi}{\partial\vartheta}\right)_j^{(n)}\left(\frac{\partial\vartheta}{\partial\tilde{\vartheta}}\right)_j^{(n)} + \left(\frac{\partial\psi}{\partial R}\right)_{j-1}^{(n)}\left(\frac{\partial R}{\partial\tilde{\vartheta}}\right)_{j-1}^{(n)} + \left(\frac{\partial\psi}{\partial\vartheta}\right)_{j-1}^{(n)}\left(\frac{\partial\vartheta}{\partial\tilde{\vartheta}}\right)_{j-1}^{(n)}\right\},$$

and $\psi_R \equiv \partial\psi/\partial R$ is of course a velocity component. The relaxation factor ρ is adjusted empirically for optimum convergence (typically $0.5 \leqq \rho \leqq 2$). Cycles of this iteration alternate with an update of U_s:

$$U_s^{(n+1)} = U_s^{(n)} - \rho \frac{\nabla\psi^{(n)}}{y_N - y_0}, \tag{3.11}$$

where $\nabla\psi^{(n)}$ is obtained by integrating $d\psi$ around the half revolution from $\tilde{\vartheta}_0$ to $\tilde{\vartheta}_N$. The velocity was calculated essentially as before (but see appendix A). As the final step of the iteration, the vortex is shifted in the y direction to put the centroid at the origin, new values of R_j being computed via interpolation.

This method has apparent advantages over Newton's method, namely, its simplicity and its speed per iteration (the cost is $O(N^2)$ per iteration versus $O(N^3)$ for Newton's method). However, highly unpredictable dependence on the initial guess and poor convergence rate in some cases is the penalty. Convergence is geometric with observed convergence factor ranging from about 0.15 for very small vortices to about 0.85 for large vortices. The method failed entirely to converge for very large vortices with small κ. Furthermore, more points are required for the same accuracy as compared with the previous method. Instability is controlled by limiting the maximum value of ρ and was not a difficulty. Values of N ranging from 50 to 400 were found to be adequate. These calculations were performed using a DEC VAX11/780 computer (64 bit floating point).

The energy can be obtained by a single contour integral over the vortex boundary requiring $O(N)$ operations (see appendix B). The momentum integral was computed by applying the trapezoidal rule over a finite contour passing between two neighboring vortices, and extending to regions where the flow is essentially a uniform stream (the perturbation decays exponentially with y). Romberg integration was used to obtain sufficient accuracy for this calculation.

4. Circular vortex approximation. The vortices of small area for the exact problem are nearly circular, and for precisely circular vortices the propagation velocity, momentum transport and energy calculations can be done analytically. The former two calculations lead to the same result as for point vortices (Goldstein [3]), since the flow field outside a uniform circular vortex is identical to that of a point vortex of the same circulation. For $d = l/2$, $\mu = \frac{1}{2}$,

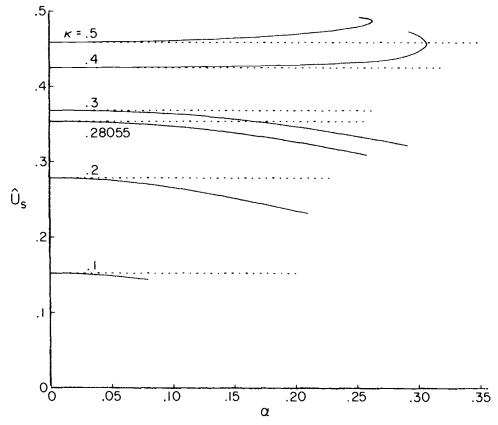
$$U_s = \frac{\Gamma}{2l}\tanh\frac{\pi h}{l}, \qquad D' = \frac{\Gamma^2}{2\pi l} - \frac{\Gamma h}{l}U_s, \tag{4.1}$$

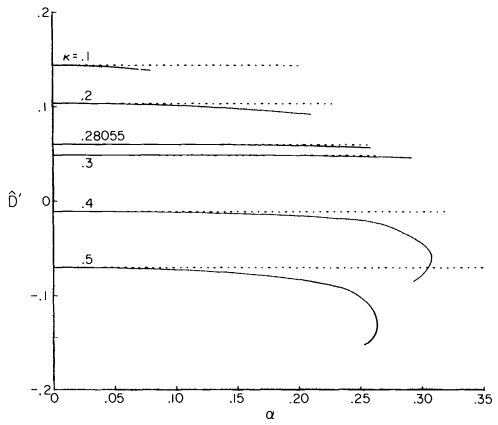
and the energy can be evaluated exactly by integration of the kinetic energy density:

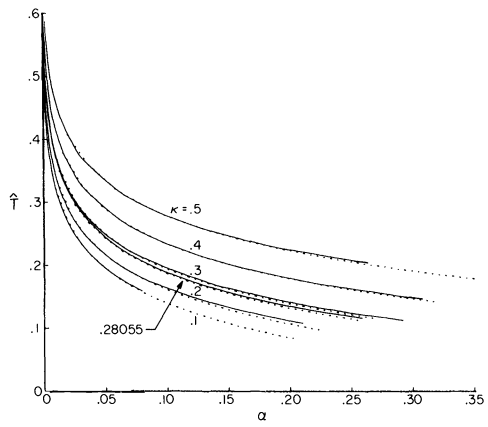
$$T = \frac{\Gamma^2}{2\pi l}\left[\log\left\{\left(\frac{l^2}{\pi A}\right)^{1/2}\cosh\frac{\pi h}{l}\right\} + \frac{1}{4}\right]. \tag{4.2}$$

Note that the circular vortex model loses physical validity when the vortices overlap, namely for $\alpha > \min\{(\pi/4), (\pi/4)(\kappa^2 + \frac{1}{4})\}$.

5. Results of the calculations. Figures 2–5 show the calculated values of $\hat{U}_s$, $\hat{T}$, and $\hat{D}'$ for the exact problem, accompanied by the corresponding results for the circular vortex approximation. The curves were traced by using as continuation parameter the quantity a which is the ratio of the x semi-axis of the vortices to l. As is evident in Fig. 5, a solution of simultaneous maximum area and minimum energy exists for each κ in accordance with Kelvin's variational principle for the steady states (Saffman and Szeto [10]). This limit is a contour in the (κ, α) plane, as depicted by curve 1 in Fig. 6. For roughly $\kappa > 0.36$, further increase in a results in a decrease in area and increase in energy, up to the point $a = 0.5$, where the vortices in each row touch. This limiting case is depicted by curve 2 in Fig. 6. Thus, between curves 1 and 2 there are two different configurations for a given (κ, α). Presumably, the solution curves could be continued beyond $a = 0.5$ by considering two adjacent distorted vortex layers in place of discrete vortices, but this was not done. Similar behavior has been observed for the linear vortex array, which in fact corresponds to the limit $\kappa \to \infty$ (Saffman and Szeto [10]). Quite different behavior was observed for κ smaller than about 0.36. In this case, the calculations indicate that the parameter a approaches a limiting value less than 0.5. To







FIGS. 2–4. *Values of $\hat{U}_s$, $\hat{D}'$ and $\hat{T}$ for the fully infinite vortex array. Solid lines denote the calculated values for the exact problem and dashed lines denote the circular vortex approximation.*

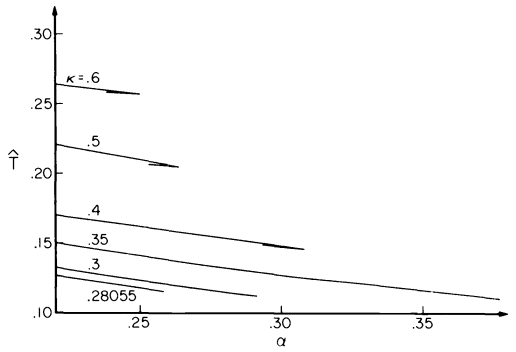


FIG. 5. *Expanded plot of energy $\hat{T}$ versus area α, showing the nonuniqueness, maximum area and minimum energy.*

check that this phenomenon was not dependent on the choice of the horizontal semi-axis for continuation, the vertical semi-axes were also used as continuation parameters. In all cases, the numerical evidence indicates that as the vortex size increases, vortices in each row protrude between vortices in the other row, and the solutions branches terminate when vortices in *opposite* rows approach and finally meet. Here there is no turnaround in area or energy, but maximum area and minimum energy occur at the limiting point of the solution branch. This behavior is similar to that observed for a pair of counter-rotating vortices as studied by Pierrehumbert [7]. The calculations for $\kappa < 0.36$ and for large area were costly, and an accurate calculation of the limiting case was not attempted. For this region, the corresponding segment of curve 1 in Fig. 6 should be regarded as a lower bound.

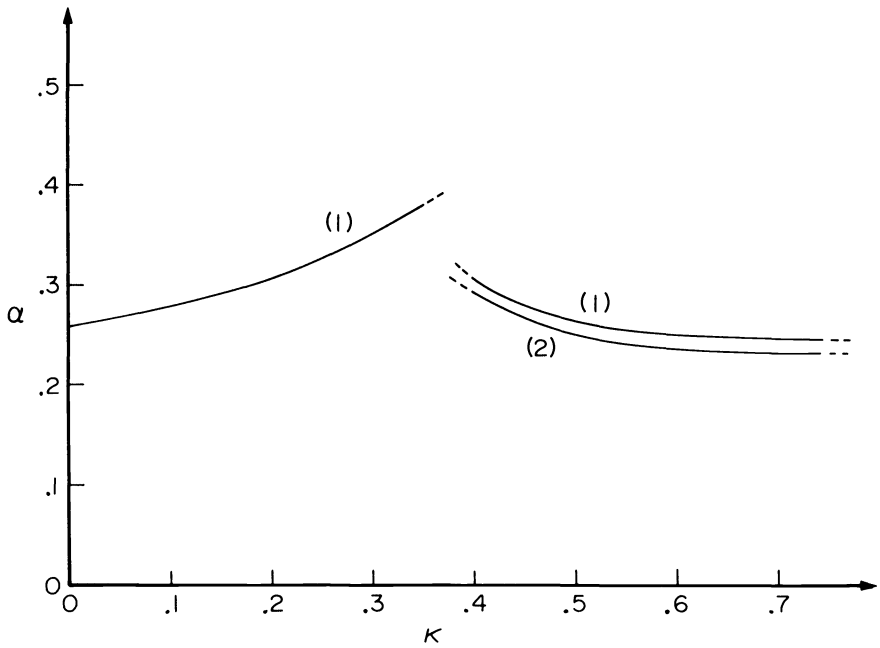


FIG. 6. *A plot of the area α versus spacing ratio κ plane. Curve 1 denotes the maximum area for a given spacing ratio. The segment corresponding to smaller κ should be regarded as a lower bound. Above curve 2, there are two solutions for a given pair (α, κ).*

Presumably, there exists a critical value of κ which divides the regions of the two types of limiting behavior. Due to cost limitations, it was not possible to determine accurately this critical value. However, it is believed to lie within the range from 0.35 to 0.365.

A geometric observation of relevance is that for small areas and for $\kappa > 0.36485$, the vortices are longer in the streamwise direction than in the transverse, and the converse for κ less than this critical value. The exact dividing value for infinitesimal area is the solution to $\cosh^2 \pi\kappa = 3$, which may be demonstrated using an elliptical vortex approximation (as has been applied to the linear vortex array; see Saffman and Szeto [10]). This is in good agreement with the numerically estimated large area critical value of κ as discussed above, but there is no evidence to suggest that the large area critical value is precisely the infinitesimal area critical value.

Figures 7–13 are plots showing the vortex shapes and the velocity fields. The apparent good qualitative agreement with experimental observations (for example,

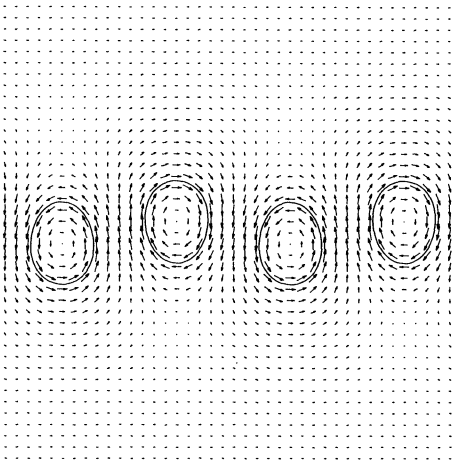


FIG. 7. $\kappa = 0.1$, $\alpha = 0.07948$.

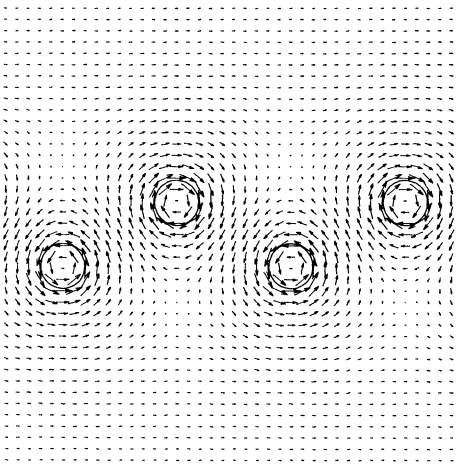


FIG. 8. $\kappa = 0.28055$, $\alpha = 0.03011$.

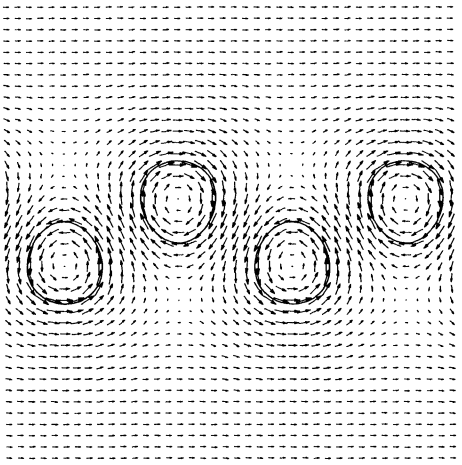


FIG. 9. $\kappa = 0.28055$, $\alpha = 0.09382$.

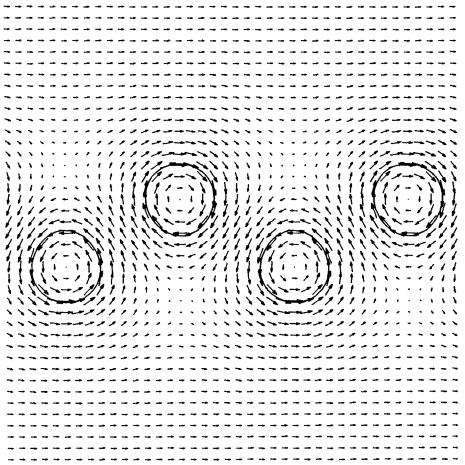


FIG. 10. $\kappa = 0.3$, $\alpha = 0.07484$.

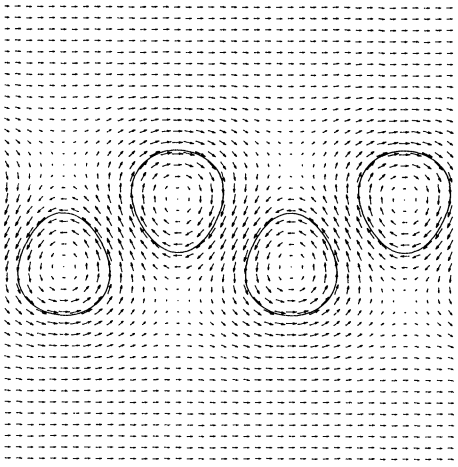


FIG. 11. $\kappa = 0.3$, $\alpha = 0.1409$.

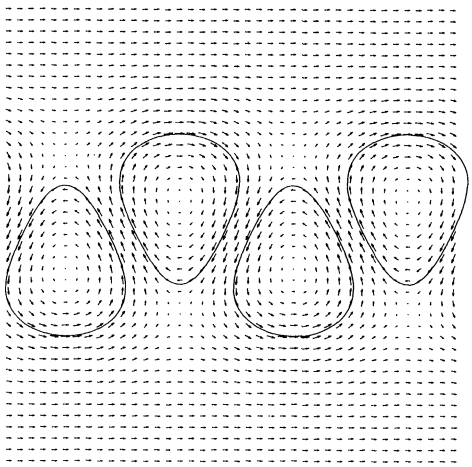


FIG. 12. $\kappa = 0.3$, $\alpha = 0.2541$.

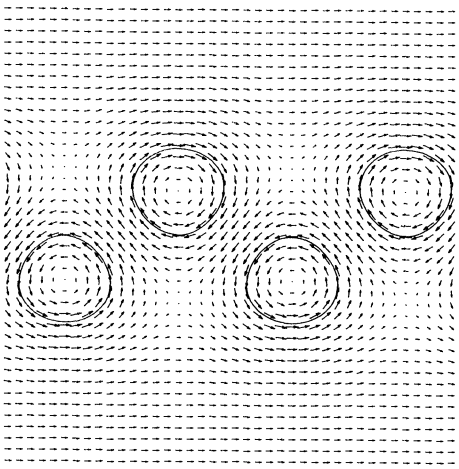


FIG. 13. $\kappa = 0.4$, $\alpha = 0.1188$.

FIGS. 7–13. *Plots of the vortex shapes and velocity field. Arrow length is proportional to the speed of the fluid at midpoint.*

Davies [2]) seems to provide some justification for the assumptions implicit in the forthcoming application of this model to the wake behind bluff bodies and its stability.

Appendix A. Calculation of the velocity. To evaluate the integrals in (2.2) it is useful to write

$$\int_{\Sigma_1} \log\left|\sin\frac{\pi}{l}(z-z')\right| dz' = \int_0^{2\pi} \log\left|\sin\frac{\pi}{l}(z-Z)\right|\left|\frac{dZ}{d\Theta}\right| d\Theta, \tag{A1}$$

where $Z \equiv z|_{\vartheta=\Theta}$ and $dZ/d\Theta$ can be determined from (3.2) as finite Fourier series in Θ. Hence, the integrand is 2π-periodic, and the trapezoidal rule gives formally infinite order accuracy—provided that the integrand has infinitely many derivatives, i.e., that z

is not on the boundary of the vortex. Unfortunately, z *is* on the boundary for one of the integrals when (3.6) is solved to obtain the vortex shapes.

To preserve accuracy in this case, the following trick is used (suggested by Dr. B. Fornberg):

$$\text{(A2)}\qquad \int \log\left|\sin\frac{\pi}{l}(z-z')\right| dz' = \int \log\left|\frac{\sin\frac{\pi}{l}(z-z')}{\frac{\pi}{l}(z-z')}\right| dz' + \int \log\left|\frac{\pi}{l}(z-z')\right| dz'.$$

The first integral presents no difficulties, and the second can be calculated as follows:

$$\begin{aligned}\int \log\left|\frac{\pi}{l}(z-z')\right| dz' &= \int \log\frac{\pi}{l}(z-z')\,dz' - \int \log\frac{z-z'}{|z-z'|}\,dz' \\ &= -i\int arg\,(z-z')\,dz' \\ &= -i\int_{\tilde{\vartheta}}^{\tilde{\vartheta}+2\pi} arg(z-Z)\frac{dZ}{d\Theta}\,d\Theta \qquad \text{(A3)} \\ &= -i\int_{\tilde{\vartheta}}^{\tilde{\vartheta}+2\pi} [arg(z-Z)-\tfrac{1}{2}\Theta]\frac{dZ}{d\Theta}\,d\Theta - \tfrac{1}{2}i\int_{\tilde{\vartheta}}^{\tilde{\vartheta}+2\pi}\Theta\frac{dZ}{d\Theta}\,d\Theta \\ &= -i\int_{\tilde{\vartheta}}^{\tilde{\vartheta}+2\pi}\left\{[arg(z-Z)-\tfrac{1}{2}\Theta]\frac{dZ}{d\Theta}-\tfrac{1}{2}Z\right\} d\Theta - i\pi z(\tilde{\vartheta}),\end{aligned}$$

where the *arg* function is taken so that the integrand is 2π-periodic.

For the second numerical scheme, R_ϑ was approximated by a fourth order centered finite difference formula, and z_ϑ then obtained from this. Hence, the integrals were approximated to third order.

Appendix B. Energy calculation. The calculation of the kinetic energy for the infinite vortex street proceeds nearly identically as for the infinite linear array (Saffman and Szeto [10]). The result (with unit density) may be written

$$\text{(B1)}\qquad T = \frac{1}{4}\frac{\Gamma}{A}\int_{\Sigma_1} R^2\frac{\partial\psi}{\partial n}\,ds - \frac{\Gamma}{2\pi}\int_{\Sigma_1}\chi\frac{\partial\psi}{\partial n}\,ds - \frac{1}{16}\frac{\Gamma^2}{A^2}\int_0^{2\pi} R^4\,d\vartheta.$$

Here (R, ϑ) are polar coordinates with origin at the centroid of the prime vortex, and χ is the integrand in (2.1) that gives the value of the stream function at the origin, after combining the integrals; that is,

$$\text{(B2)}\qquad \psi(0,0) = \frac{\Gamma}{2\pi A}\iint_{\Sigma_1}\chi\,dA.$$

The functional form of χ depends on the symmetry presumed to exist between the two rows of the street. The actual function that was used in these calculations is

$$\text{(B3)}\qquad \chi(z) = \log\left|\frac{\sin\frac{\pi}{l}z}{\sin\frac{\pi}{l}(z-d+ih)}\right|.$$

Acknowledgments. We acknowledge with gratitude the granting of time by Control Data Corporation on the Cyber 203 computer at the CDC Service Center, Arden Hills, Minnesota.

REFERENCES

[1] B. CHEN AND P. G. 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.

[2] M. E. DAVIES, *A comparison of the wake structure of a stationary and oscillating bluff body, using a conditional averaging technique*, J. Fluid Mech., 75 (1976), pp. 209–231.

[3] S. GOLDSTEIN, ed., *Modern Development in Fluid Dynamics*, Vol. II, Clarendon Press, Oxford, 1938, pp. 556–565.

[4] T. v. KÁRMÁN, *Über den Mechanismus des Widerstands, den ein bewegter Körper in einer Flüssigkeit erfährt*, Göttinger Nachr. Math. Phys. Kl.,(1911), pp. 509–517.

[5] ———, *Über den Mechanismus des Widerstands, den ein bewegter Körper in einer Flüssigkeit erfährt*, Göttinger Nachr. Math. Phys. Kl., (1912), pp. 547–556.

[6] T. v. KÁRMÁN AND H. L. RUBACH, *Über den Mechanismus des Flüssigkeits- und Luftwiderstands*, Phys. Z., 13 (1912), pp. 49–59.

[7] R. T. PIERREHUMBERT, *A family of steady, translating vortex pairs with distributed vorticity*, J. Fluid Mech., 99 (1980), pp. 129–144.

[8] R. T. PIERREHUMBERT AND S. E. WIDNALL, *The structure of organized vortices in a shear layer*, AIAA Paper 79-1560, 1979.

[9] L. ROSENHEAD, *Double row of vortices with arbitrary stagger*, Proc. Cambridge Philos. Soc., 25 (1929), pp. 132–138.

[10] P. G. SAFFMAN AND R. SZETO, *Structure of a linear array of uniform vortices*, to appear, 1981.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0004 $01.00/0

SPECTRAL CALCULATIONS OF ONE-DIMENSIONAL INVISCID COMPRESSIBLE FLOWS*

DAVID GOTTLIEB,† LIVIU LUSTMAN‡ AND STEVEN A. ORSZAG§

Abstract. The extension of spectral methods to inviscid compressible flows is considered. Techniques for high resolution treatment of shocks and contact discontinuities are introduced. Model problems that demonstrate resolution of shocks and contact discontinuities over one effective grid interval are given.

Key words. spectral methods, shock waves, compressible flows, conservation laws, filtering

1. Introduction. Spectral methods [4] are based on representing the solution to a problem as a truncated series of smooth functions of the independent variables. They have been applied to the numerical simulation of a variety of viscous flows, including the numerical simulation of turbulence [15] and the numerical simulation of transition to turbulence [14] in incompressible fluids. It may seem that spectral methods are limited to problems where Fourier series are appropriate in rectangular geometries and spherical harmonic series are appropriate in spherical geometries. In these cases, smooth solutions can be represented as rapidly converging spectral series, leading to significant economies over discrete approximations that lead to finite difference methods.

Spectral methods have now developed as a useful tool in areas far removed from their original, and perhaps obvious, applications. Transform methods [10] have allowed their application to problems with general nonlinear and nonconstant coefficients. Orthogonal polynomial expansions [4] have expanded widely the kinds of boundary conditions amenable to spectral treatment. New extensions of the fast Fourier transform to nearly arbitrary Sturm–Liouville eigenfunction bases [11] may improve the efficiency of spectral methods based on exotic function bases. A fast, general iteration method has improved the efficiency of solving general spectral equations so that spectral solution of problems in general complicated geometries requires little more work than that required to solve the lowest-order finite-difference approximation to the problem in the complex geometry [12].

One kind of problem has not yet received much attention for treatment by spectral methods, namely, the approximation of discontinuous solutions by spectral methods. Some early partial results were encouraging. It was shown [13] that *continuous* solutions with *discontinuous* derivatives are well represented spectrally and that the accuracy advantages of spectral methods over finite difference methods survive for such solutions.

In the present paper, we provide an initial glimpse into the extension of spectral methods to treat discontinuous solutions with shocks and contact discontinuities. The

* Received by the editors November 26, 1979 and in revised form February 17, 1981.

† Department of Applied Mathematics, Tel-Aviv University, Tel-Aviv, Israel. The work of this author was supported by the U.S. Air Force Office of Scientific Research under grant 77-3405, the Office of Naval Research under contract N00014-77-C-0138 and NASA contract NAS1-15810 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

‡ Department of Mathematics, Old Dominion University, Norfolk, Virginia 23508. The work of this author was supported by the National Aeronautics and Space Administration under Grant NAG1-25.

§ Department of Mathematics, M.I.T., Cambridge, MA 02139. The work of this author was supported by the Air Force Office of Scientific Research under grant 77-3405, the Los Alamos Scientific Laboratory which is supported by the U.S. Department of Energy, and the Office of Naval Research under contract N00014-77-C-0138.

results are encouraging. It seems that spectral methods allow the resolution of shock fronts and contact discontinuities by shock-capturing techniques with only one grid interval across the discontinuity. Of course, spectral methods may also be used with shock-fitting techniques to represent a perfectly sharp discontinuity.

2. Linear hyperbolic problems. It has been shown [9] that by pre- and post-processing discontinuous data it is possible to achieve high accuracy with spectral approximations to discontinuous solutions of linear problems.

A standard model problem [1] is given by the one-dimensional wave equation

$$\frac{\partial u(x,t)}{\partial t}+\frac{\partial u(x,t)}{\partial x}=0, \tag{1}$$

with periodic boundary conditions on the interval $0 \leqq x \leqq 2\pi$. With nonsmooth initial data $u(x,0)=f(x)$, the solution $u(x,t)=f(x-t)$ (extended periodically) is nonsmooth for all t and a Fourier-spectral method should be expected to converge slowly. Nevertheless, results obtained by pseudospectral Fourier solution of (1) on a uniform grid are spectacular (at least in comparison with those obtained by such techniques as the flux-corrected transport (FCT) algorithm [1]). Of course, both the spectral and finite-difference results may be improved using adaptive mesh-refinement methods to improve resolution near discontinuities. In Fig. 1, the solution to (1) is plotted at $t=4\pi$ (after two full propagation periods over the spatial domain) with $f(x)=\exp(-(x-\pi)^2/4\Delta x^2)$, so the solution is a Gaussian with width $2\Delta x$, where $\Delta x=2\pi/64$ is the effective grid resolution with 64 Fourier modes. The results plotted in Fig. 1 were obtained using a weak low-pass filter to pre- and post-process the results [see (24) below].

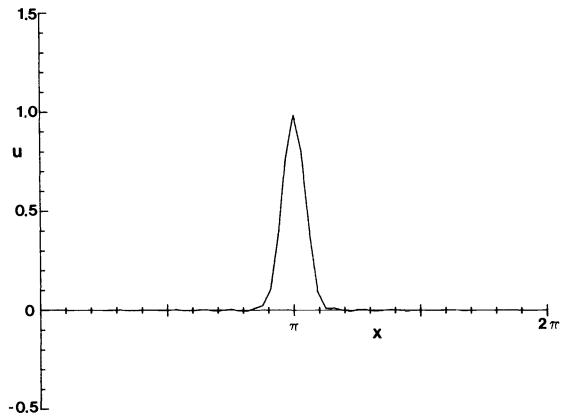


FIG. 1. *A plot of the solution to the wave equation* (1), *with* $f(x)=\exp(-(x-\pi)^2/4\Delta x^2)$ *at* $t=4\pi$, *when the initial pulse has been transported two full periods. A Fourier spectral method with* $N=64$ *modes and a weak low-pass filter was used. The plotted results are in excellent agreement with the exact solution* $f(x-t)$, *with a maximum pointwise error of about* 2%.

In Fig. 2, similar plots are given for the solution to (1) when $f(x)$ is a top-hat function. Here a stronger post-processing [see (22) below] was necessary to remove large oscillations, due to the Gibbs phenomenon, near the discontinuities. The particular post-processing does not appear to be too important for this problem; we used a one-sided average in the neighborhood of rapid changes of the solution. The

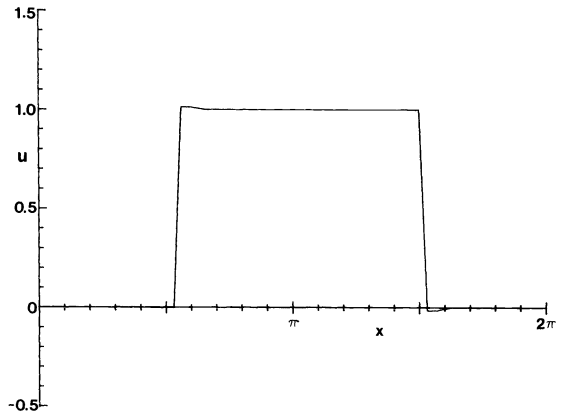


FIG. 2. *Same as Fig.* 1, *except* $f(x) = 1(|x-\pi| < \frac{1}{2}\pi)$, $O(|x-\pi| > \frac{1}{2}\pi)$. *Here a one-sided average was used as a post-filter to remove the oscillations. Again the plotted results at* $t = 4\pi$ *are in excellent agreement with the exact solution* $f(x-t)$.

motivation for this choice is similar to that of the FCT algorithm [1]. The results plotted in Figs. 1 and 2 are obtained using 64 Fourier modes, corresponding to 64 collocation points.

3. Compressible flow problems. In this section, we study the application of spectral methods to one-dimensional compressible flow problems. In § 7, we compare the results with those obtained by more conventional finite-difference methods. The one-dimensional Eulerian equations of motion in a finite shock tube are, in conservation form,

$$\frac{\partial \mathbf{w}}{\partial t} + \frac{\partial}{\partial x}\mathbf{F}(\mathbf{w}) = 0, \qquad -1 \leqq x \leqq 1, \tag{2}$$

$$\mathbf{w} = (\rho, m, E)^T, \qquad \mathbf{F}(\mathbf{w}) = u\mathbf{w} + (0, p, pu)^T, \tag{3}$$

where ρ is the mass density, m is the momentum density, E is the total energy density, p is the pressure and $u = m/\rho$ is the velocity.

We assume the equation of state to be

$$p = (\gamma - 1)[E - \tfrac{1}{2}\rho u^2]; \tag{4}$$

we usually take $\gamma = 1.4$, as is appropriate for a diatomic gas.

For simple shock tube problems, that involve only constant states separated by shocks and contact discontinuities, there is no immediate need for high-order accurate numerical methods. However, for more complicated flow problems, especially in higher dimensions, where there may be interacting shocks, rarefaction waves or contact discontinuities, as well as the interaction of shocks with boundary layers and interfaces, it is necessary to have both high accuracy in the interior of the flow, in the boundary layers and near the interfaces, and good representations of discontinuities. From this point of view it seems appropriate to use the Chebyshev spectral method, since it provides both high interior accuracy and very high resolution in the boundary layer region [4].

We regard the classical simple shock tube problems as extremely severe tests of spectral methods. After all, spectral methods are designed to give good resolution of complicated flow structures distributed through the flow domain and, since they are

based on expansions in orthogonal functions, it would seem that isolated local jumps at shocks and contact discontinuities would be most inhospitable for them. On the other hand, it would seem that low-order finite-difference methods would be best for treating shock discontinuities because of their localized character. One of the important conclusions of this paper is that the accuracy (or rather, the resolution) advantages of spectral methods hold up in the neighborhood of shock discontinuities, not just in regions of smooth flows and boundary layers.

There are three ways to apply Chebyshev-spectral methods to these problems: namely, collocation, Galerkin, and tau approximations [4]. We choose to use the collocation (or pseudospectral) technique here for two reasons. First, collocation is the easiest and most efficient method to apply for complicated problems. Also, since collocation involves solving the equations in physical space rather than in transform space with transforms used only to evaluate derivatives, boundary conditions are also easier to apply.

Let us give a brief description of the collocation method. At each time step, we evaluate the components of $\mathbf{F}$ in (2) at the points

$$x_j = \cos\frac{\pi j}{N}, \qquad 0 \leqq j \leqq N. \tag{5}$$

Next, the Chebyshev expansion coefficients $\mathbf{a}_n$ of $\mathbf{F}$ are found from

$$\mathbf{F}(x_j) = \sum_{n=0}^{N} \mathbf{a}_n T_n(x_j), \qquad 0 \leqq j \leqq N, \tag{6}$$

where $T_n(x)$ is the Chebyshev polynomial of degree n defined by $T_n(x) = \cos(n \cos^{-1} x)$. Since $T_n(x_j) = \cos(\pi jn/N)$ it follows that

$$\mathbf{a}_n = \frac{2}{N\bar{c}_n} \sum_{p=0}^{N} \frac{1}{\bar{c}_p} \mathbf{F}(x_p) \cos\frac{\pi pn}{N}, \qquad 0 \leqq n \leqq N, \tag{7}$$

where $\bar{c}_0 = \bar{c}_N = 2$ and $\bar{c}_n = 1$ for $0 < n < N$. Differentiating (6) gives

$$\left(\frac{\partial \mathbf{F}}{\partial x}\right)_j = \sum_{n=0}^{N} \mathbf{a}_n T_n' = \sum_{n=0}^{N} \mathbf{s}_n T_n, \tag{8}$$

where

$$\mathbf{s}_n = \frac{2}{\bar{c}_n} \sum_{\substack{p=n+1 \\ p+n \text{ odd}}}^{N} p\mathbf{a}_p. \tag{9}$$

Equations (7)–(8) are implemented using the fast Fourier transform. In order to avoid having to perform order N^2 operations to evaluate (9), we observe that $\mathbf{s}_N = 0$, $\mathbf{s}_{N-1} = 2N\mathbf{a}_N$ and $\mathbf{s}_n (n \leqq N-2)$ satisfies the recurrence relation

$$\bar{c}_n \mathbf{s}_n = \mathbf{s}_{n+2} + 2(n+1)\mathbf{a}_{n+1} \qquad (n \leqq N-2). \tag{10}$$

Using (10), only $O(N)$ operations are necessary to obtain $\mathbf{s}_n$ from $\mathbf{a}_n$. Thus, using the fast Fourier transform, evaluation of $\partial \mathbf{F}/\partial x$ from $\mathbf{F}$ requires only order $N \log N$ operations.

It should be noted that the collocation points x_j are crowded in the neighborhood of $x = 1$ and $x = -1$. [x_1 and x_{N-1} are located at distances of $\pi^2/2N^2$ from $x = 1, -1$, respectively.] For $N = 128$, there are 40 points located in the interval $0.9 < |x| \leqq 1$. This high boundary resolution can be of great value when boundary layers or interfaces are also present, but is wasteful for the simple test problems of the present paper.

Next, we describe some time marching techniques. Let us denote the discrete approximation for $\mathbf{w}$ at collocation point x_j and time step $n\Delta t$ by $\mathbf{w}_j^n$, the discrete approximation for $\mathbf{F}$ by $\mathbf{F}_j^n$. Then we advance in time using the two-step (modified Euler) method:

$$\mathbf{w}_j^{n+1/2} = \mathbf{w}_j^n - \frac{1}{2}\Delta t\left(\frac{\partial \mathbf{F}}{\partial x}\right)_j^n, \tag{11}$$

$$\mathbf{w}_j^{n+1} = \mathbf{w}_j^n - \Delta t\left(\frac{\partial \mathbf{F}}{\partial x}\right)_j^{n+1/2}. \tag{12}$$

This scheme results in a linear stability condition of the form

$$\Delta t \leqq \frac{8}{N^2 \max(|u|+c)}, \tag{13}$$

where c is the sound speed. This condition is very severe; indeed, using finite difference methods with N points leads to stability restrictions like $\Delta t = O(1/N)$. An alternative approach is given in [5] and [12] that avoids the latter difficulty. The approach described in [5] yields an unconditionally explicit stable scheme whereas that described in [12] involves a very efficient implicit technique.

In this paper, however, we present results gotten by using (11), (12). Application of the other time marching techniques will be given elsewhere.

4. Conservation properties of pseudospectral methods. In this section, we discuss the conservation properties of pseudospectral methods applied to a nonlinear system of equations. Consider

$$\frac{\partial \mathbf{w}}{\partial t} = \frac{\partial \mathbf{F}(\mathbf{w})}{\partial x}, \qquad \mathbf{w}(x, 0) = \boldsymbol{\phi}(x). \tag{14}$$

It is well known that, in general, no smooth solution can exist for all time. Instead, one seeks a weak solution defined by the requirement that the integral relation

$$\int\int (\boldsymbol{\psi}_t \mathbf{w} - \boldsymbol{\psi}_x \mathbf{F})\, dx\, dt + \int \boldsymbol{\psi}(x, 0)\boldsymbol{\phi}(x)\, dx = 0 \tag{15}$$

be specified for all smooth test vectors $\boldsymbol{\psi}$ which vanish both for large t and on the boundary of the domain.

It can be shown that, as a result of the integral relation (15), the weak solution must satisfy the Rankine–Hugoniot shock conditions. Assume now that (14) is spatially discretized by the pseudospectral Fourier method (see [4]). By this we mean that $\mathbf{w}_N(x, t)$ is defined as the trigonometric interpolant of $\mathbf{w}(x, t)$ at the points $x = \pi j/(N+1)(j = 0, \cdots, 2N)$, $\mathbf{F}_N(x, t)$, is defined as the interpolant of $\mathbf{F}$ at x_j and one solves the $2N+1$ vector equations

$$\frac{\partial \mathbf{w}_N}{\partial t}(x, t) = \frac{\partial \mathbf{F}_N}{\partial x}(x_j, t), \qquad j = 0, \cdots, 2N. \tag{16}$$

This discretization should be used only if the boundary conditions of (14) are periodic with period 2π.

Following Lax and Wendroff [8], we can prove

THEOREM 1. *Assume that as* $N \to \infty$, $\mathbf{w}_N$ *converges boundedly to some vector* $\mathbf{w}$. *Then* $\mathbf{w}(x, t)$ *is a weak solution of* (14) *with initial value* $\boldsymbol{\phi}$.

Proof. Let $\psi(x, t)$ be a periodic test function and let $\psi_N(x, t)$ be the interpolant of ψ at $x = x_j$. Let $\phi_N(x)$ be the interpolant of the initial condition ϕ at the same points.

Multiplying (16) by $\psi_N(x_j)$ and summing, one gets

$$\sum_{j=0}^{2N} \psi_N(x_j, t)\frac{\partial \mathbf{w}_N}{\partial t}(x_j, t) = \sum_{j=0}^{2N} \psi_N(x_j, t)\frac{\partial \mathbf{F}_N(\mathbf{w}_N(x_{j,t}))}{\partial x}.$$

We interpret now the sums on both sides of this result in terms of the trapezoidal integration rule. Since $\psi_N w_N$ and $\psi_N F_N$ are trigonometric polynomials of degree $2N$, the trapezoidal rule is exact (see [2]). Therefore, we obtain

$$\int_0^{2\pi} \psi_N(x, t)\frac{\partial \mathbf{w}_N}{\partial t}(x, t)\, dx = \int_0^{2\pi} \psi_N(x, t)\frac{\partial \mathbf{F}_N}{\partial x}(x, t)\, dx = -\int_0^{2\pi} \mathbf{F}_N(x, t)\frac{\partial \psi_N}{\partial x}\, dx.$$

We integrate now with respect to t to get

$$\int_0^{2\pi} dx \int_0^{\infty} dt\, \psi_N(x, t)\frac{\partial \mathbf{w}_N}{\partial t}(x, t) + \int_0^{\infty} dt \int_0^{2\pi} dx\, \mathbf{F}_N(x, t)\frac{\partial \psi_N}{\partial x} = 0.$$

Integrating the left-hand side by parts and taking into account that $\psi_N(x, t) \to 0$, $t \to \infty$, we obtain

$$-\int_0^{2\pi} \psi_N(x, 0)\mathbf{w}_N(x, 0)\, dx - \int_0^{2\pi} \int_0^{\infty} \left[\mathbf{w}_N(x, t)\frac{\partial \psi_N}{\partial t}(x, t) - \mathbf{F}_N(x, t)\frac{\partial \psi_N}{\partial x}(x, t)\right] dx\, dt = 0$$

or

$$\int_0^{2\pi} \psi_N(x, 0)\phi_N(x)\, dx + \int_0^{2\pi} \int_0^{\infty} \left[\mathbf{w}_N(x, t)\frac{\partial \psi_N}{\partial t}(x, t) - \mathbf{F}_N(x, t)\frac{\partial \psi_N}{\partial x}(x, t)\right] dx\, dt = 0.$$

Letting $N \to \infty$ completes the proof of the theorem.

Theorem 1 establishes the fact that the pseudospectral Fourier method applied to problems with shocks yields the correct shock speed. The following argument [7] asserts that this method can achieve resolution of shock discontinuities over one effective grid interval. If the numerical approximation u_{ap} to an exact shock solution u_{ex} over the interval $0 \leqq x < 2\pi$ is smeared over a distance h near a shock of strength U then

$$\int_0^{2\pi} (u_{ex} - u_{ap})^2\, dx \gtrsim \frac{hU^2}{3}$$

where the constant $\frac{1}{3}$ obtains if the error $u_{ex} - u_{ap}$ varies linearly from 0 to U over a distance $h/2$. On the other hand, it is in principle possible that the first N Fourier coefficients of u_{ap} in an N-term Fourier spectral calculation agree closely with those of u_{ex}. Since the nth Fourier coefficient a_n of a shock solution on $[0, 2\pi]$ behaves asymptotically as $U/2\pi n$ as $n \to \infty$, it follows that

$$\int_0^{2\pi} (u_{ex} - u_{ap})^2\, dx \leqq 2\pi \sum_{|n| \geqq N/2} |a_n|^2 \sim \frac{2U^2}{\pi N}.$$

Therefore, $h \lesssim 6/\pi N$. Since the effective grid separation Δ is $2\pi/N$, it follows that, optimally,

$$h \lesssim \frac{\Delta}{3},$$

or better than one grid interval resolution.

It is interesting to note that (16) implies that the components of $\mathbf{w}_N$ are conserved. In fact, summing up both sides of (16), one gets

$$\sum_{j=0}^{2N} \frac{\partial \mathbf{w}_N}{\partial t}(x_j, t) = \sum_{j=0}^{2N} \frac{\partial \mathbf{F}_N}{\partial x}(x_j, t).$$

By the same argument as before, one can replace the sums by integrals to get

$$\frac{d}{dt}\int_0^{2\pi} \mathbf{w}_N(x, t)\, dx = \int_0^{2\pi} \frac{\partial \mathbf{F}_N}{\partial x}(x, t)\, dx = 0.$$

Therefore

$$\int_0^{2\pi} \mathbf{w}_N(x, t)\, dx = \int_0^{2\pi} \mathbf{w}_N(x, 0)\, dx.$$

For compressible flow problems, this shows that the mass, momentum, and energy are conserved.

We would like now to demonstrate that the same conservation properties that were established for the pseudospectral Fourier method hold also for the pseudospectral Chebyshev method. In analogy to Theorem 1, we can prove

THEOREM 2. *Let* $\mathbf{w}_N$ *and* $\mathbf{F}_N$ *be the pseudospectral Chebyshev approximation described in* (5)–(10) *to* (14). *Then if* $\mathbf{w}_N$ *converges to* $\mathbf{w}$ *and* ψ *is a smooth test function such that* $\psi(-1, t) = \psi(1, t) = 0$, $\psi(x, \infty) = 0$ *then* $\mathbf{w}(x, t)$ *is a weak solution of* (14).

Proof. It is shown in [4, p. 15] that $\mathbf{w}_N$ satisfies exactly the equation

$$\frac{\partial \mathbf{w}_N}{\partial t} = \frac{\partial \mathbf{F}_N}{\partial x} + \tau(t) U_{N-1}(x) P(x), \tag{17}$$

where U_{N-1} is the Chebyshev polynomial of the second kind so that $U_{N-1}(x_j) = 0$ for $j = 1, \cdots, N-1$ and $P(x)$ is a polynomial of degree 1 that corresponds to the boundary condition. More precisely, $P(x) = 1 - x$ when the boundary data is prescribed at $x = -1$, and $P(x) = 1 + x$ when the boundary data is prescribed at $x = +1$. Suppose ψ_{N-3} is the interpolant of $\psi/\sqrt{1-x^2}$ at the points $x_j = \cos(\pi j/(N-3))(j = 0, \cdots, N-3)$, then

$$\begin{aligned}\int_{-1}^{1} \sqrt{1-x^2}\psi_{N-3}\frac{\partial \mathbf{w}_N}{\partial t}(x, t) &= \int_{-1}^{1} \sqrt{1-x^2}\psi_{N-3}\frac{\partial \mathbf{F}_N}{\partial x} \\ &\quad + \tau \int_{-1}^{1} U_{N-1}P(x)\psi_{N-3}\sqrt{1-x^2}\, dx.\end{aligned} \tag{18}$$

Since $P(x)\psi_{N-3}(x)$ is a polynomial of degree $N-2$, the last integral at the right-hand side of (18) vanishes.

The rest of the proof is similar to the proof of Theorem 1. Defining $\rho_N = \sqrt{1-x^2}\psi_{N-3}$ and integrating, we obtain

$$\int_{-1}^{1} \mathbf{w}_N(x, 0)\rho_N(x, 0)\, dx + \int_0^{\infty}\int_{-1}^{1} \left(\mathbf{w}_N\frac{\partial \rho_N}{\partial t} - \mathbf{F}_N\frac{\partial \rho_N}{\partial x}\right) dx\, dt = 0.$$

Since $\rho_N \to \psi$, uniformly, and the derivatives of ρ_N converge similarly to the derivatives of ψ and since $\mathbf{w}_N \to \mathbf{w}$, the proof is completed.

We are also able to show that the components of $\mathbf{w}$ are conserved. In fact, since

$$\frac{\partial \mathbf{w}_N}{\partial t}(x_j) = \frac{\partial \mathbf{F}_N}{\partial x}(x_j) \tag{19}$$

for all interior points x_j, one can use the Clenshaw–Curtiss quadrature formula [2] to get

$$\sum_{j=0}^{N} \frac{\partial w_N}{\partial t}(x_j)\alpha_j = \int_{-1}^{1} \frac{\partial w_N}{\partial t}\, dx,$$

$$\sum_{j=0}^{N} \frac{\partial F_N}{\partial x}(x_j)\alpha_j = \int_{-1}^{1} \frac{\partial F_N}{\partial x}\, dx - F_N(1) - F_N(-1),$$

where

$$\alpha_j = \alpha_{N-j} = \frac{4}{N} \sum_{k=0}^{N/2} \frac{1}{1-4k^2} \cos\frac{\pi jk}{N},$$

$$\alpha_0 = \alpha_N = \frac{1}{N^2-1}.$$

Therefore, (19) gives

$$\frac{d}{dt}\int_{-1}^{1} \mathbf{w}_N\, dx = \boldsymbol{\sigma},$$

where $\boldsymbol{\sigma}$ represents the contribution from the boundaries. In the case of homogeneous boundary conditions, where $\mathbf{w}_N$ and $\partial \mathbf{F}_N/\partial x$ both vanish at the boundaries so that $\boldsymbol{\sigma} = 0$,

$$\int_{-1}^{1} \mathbf{w}_N(x, t)\, dx = \int_{-1}^{1} \mathbf{w}_N(x, 0)\, dx, \tag{20}$$

demonstrating conservation.

5. Boundary conditions. Boundary conditions play a crucial role in the application of spectral methods. Incorrect boundary treatment may give strong instabilities, in contrast to finite difference methods in which instabilities due to boundaries usually appear as relatively weak oscillations. On the other hand, as opposed to high-order finite-difference methods, spectral methods normally do not require numerical boundary conditions in addition to the physical boundary conditions required by the partial differential equation.

For shock tube problems, we must specify all the flow variables at supersonic inflow points, two flow variables at subsonic inflow points and one flow variable at subsonic outflow points. If we overspecify or underspecify the boundary conditions, spectral calculations are usually spectacularly unstable.

At subsonic outflow points, it is not satisfactory to specify arbitrarily any one of the flow variables m, ρ, u or E. With arbitrary outflow boundary conditions, one can obtain oscillations that originate at the boundary. The outflow boundary conditions used here were obtained following the analysis given in [3]. We advance one time step without imposing the boundary conditions and denote the calculated quantities by $\mathbf{w}_c$. We observe that if $x = +1$ is a subsonic outflow point, the incoming characteristic quantity at $x = +1$ is $v_1 = p - (\rho c)u$, whereas the outgoing characteristic quantities are $v_2 = p + (\rho c)u$, $v_3 = p - c^2\rho$. Let us assume that one flow variable is given on the inflow characteristic at $x = +1$. Then we solve the system

$$v_2 = (v_2)_c, \qquad v_3 = (v_3)_c \tag{21}$$

for the two remaining flow variables at $x = 1$. This procedure yields a stable scheme with no oscillations emanating from the boundaries. In contrast to inflow–outflow

boundaries whose treatment is quite systematic by the above procedure, material boundaries evidently do require boundary conditions in addition to those required by the mathematical theory of characteristic initial value problems. At characteristic surfaces, like material boundaries, the specification of one flow variable, like $u=0$, should suffice. However, we find that it is necessary to supplement this boundary condition at only one characteristic boundary by one additional condition, like p given. An analysis of these boundary conditions will be given elsewhere.

6. Smoothing and filtering. The approximation of discontinuous functions by truncated Chebyshev polynomial expansions exhibits large oscillations near jumps and has two point oscillations over the whole region [4]. Similar oscillations are observed when approximating shock waves in inviscid flows. These oscillations may induce instabilities in nonlinear problems since they can interact with the smooth part of the solution and be amplified. It is essential for stability reasons either to eliminate completely these oscillations or to control them in such a way that stability is not affected.

Several methods to achieve stable computations have been investigated including artificial viscosity, Shuman filtering and a new spectral filtering method to be described below. We will report results obtained by the latter two methods.

Shuman filtering involves applying the filter

$$\bar{w}_j^n = w_j^n + \theta_{j+1/2}(w_{j+1}^n - w_j^n) + \theta_{j-1/2}(w_j^n - w_{j-1}^n). \tag{22}$$

The idea is to choose smoothing factors θ_j that vanish in smooth parts of the solution and become large only in the neighborhood of discontinuities. Following Harten and Tal-Ezer [6], we choose

$$\theta_{j+1/2} = \beta \frac{|\rho_{j+2} - \rho_{j+1} - 2(\rho_{j+1} - \rho_j) + (\rho_j - \rho_{j-1})|}{|\rho_{j+2} - \rho_{j+1}| + 2|\rho_{j+1} - \rho_j| + |\rho_j - \rho_{j-1}|}, \tag{23}$$

where $0 < \beta < 1$ is a suitable constant. Typically, $\beta = 0.01$ in our calculations.

A more intriguing kind of filtering is based on the following idea. If a low-pass spectral filter just strong enough to remove those high frequency waves that lead to numerical instabilities is applied to the inviscid compressible flow equations, the spectral equations will give bad oscillations near shock fronts and other discontinuities. However, as recently pointed out by Lax [7], these oscillatory solutions obtained by a high-order method like a spectral method should contain enough information to be able to reconstruct the proper nonoscillatory discontinuous solution by a post-processing filter. The idea is that very weak filtering or damping to stabilize together with a final "cosmetic" filter to present the results should be able to give great improvements in resolution.

In practice, we use a low-pass filter for stabilizing purposes that is of the form

$$f(k) = \begin{cases} 1, & k < k_0, \\ e^{-\alpha(k-k_0)^4}, & k > k_0, \end{cases} \tag{24}$$

where k is a spectral (wavenumber) index, k_0 depends on the strength of the shock and α is a constant of order 1. For typical fluid dynamical shocks, we choose $k_0 \sim \frac{5}{6}N$, where N is the maximum wavenumber in the spectral representation of the flow. We have found that, for moderate shocks, the results are insensitive to the detailed form of (24). However, it is important that $f(k)$ be low-pass so $f(k) = 1$ for $k < k_0$, because the large-scales are treated very accurately (without phase error) by the spectral method.

Finally, we explain how to post-process the numerical results to retrieve smoothed results with localized discontinuities from the noisy, but stable, calculations. Our method is to first determine the location of the discontinuities in the flow and then to apply a Shuman filter to smooth the data on either side of the discontinuities (without using data from the opposite side of the discontinuity in the smoothing process). The key to the success of this procedure is the apparent ability of the spectral results to preserve information on the precise location of discontinuities. Similar post-processing methods do not work so well on finite-difference results because, in such calculations, the shock position does not seem to be localized within one grid interval.

The results obtained by this procedure of pre- and post-processing the results are not very sensitive to details of the procedure (e.g., α and k_0 in (24)), provided the filtration *is* strong enough to remove instabilities. At the present time, precise stability bounds for α and k_0 are not known.

The location of discontinuities is found by an examination of the spectral coefficients. One may assume that the smooth part of the solution is well represented by the first few coefficients, while the leading behavior of the high-order coefficients is generated by the discontinuities. As a rule of thumb, we examine the middle third of the coefficients, since the highest third may be unreliable due to the stabilizing smoothing.

The expansion coefficients a_k should be fit by an expression of the form

$$a_k = \sum_{s=1}^{S} B_s A_k(X_s), \qquad \frac{N}{3} \lesssim k \lesssim \frac{2N}{3}, \tag{25}$$

where S is the number of shocks, $A_k(X)$ is the spectral coefficient of a Heaviside function with a jump at X and B_s, X_s are the intensities and locations, respectively, of the various shocks.

In our case (Chebyshev collocation), it may be readily verified that

$$A_k(X) = \frac{1}{N} \frac{\sin \dfrac{k\pi}{N}(L+1/2)}{\sin \dfrac{k\pi}{2N}}, \qquad 0 < k < N,$$

$$A_0(X) = \frac{L+1/2}{N}, \tag{26}$$

$$A_N(X) = \frac{1}{2N} \sin \pi(L+\tfrac{1}{2}),$$

with L an integer related to X by

$$\cos \frac{(L+1)\pi}{N} < X < \cos \frac{L\pi}{N}. \tag{27}$$

The solution of (25) for B_s, X_s proceeds as follows. First, we rewrite (25) in the form

$$Na_k \sin \frac{k\pi}{2N} \overset{\text{def}}{\equiv} \tilde{a}_k = \sum_{s=1}^{S} B_s \sin(k\omega_s),$$

with $\omega_s = (L(X_s)+\frac{1}{2})(\pi/N)$. Now note that a sequence $b_k = \sin k\omega$ satisfies the identity

$$\frac{b_{k-1}+b_{k+1}}{2} \overset{\text{def}}{\equiv} (\Omega b)_k = \cos \omega b_k.$$

Therefore, the values ω_s are simply related to the eigenvalues of the summation operator Ω. They may be computed independently of B_s; the jump magnitudes are found after the ω_s are determined, by a least squares procedure.

We applied this algorithm to the test case of several Heaviside functions added to a smooth background, such as e^x. Shock locations are recovered exactly. They are assigned to the nearest "midpoint" by (27). With $N=64$, the predicted shock intensities are correct within a few percent for as many as 7 shocks. Moreover, when more shocks are sought than are actually present, some ω_s become imaginary, which serves as a useful check.

When actual computational data are input to the "shock locator" program described above, several jumps may be identified (of course, with the number of shocks not specified beforehand). One may then proceed in different ways:

a) Smooth between the shock locations, using one-sided smoothing at the shocks.

b) Subtract the sum of Heaviside functions and smooth the result in physical space (e.g., using a Shuman filter).

c) Subtract the coefficients of the Heaviside functions found, and smooth in the coefficient space (e.g., by deleting high order coefficients).

Of these, method a) seems the most robust, as it needs no values for the actual shock intensities. In computer experiments, all three methods perform comparably.

There are many fine adjustments that have to be made on the general algorithm to obtain the best results. On wildly oscillatory data, one may obtain spurious shocks due to large differences from "point to point"; sometimes, since several such oscillations are interpreted as shocks, the algorithm errs simply because S is too large. We are investigating some ways of avoiding spurious shocks: namely, imposing an entropy condition, ignoring shocks of low intensity, weighting the coefficients or data, and iterating the smoothing procedure. There is not yet an approach which will solve all hard problems, although reasonably good results are not hard to obtain.

7. Numerical results. The first model problem is a shock tube problem with a diatomic gas ($\gamma = 1.4$), satisfying supersonic inflow at $x=-1$ and subsonic outflow at $x=+1$. The initial conditions are

$$p=\rho_1, \quad \rho=\rho_1, \quad u=u_1, \qquad -1<x\leqq 1, \tag{28}$$

while the conditions applied at the inflow point $x=-1$ are

$$p=p_2, \quad \rho=r(\xi,\rho_1), \quad u=\bar{u}(\xi,\rho_1,p_1,u_1), \tag{29}$$

where

$$r(\xi,\rho_1)=\frac{1+6\xi}{\xi+6}\rho_1,\ \bar{u}(\xi,\rho_1,p_1,u_1)=u_1-5\frac{(\xi-1)}{\sqrt{42\xi+7}}\sqrt{\frac{p_1}{\rho_1}}, \tag{30}$$

and $\xi=p_2/p_1$ is the strength of the resulting shock. With these initial and boundary conditions, a pure shock propagates from $x=-1$ toward $x=+1$ at a speed

$$v_s=u_1+\sqrt{\frac{p_1+6p_2}{7\rho_1}}. \tag{31}$$

In Fig. 3, we plot the density structure of the resulting shock wave at $t=0.1$ determined by the Chebyshev spectral method with $N=64$ polynomials, $\xi=5$ in (28)–(31), and $p_1=\rho_1=u_1=1$. The exact pressure jump across this shock wave is 5 and the density jump is 2.81818. The shock speed is 3.48997. In these calculations the low-pass filter (24) is applied every time step and the final results are cosmetically

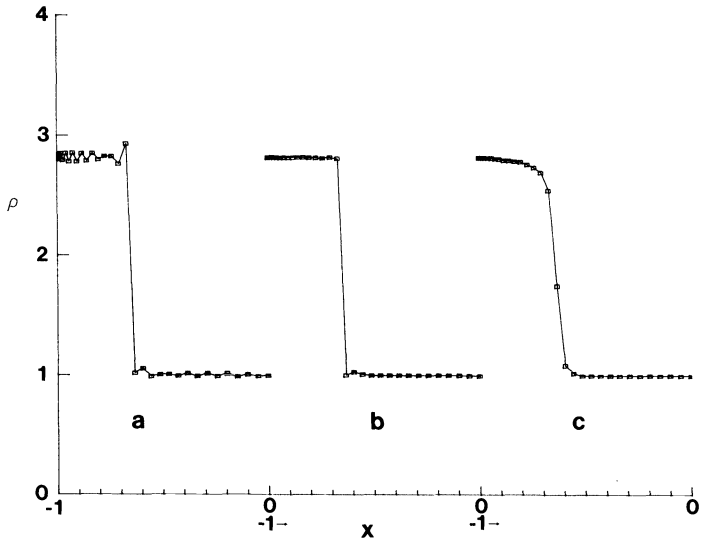


FIG. 3. *A plot of the density ρ vs x for the shock tube problem with initial conditions* (28)–(29) *with shock strength* $\xi = 5$ *at* $t = 0.1$, *obtained using the pseudospectral method with* $N = 64$ *Chebyshev polynomials.* (a) *Results obtained applying the low-pass filter* (24) *at every time step.* (b) *Results obtained by applying the post-processing Shuman filter* (22) *with constant* θ_j *(except one-sided at the shock) to the results plotted in* (a). (c) *Results obtained using a Shuman filter* (22) *with* (23) [*with no low-pass filter*] *at every time step.*

filtered by a one-sided Shuman filter (22) with constant θ_j, replaced by one-sided smoothing at shocks. Also, we plot in Fig. 3 the results obtained using a localized Shuman filter (22) with (23) at every time step. The latter result is similar to results obtained using a von Neumann–Richtmyer artificial viscosity. Observe that the cosmetically filtered results yield a one-point shock while the localized Shuman filter yields a three-point shock. Similar results are obtained applying the localized Shuman filter only every 100 steps or so.

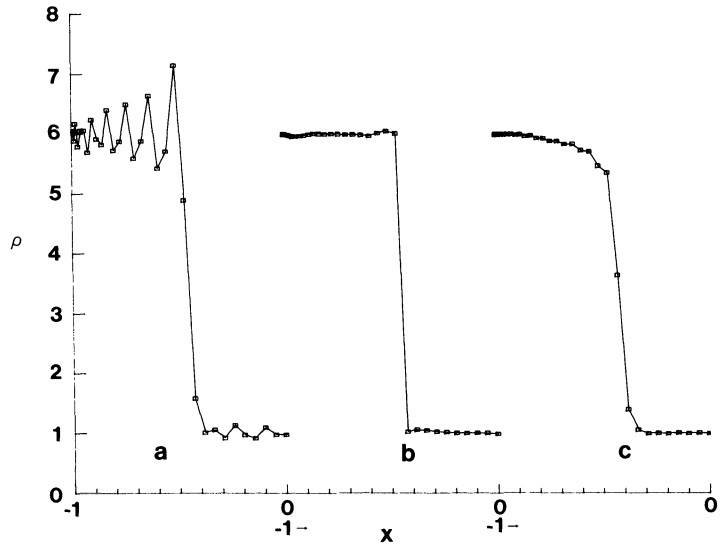


FIG. 4. *Same as Fig.* 3, *except* $\xi = 10^4$, $t = 0.005$.

In Fig. 4, we give a similar plot of the density across a strong shock with $\xi = 10^4$ at $t = 0.005$. Here the density jump is 5.9965. The results plotted in Figs. 3 and 4 demonstrate that our methods achieve high resolution shocks without significant oscillations.

The second model problem is a shock tube problem with $x = \pm 1$ as material boundaries. The initial conditions at $t = 0$ are

$$p = \begin{cases} 1.0, & x < 0, \\ 0.55, & x = 0, \\ 0.1, & x > 0, \end{cases}$$

$$\rho = \begin{cases} 1.0, & x < 0, \\ 0.5625, & x = 0, \\ 0.125, & x > 0, \end{cases} \tag{32}$$

$$u = 0,$$

while the boundary conditions are

$$\begin{aligned} u &= 0, \quad x \pm 1, \\ \rho &= 1, \quad x = -1. \end{aligned} \tag{33}$$

For moderate t, the solution to this problem consists of one shock, one contact discontinuity and a rarefaction wave.

A variety of difference methods for solution of the flow that evolves from (32)–(33) have been compared by Sod [16]. We have solved this problem using the spectral filtering method with $N = 64$ Chebyshev polynomials to represent the flow (in contrast to Sod's 100 point grid). The results are plotted in Fig. 5 at $t = 0.3$. Both the shock and contact discontinuity are resolved over only one grid point. The overall solution is in

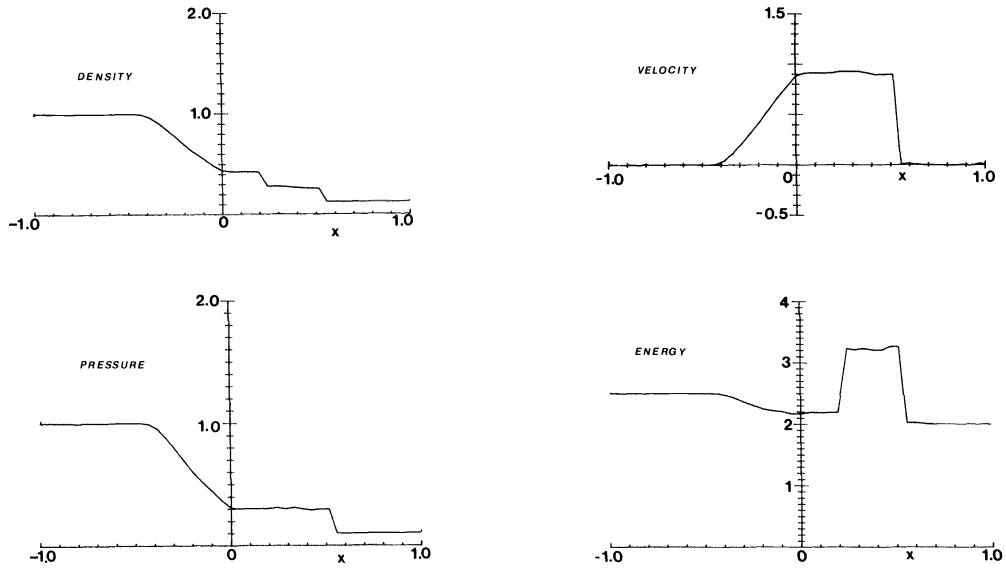


FIG. 5. *Plots of the density, pressure, velocity, and energy at $t = 0.3$ for the shock tube problem with initial-boundary conditions* (32)–(33). *The Chebyshev spectral equations were truncated at $N = 64$ polynomials, and the spectral filtering method was applied to smooth the final results.*

good agreement with the exact solution to the problem. However, without the "cosmetic" final filter, the results are highly oscillatory. Evidently, the highly oscillatory, but stable, spectral solutions do contain enough information to reconstruct sharp discontinuities.

Finally, we consider a problem in which we test for possible degrading of the solution due to the nonlinear interaction between oscillations arising from interacting shock waves. The initial conditions are

$$
\begin{aligned}
&p = \rho = u = 1, \qquad -0.9 < x \leqq 1, \\
&p = 2, \quad \rho = r(2, 1), \quad u = \bar{u}(2, 1, 1, 1), \qquad -1 < x \leqq -0.9, \\
&p = 5, \quad \rho = r(2.5, r(2, 1)), \quad u = \bar{u}(2.5, r(2, 1), 2, \bar{u}(2, 1, 1, 1)), \quad x = -1.
\end{aligned} \tag{34}
$$

These initial conditions correspond to a $\xi = 2.5$ shock lying a distance 0.1 behind a $\xi = 2$ shock. The stronger shock overtakes the weaker shock and generates a single coalesced shock and associated contact discontinuities and rarefaction wave. The resulting density at $t = 0.4$ after shock coalescence is plotted in Fig. 6. Observe that the cosmetic filtering method gives good resolution of both the shock and the contact

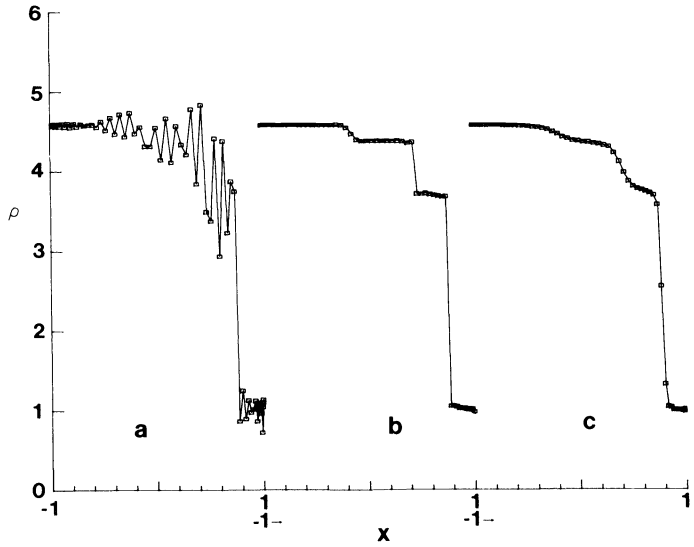


FIG. 6. *A plot of the density ρ vs x for the two-shock problem with initial conditions* (34). *Here, a* $\xi = 2.5$ *shock lies initially a distance* 0.1 *behind a* $\xi = 2$ *shock. The density is plotted at* $t = 0.4$ *after shock coalescence.* (a) *Low-pass filter* (24) *only.* (b) *Post-processing filter* (22) *with constant* θ_i *(except one-sided at calculated discontinuities) applied to the results of* (a). (c) *Shuman filter* (22) *with* (23) *at every time step.*

discontinuity while the localized Shuman filtering method deteriorates in the neighborhood of the contact.

In conclusion, we believe that the excellent results achieved by the present initial investigation of spectral methods for highly compressible flows suggests that these methods may prove useful for problems of practical interest. Such applications are now underway.

REFERENCES

[1] J. P. BORIS AND D. L. BOOK, *Solution of continuity equations by the method of flux-corrected transport*, Methods in Computational Physics, Vol. 16, Academic Press, New York, 1976.

[2] P. J. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*, Academic Press, New York, 1975.

[3] D. GOTTLIEB, M. GUNZBURGER AND E. TURKEL, *On numerical boundary treatment for hyperbolic systems*, SIAM J. Numer. Anal., to appear.

[4] D. GOTTLIEB AND S. A. ORSZAG, *Numerical Analysis of Spectral Methods: Theory and Applications*, NSF-CBMS Regional Conference Series in Applied Mathematics 26, Society for Industrial and Applied Mathematics, Philadelphia, 1977.

[5] D. GOTTLIEB AND E. TURKEL, *On time discretizations for spectral methods*, Stud. in Appl. Math., 63 (1980), pp. 67–86.

[6] A. HARTEN AND H. TAL-EZER, *On a fourth order accurate implicit finite difference scheme for hyperbolic conservation laws*, Math. Comp., 36 (1981), pp. 353–373.

[7] P. D. LAX, *Accuracy and resolution in the computation of solutions of linear and nonlinear equations*, in Recent Advances in Numerical Analysis, Proc. Symposium, Mathematics Research Center, Univ. of Wisconsin, Academic Press, New York, 1978, pp. 107–117.

[8] P. D. LAX AND B. WENDROFF, *Systems of conservation laws*, Comm. Pure Appl., Math. 23 (1960), pp. 217–237.

[9] A. MAJDA, J. MCDONOUGH AND S. OSHER, *The Fourier method for nonsmooth initial data*, Math. Comp., 32 (1978), pp. 1041–1081.

[10] S. A. ORSZAG, *Numerical simulation of incompressible flows within simple boundaries: I. Galerkin (spectral) representations*, Stud. in Appl. Math., 50 (1971), pp. 293–327.

[11] ———, *"Fast" eigenfunction transforms*, Adv. in Math. (1981), to appear.

[12] ———, *Spectral methods for problems in complex geometries*, J. Comp. Phys., 37 (1980), pp. 70–92.

[13] S. A. ORSZAG AND L. W. JAYNE, *Local errors of difference approximations to hyperbolic equations*, J. Comp. Phys., 14 (1974), pp. 93–103.

[14] S. A. ORSZAG AND A. T. PATERA, *Subcritical transition to turbulence in plane channel flows*, Phys. Rev. Letters, 45 (1980), pp. 989–993.

[15] S. A. ORSZAG AND G. S. PATTERSON, *Numerical simulation of three-dimensional homogeneous isotropic turbulence*, Phys. Rev. Letters, 28 (1972), pp. 76–79.

[16] G. A. SOD, *A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws*, J. Comp. Phys., 27 (1978), pp. 1–31.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0005 $01.00/0

EXACT EXTENSION TO THE INFINITE DOMAIN FOR THE VORTEX-IN-CELL METHOD*

BENOIT COUËT† AND A. LEONARD‡

Abstract. The three-dimensional vortex-in-cell method has been extended to allow for the numerical simulation of plane layers of vortical fluid bounded on each side by a semi-infinite domain of unsteady potential flow. Exact boundary conditions for Poisson's equation for the velocity field were derived for the computational domain enclosing all vorticity-containing fluid. A variational principle with quadratic spline basis functions was used to derive the finite pentadiagonal systems that discretize the differential equations.

Key words. vortex-in-cell, variational principle, pentadiagonal, infinite domain, quadratic spline, Poisson's equation, incompressible fluid dynamics.

1. Introduction. In this paper, we present an extension of the three-dimensional vortex-in-cell (VIC) method [2] that allows for the simulation of unbounded incompressible flows with exact boundary conditions in the direction normal to the flow. As before, the vorticity field is represented by a set of vortex filaments which move under the influence of the velocity field that these filaments create. Consequently, sharp gradients of vorticity can be tracked accurately and not diffused by numerical effects. To treat a large number of filaments at a reasonable computer cost, the velocity field is not calculated directly by the Biot–Savart law of interaction [4] but by creating a mesh record of the vorticity field using best fit quadratic spline interpolation, then integrating a Poisson's equation to generate a mesh record of the velocity field. The VIC method has been tested on fully periodic flows and no undesirable grid effects or numerical instabilities were found [2]. Excellent comparisons were achieved with a spectral method calculation of the inviscid Taylor–Green problem [1] and with a pure Lagrangian computation of the propagation of a periodic system of vortex rings [2]. In the present work, Fourier transformation, which imposes periodic boundary conditions, is used only in the streamwise and spanwise directions x and z. This leads to a system of uncoupled second order ODE's in the normal direction y for each Fourier component of the velocity field. As discussed in the next section, discretization of each ODE produces an infinite pentadiagonal system which is then transformed into an equivalent finite pentadiagonal system. The solution is demonstrated to be invariant (within roundoff) to translations in the y-direction as long as the vorticity remains within the computational domain.

2. Fundamentals. We assume an unbounded incompressible flow where the vorticity field, $\boldsymbol{\omega} = \nabla \times \mathbf{v}$, consists of a collection of vortex filaments with Gaussian cross-section. This field at time t may be represented as follows [2]:

$$\boldsymbol{\omega}(\mathbf{r}) = \sum_i \Gamma_i \oint G(\mathbf{r} - \mathbf{r}_i(\xi)) \frac{\partial \mathbf{r}_i}{\partial \xi} d\xi, \tag{1}$$

where ξ is a parameter which traces each filament along its length, $\mathbf{r}_i(\xi)$ are the space curves describing the filaments with circulations Γ_i, and G indicates the Gaussian profile of the filaments. The summation is over individual vortex filaments. The governing

* Received by the editors October 22, 1980.
† Institute for Plasma Research, Stanford University, Stanford, California 94305.
‡ NASA-Ames Research Center, Moffett Field, California 94035.

dynamic equation for the vorticity in these filaments is

$$\frac{D\boldsymbol{\omega}}{Dt} = \boldsymbol{\omega}\cdot\nabla\mathbf{v} + \nu\nabla^2\boldsymbol{\omega}$$

where ν is the kinematic viscosity and the velocity field $\mathbf{v}$ is determined kinematically from

$$\nabla^2\mathbf{v} = -\nabla\times\boldsymbol{\omega}. \tag{2}$$

The motion of the vortex filaments is given by the velocity field averaged over the core of the filaments, or

$$\frac{\partial\mathbf{r}_i(\xi)}{\partial t} = \mathbf{u}(\mathbf{r}_i), \tag{3a}$$

where

$$\mathbf{u}(\mathbf{r}) = \iiint G(\mathbf{r}-\mathbf{r}')\mathbf{v}(\mathbf{r}')\,d\mathbf{r}'. \tag{3b}$$

Note that the computed scales of motion are assumed to be essentially inviscid. Any viscous or subgrid scale dissipation effects are modeled through the filter function G.

3. Vorticity distribution. As a first step towards solving Poisson's equation (2) on a fixed mesh we must create a mesh record of the vorticity field starting from the Lagrangian representation (1). We follow the procedure of our previous paper [2] of discretizing the vortex filaments and distributing the results onto the mesh.

Assume the vorticity field in each periodic domain is produced by a single filament of circulation Γ whose space curve $\mathbf{r}(\xi)$ is approximated by piecewise linear sections between m consecutive node points $\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_m$. From (1) the approximate vorticity field is then

$$\boldsymbol{\omega}(\mathbf{r}) = \Gamma\sum_{j=1}^{m}(\mathbf{r}_j-\mathbf{r}_{j-1})\tfrac{1}{2}[G(\mathbf{r}-\mathbf{r}_j^+)+G(\mathbf{r}-\mathbf{r}_j^-)] \tag{4}$$

where $\mathbf{r}_0 = \mathbf{r}_m$ and where we have assumed two-point Gauss–Legendre quadrature to integrate over each linear section. The quadrature points are given by

$$\mathbf{r}_j^{\pm} = \tfrac{1}{2}[(1\pm\sqrt{\tfrac{1}{3}})\mathbf{r}_j + (1\mp\sqrt{\tfrac{1}{3}})\mathbf{r}_{j-1}].$$

Rather than using (4) to obtain the vorticity at each mesh point, we use quadratic spline weighting to the nearest 27 grid points (3 in each direction). This procedure improves computational efficiency and also insures conservation of $\int\boldsymbol{\omega}\,d\mathbf{V}$ for each linear section of the filament. On the other hand, this method of distributing vorticity onto the mesh implicitly involves its own characteristic filter. Therefore a correction must be applied to yield the effect of the desired filter function $G(\mathbf{r})$. In the two spatial directions, x and z, where Fourier transforms are used, this correction is derived as follows [1]. In the x-direction, for example, the Fourier transform of the filter functions in (4) produces the factors $\exp(ik_x x_j^{\pm})\hat{G}(k_x)$, where $\hat{G}(k_x)$, as mentioned earlier, is taken to be a component of the spherically symmetric Gaussian function, that is $\hat{G}(k_x) = \exp(-k_x^2\sigma^2/4)$. The phase factor $\exp(ik_x x_j^{\pm})$ can be approximated in terms of quadratic splines as

$$\exp(ik_x x_j^{\pm}) \approx S(k_x)\sum_{l=-N_x/2+1}^{N_x/2}\phi_l(x_j^{\pm})\exp(ik_x l) \tag{5}$$

where N_x is the number of mesh points in x for each period and the ϕ_l are the quadratic

spline basis functions given by

$$\phi_l(x)=\begin{cases}\frac{1}{2}\left(\frac{3}{2}+\frac{x}{h}-l\right)^2, & l-\frac{3}{2}\leqq\frac{x}{h}\leqq l-\frac{1}{2},\\ \frac{3}{4}-\left(\frac{x}{h}-l\right)^2, & l-\frac{1}{2}\leqq\frac{x}{h}\leqq l+\frac{1}{2},\\ \frac{1}{2}\left(\frac{3}{2}-\frac{x}{h}+l\right)^2, & l+\frac{1}{2}\leqq\frac{x}{h}\leqq l+\frac{3}{2},\\ 0, & \text{elsewhere.}\end{cases} \tag{6}$$

The mesh spacing $h=1$. Note that the sum in (5) reduces to contributions from the three mesh points nearest to $x_j^{\pm}$. The function $S(k_x)$ is chosen to minimize the rms error of the approximation of all positions $x_j^{\pm}$ between mesh points [1] and is found to be

$$S(k_x)=\left(\frac{2}{k_x}\sin\frac{k_x}{2}\right)^3\Big/D(k_x), \tag{7a}$$

where

$$D(k_x)=1-\sin^2\frac{k_x}{2}+\frac{2}{15}\sin^4\frac{k_x}{2}. \tag{7b}$$

Thus in k-space, the correction amounts to multiplication of the transformed vorticity field by $S(k)\hat{G}(k)$ for each of the two periodic directions. Before the inverse transformation is performed to determine the filament velocity $\mathbf{u}(\mathbf{r})$, we again multiply by $S(k)\hat{G}(k)$ for each periodic direction. The resulting factor $\hat{G}^2(k)$ corresponds to the filtering operations of (1) and (3b) and the consecutive multiplications by $S(k)$ yield respectively the best-fit quadratic spline coefficients of $\boldsymbol{\omega}$ and $\mathbf{u}$ at the mesh points rather than the values of $\boldsymbol{\omega}$ and $\mathbf{u}$.

In the y-direction normal to the flow, we use finite element approximations for $\boldsymbol{\omega}$ and $\mathbf{u}$ in terms of quadratic spline functions, as we will describe in the next section. Thus a quadrature point y' yields a contribution to the vorticity field at y given by

$$H(y,y')=\sum_{l=-\infty}^{\infty}\phi_l(y)\phi_l(y'). \tag{8}$$

Again we want to eliminate the effect of the filter H and impose G. Fourier transformation of (8) over the infinite y' domain gives

$$\int_{-\infty}^{\infty}H(y,y')\exp(ik_y y')\,dy'=\sum_l\phi_l(y)\hat{\phi}(k_y)\exp(ik_y l),$$

where $\hat{\phi}=((2/k_y)\sin(k_y/2))^3$. Using the approximation (5), we find

$$\int_{-\infty}^{\infty}H(y,y')\exp(ik_y y')\,dy'\approx\frac{\hat{\phi}(k_y)}{S(k_y)}\exp(ik_y y)$$

or

$$\int_{-\infty}^{\infty}H(y,y')\exp(ik_y(y'-y))\,dy'\approx D(k_y).$$

Therefore, consistent with the approximation (5), we find that H is approximately a

difference kernel

$$H(y, y') \approx H(y - y'),$$

whose Fourier transform is $D(k_y)$. The k-space representation of the correction for the y-direction is then $\hat{G}^2(k_y)/D(k_y)$. Finally, since

$$\hat{G}^2(k_y)/D(k_y) \approx 1 - \left(\frac{\sigma^2}{2} - \frac{1}{4}\right) k_y^2 \tag{9a}$$

for small k_y, we can approximate this correction by the differential operator $1 + b\,\partial^2/\partial y^2$, where

$$b = \frac{\sigma^2}{2} - \frac{1}{4}. \tag{9b}$$

4. Variational principle. Following the analysis of the previous section where we use FFT's in the streamwise x-direction and the spanwise z-direction, Poisson's equation (2) becomes

$$\frac{\partial^2 \mathbf{u}}{\partial y^2}(k_x, y, k_z) - c\mathbf{u}(k_x, y, k_z) = -\left(1 + b\frac{\partial^2}{\partial y^2}\right)\mathbf{L}\boldsymbol{\omega}(k_x, y, k_z), \tag{10}$$

where $c = k_x^2 + k_z^2$ and $\mathbf{L}$ is the linear operator defined by

$$\mathbf{L}\boldsymbol{\omega} = \left(\frac{\partial \omega_z}{\partial y} - ik_z\omega_y\right)\mathbf{e}_x + (ik_z\omega_x - ik_x\omega_z)\mathbf{e}_y + \left(ik_x\omega_y - \frac{\partial \omega_x}{\partial y}\right)\mathbf{e}_z. \tag{11}$$

$\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are the unit vectors and $i \equiv \sqrt{-1}$. Note also that, from now on, $\mathbf{u}$ and $\boldsymbol{\omega}$ stand for the bi-transformed fields in x and z. Through a variational principle derivation, it is easy to show that (10) is the necessary condition for the integral

$$I(\mathbf{u}) = \int \frac{1}{2}\left[\left(\frac{\partial \mathbf{u}}{\partial y}\right)^2 + c\mathbf{u}^2 - 2\mathbf{u}\left(1 + b\frac{\partial^2}{\partial y^2}\right)\mathbf{L}\boldsymbol{\omega}\right] dy \tag{12}$$

to be a minimum. A finite element approximation of the solution $\mathbf{u}$ in the y-direction can be obtained by using the piecewise quadratic spline functions that interpolate the fields from and onto the mesh. The approximate vorticity and velocity fields are then of the form

$$\boldsymbol{\omega}(k_x, y, k_z) = \sum_{l=-\infty}^{\infty} \boldsymbol{\omega}_l(k_x, k_z)\phi_l(y),$$

$$\mathbf{u}(k_x, y, k_z) = \sum_{l=-\infty}^{\infty} \mathbf{u}_l(k_x, k_z)\phi_l(y),$$

where $\boldsymbol{\omega}_l$ and $\mathbf{u}_l$ are the spline coefficients obtained at the mesh points y_l and the ϕ_l are the basis functions defined by (6). Considering, for example, the x-component of the fields, the stationary value to the functional I is given by

$$\frac{\partial}{\partial u_j} I\left(\sum_{l=-\infty}^{\infty} u_l\phi_l(y)\right) = 0, \qquad -\infty < j < \infty,$$

leading to

$$\sum_{l=-\infty}^{\infty} u_l \int \left[\left(\frac{\partial \phi_l}{\partial y}\right)\left(\frac{\partial \phi_j}{\partial y}\right) + c\phi_l\phi_j\right] dy$$

$$= \sum_{l=-\infty}^{\infty} \left[\omega_{zl} \int \phi_j \left(\frac{\partial \phi_l}{\partial y} + b\frac{\partial^3 \phi_l}{\partial y^3}\right) dy - ik_z\omega_{yl} \int \phi_j \left(\phi_l + b\frac{\partial^2 \phi_l}{\partial y^2}\right) dy\right].$$

Straightforward evaluation of the integrals yields the equation

$$\alpha(u_{j+2} + u_{j-2}) + \beta(u_{j+1} + u_{j-1}) + \gamma u_j$$

$$= \left(\frac{5}{12} - b\right)(\omega_{zj+1} - \omega_{zj-1}) + \left(\frac{1}{24} + \frac{b}{2}\right)(\omega_{zj+2} - \omega_{zj-2})$$

$$- \frac{ik_z}{120}[(66 - 120b)\omega_{yj} + (26 + 40b)(\omega_{yj+1} + \omega_{yj-1}) + (1 + 20b)(\omega_{yj+2} + \omega_{yj} - 2)]$$

where

$$\alpha = \frac{c}{120} - \frac{1}{6}, \quad \beta = \frac{26c}{120} - \frac{1}{3}, \quad \gamma = \frac{66c}{120} + 1. \tag{13}$$

Similar equations for the two other components can be obtained using (11).

For practical purposes, one needs to solve the above pentadiagonal linear system in a finite domain, thus requiring proper boundary treatment. In our case, with a $32 \times 33 \times 32$ mesh and the use of FFT's in the x- and z-direction, we assume no vorticity outside our computational domain in the y-direction. Therefore the boundary conditions necessary to solve the finite pentadiagonal system, given by (13) with $|j| \leqq 16$, are obtained by considering the homogeneous system of equations:

$$\alpha(u_{j+2} + u_{j-2}) + \beta(u_{j+1} + u_{j-1}) + \gamma u_j = 0 \quad \text{for } |j| > 16.$$

These equations admit Z-transform solutions of the form ρ^j. The ρ's are functions of α, β and γ. Only the exponentially decaying solutions are kept to match the velocity at infinity. Subsequent reduction of the infinite system to the finite computational system of equations results in the modification of the four upper and four lower terms of the pentadiagonal matrix with symmetry preserved. Finally, the system to be solved is of the form

$$\begin{pmatrix} \gamma'' & \beta' & \alpha & & & & & & \\ \beta' & \gamma' & \beta & \alpha & & & & & \\ \alpha & \beta & \gamma & \beta & \alpha & & & & \\ & \alpha & \beta & \gamma & \beta & \alpha & & & \\ & & \alpha & \beta & \gamma' & \beta' & & & \\ & & & \alpha & \beta' & \gamma'' & & & \end{pmatrix} \begin{pmatrix} u_{16} \\ u_{15} \\ \\ \\ u_{-15} \\ u_{-16} \end{pmatrix} = \text{R.H.S.}$$

The singular case where $c = 0$ ($k_x = k_z = 0$) is treated by solving a reduced system and then adding a component of the homogeneous solution to obtain the desired value of u at $y \to +\infty$ or $-\infty$. Solvability of the singular case is assured by the numerical equivalent

of

$$\int_R \frac{\partial \omega_x}{\partial y}\, dy = \int_R \frac{\partial \omega_z}{\partial y}\, dy = 0,$$

where R is the computational range in y.

5. Results. In order to test our approximation of the filter in the y-direction, derived in § 3, we form two layers of vorticity, each of opposite circulation, producing a wake-like flow. Here each layer of vorticity is confined between two parallel planes and the flow is uniform in the two directions x and z parallel to the planes. This distribution of vorticity produces a velocity field that varies in the y-direction. After normalizing with respect to the velocity difference across a layer of vorticity, we obtain a normalized velocity profile $U(y)$ with respect to y.

Figures 1, 2 and 3 display three comparisons of velocity profiles between our fully periodic program (full line) and our new method where the y-direction is nonperiodic (dash line). In each comparison, the filter applied in the y-direction in the periodic case corresponds to a given value of b (see (9b)) for the real-space differential operator used in the nonperiodic case. In Table 1, we describe the three cases. In Fig. 1, we simulate the effect of the filter H alone; in Fig. 2, we assume no filtering and in Fig. 3, only G is effectively acting. For the Gaussian filter G, $\sigma^2 = 12/\pi^2$. All three figures show very good agreement confirming the analysis of § 3. Best agreement is obtained in the case of Fig. 1 where the approximation given in (9a) is not required. In Fig. 2, with no filter, we see what is approximately a truncated Fourier expansion of a ramp function velocity distribution—the truncation or cutoff due to the finite mesh width. From Fig. 3, it is

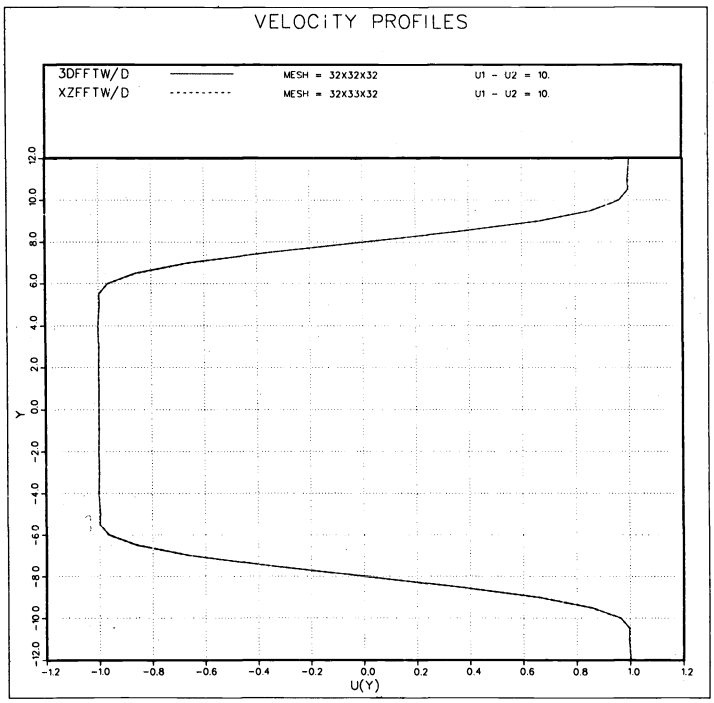


FIG. 1. *Velocity profile for the periodic case (full line) with H alone and for the nonperiodic case (dash line) with* $b = 0$.

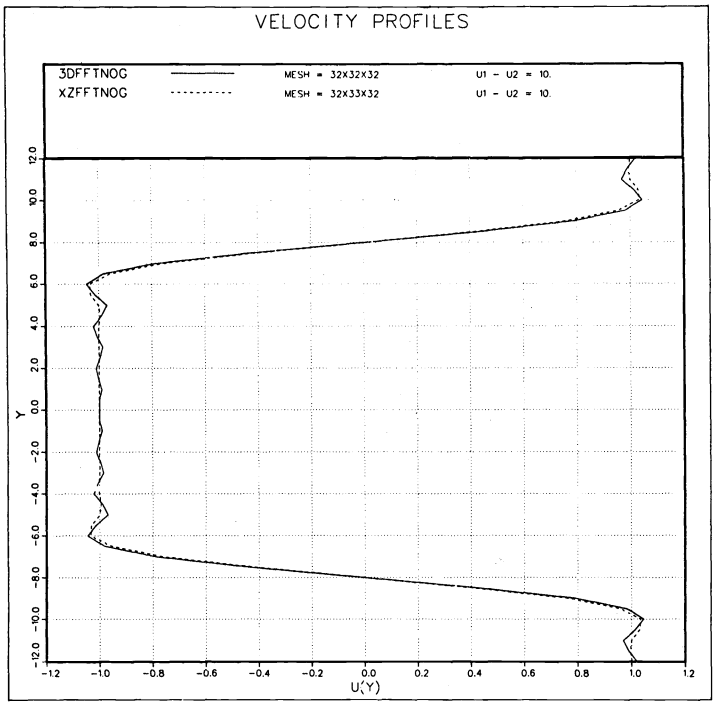


FIG. 2. *Velocity profile for the periodic case (full line) with no filter and for the nonperiodic case (dash line) with* $b = -\frac{1}{4}$.

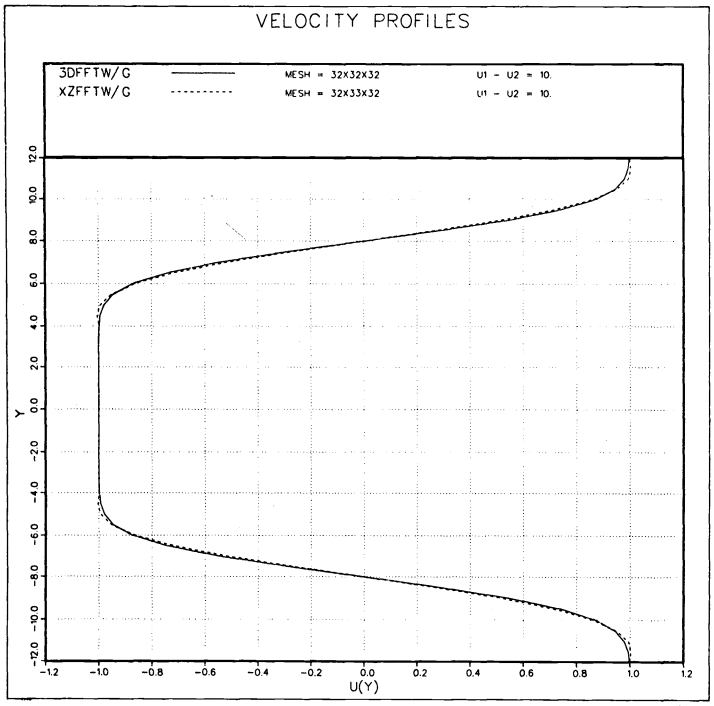


FIG. 3. *Velocity profile for the periodic case (full line) with G and for the nonperiodic case (dash line) with* $b = \sigma^2/2 - \frac{1}{4}$.

TABLE 1

Figures	Nonperiodic (value of b)	Periodic (filter in y-direction)	Max Difference (% of max $U(y)$)
1	0	$D(k_y)$	0.22
2	$-\frac{1}{4}$	1	3.1
3	$\frac{\sigma^2}{2}-\frac{1}{4}$	$\hat{G}^2(k_y)$	3.2

clear that for practical purposes, b could be adjusted empirically in order to improve the fit. This could be accomplished by slightly altering the value of σ^2.

To investigate the truncation error due to the mesh and test the nonperiodic condition in the y-direction, we computed the velocity field due to an infinite row of equidistant line vortices, each of strength Γ, whose (x, y)-coordinates are $(0, 0), (\pm a, 0), (\pm 2a, 0), \cdots$. The exact solution for this system of singular vortices is given by [3]:

$$u = \frac{-\Gamma}{2a}\frac{\sinh(2\pi y/a)}{\cosh(2\pi y/a)-\cos(2\pi x/a)},$$
$$v = \frac{\Gamma}{2a}\frac{\sin(2\pi x/a)}{\cosh(2\pi y/a)-\cos(2\pi x/a)}. \tag{14}$$

These expressions yield $u = \mp\frac{1}{2}\Gamma/a$, $v = 0$, for $y \to \pm\infty$; as viewed from a distant point, the row of vortices becomes equivalent to a vortex-sheet of uniform strength Γ/a. In our calculation, we place a single filament at $(0, 0)$: periodic conditions in x and z provide us with the infinite row of equidistant vortices in z. Thus $a = 32$; we set $\Gamma = 2$ and $b = 0$. Figures 4 and 5 respectively display u and v in the fourth quadrant of the (x, y)-plane.

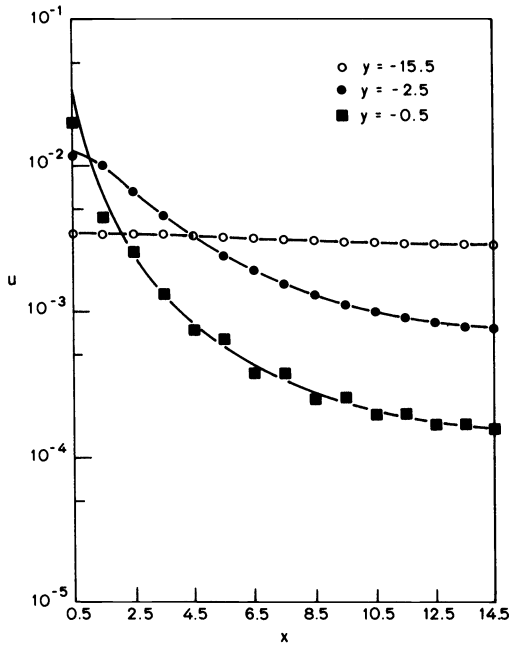


FIG. 4. *x-component of velocity, u, in the fourth quadrant of the* (x, y)*-plane.*

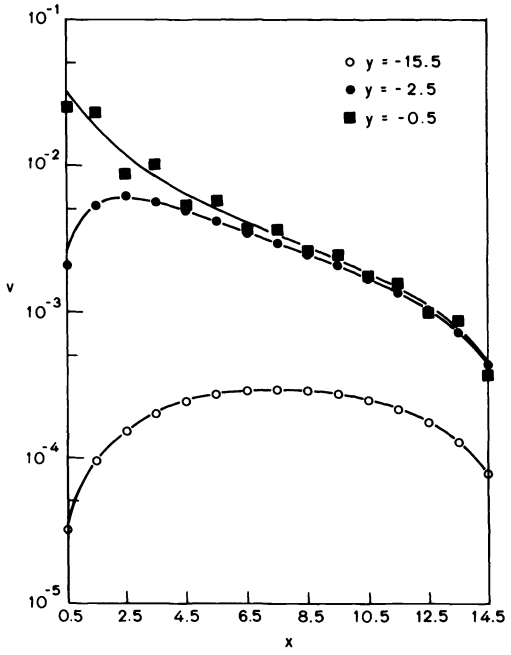


FIG. 5. *y-component of velocity, v, in the fourth quadrant of the (x, y)-plane.*

The full lines indicate the exact solution given by (14). Close to the boundary of the computational domain, at $y = -15.5$, the maximum error between theory and calculations is 0.00089% for u and 0.015% for v. Closer to the filaments, the error increases due to the fact that the mesh does not resolve the high-wavenumber components of the exact solution for the singular filaments. To test the translation invariance of the numerical solution, the vortex was moved in unit steps along the y-axis to $y = 13$. The results were exactly the same within roundoff (14 digits on a CDC 7600). When we place the vortex at $y = 14$, some discrepancies occur due to the fact that vorticity is spread outside the computation domain by the spline functions (see (13)).

6. Summary and conclusions. The three-dimensional vortex-in-cell method has been extended to allow for the simulation of infinite plane layers of vortical fluid bounded on each side by a semi-infinite domain of unsteady potential flow. Exact boundary conditions for the Poisson's equation for the velocity field were derived for the computational domain enclosing all vorticity-containing fluid.

Two new developments were required. First, the filtering implicit in the use of distributing Lagrangian quantities onto a fixed mesh and interpolating back onto the Lagrangian points was determined. A correction that cancels the effect of this filter and imposes the desired filter could then be applied. Second, the solution to the Poisson's equation required the solutions to a set of infinite pentadiagonal linear systems. These equations were solved analytically in the potential flow regions, allowing the derivation of equivalent finite pentadiagonal systems.

The results concerning the inherent filtering of the distribution and interpolation processes could be used to improve a number of particle-in-cell algorithms. In addition, the derivation of exact boundary conditions for Poisson's equation could be used more generally, for example, whenever the source term (vorticity) vanishes in a semi-infinite domain.

REFERENCES

[1] B. COUËT, *Evolution of Turbulence by Three-Dimensional Numerical Particle-Vortex Tracing*, Stanford University Institute for Plasma Research Report 793, 1979.

[2] B. COUËT, O. BUNEMAN AND A. LEONARD, *Simulation of three-dimensional incompressible flows with a vortex-in-cell method*, J. Comp. Phys., 39 (1981), pp. 305–328.

[3] H. LAMB, *Hydrodynamics*, Dover, New York, 1945.

[4] A. LEONARD, *Vortex methods for flow simulation*, J. Comp. Phys., 37 (1980), pp. 289–335.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0006 $01.00/0

MODIFIED DIAGONALLY IMPLICIT RUNGE-KUTTA METHODS*

ZAHARI ZLATEV†

Abstract. The experimental evidence indicates that the implementation of Newton's method in the numerical solution of systems of ordinary differential equations (ODE's) $y'=f(t, y)$, $y(a)=y_0$, $t\in[a, b]$ by implicit computational schemes may cause difficulties. This is especially true if (i) $f(t, y)$ and/or $f'_y(t, y)$ are quickly varying in t and/or y and (ii) a low degree of accuracy is required. Such difficulties may also arise when diagonally implicit Runge–Kutta methods (DIRKM's) are used in the situation described by (i) and (ii). In this paper some modified DIRKM's (MDIRKM's) are derived. The use of MDIRKM's is an attempt to improve the performance of Newton's method in the case where f and f'_y are quickly varying only in t. The stability properties of the MDIRKM's are studied. An error estimation technique for the new methods is proposed. Some numerical examples are presented.

Key words. ordinary differential equations (ODE's), numerical solution, Runge–Kutta methods, diagonally implicit schemes, order of accuracy, quasi-Newton iterative process, Gaussian elimination, matrix factorizations per step, starting approximations, absolute stability, *AN*-stability, *LN*-stability, error estimation, embedding, computational work per step, solving linear systems of ODE's

1. Introduction.

Consider the initial value problem for first order systems of ordinary differential equations (following Stetter [23] we shall call this problem IVP1):

$$y'=f(t, y),\quad y(a)=y_0,\quad t\in[a, b]\subset\mathbb{R},\quad y\in(\mathbb{C}^{(p+1)}[a, b])^s, \tag{1.1}$$

where s and p are positive integers.

Denote the true solution of the IVP1 by $y(t)$. Consider the grid

$$\mathbb{G}_N=\{t_\nu\in[a, b]/\nu=0(1)N,\ t_0=a,\ t_\nu<t_{\nu+1}\text{ for }\nu=0(1)N-1,\ t_N=b\}. \tag{1.2}$$

Very often numerical methods are used to obtain approximations y_ν to $y(t_\nu)$ at the points of the grid $\mathbb{G}_N$ according to some error tolerance ε. The methods introduced by Nørsett [18] will be discussed in this paper. Following Alexander [1] we shall call these methods diagonally implicit Runge–Kutta methods (DIRKM's). An m-stage DIRKM is based on the formulae

$$k_i(h_{n+1})=f\Big(t_n+\alpha_i h_{n+1},\ y_n+h_{n+1}\Big(\sum_{j=1}^{i-1}\beta_{ij}k_j(h_{n+1})+\gamma k_i(h_{n+1})\Big)\Big),\qquad i=1(1)m, \tag{1.3}$$

$$y_{n+1}=y_n+h_{n+1}\sum_{i=1}^{m}p_i k_i(h_{n+1}), \tag{1.4}$$

where $h_{n+1}=t_{n+1}-t_n$ is the stepsize used at step $n+1$ ($n=0(1)N-1$). The coefficients of (1.3)–(1.4) are often written in the form

$$\begin{array}{c|cccccc} \alpha_1 & \gamma & & & & & \\ \alpha_2 & \beta_{21} & \gamma & & & & \\ \vdots & \vdots & \vdots & \ddots & & & \\ \alpha_m & \beta_{m1} & \beta_{m2} & \cdots & \beta_{mm-1} & \gamma & \\ \hline & p_1 & p_2 & \cdots & p_{m-1} & p_m & \end{array}\quad\text{or}\quad \begin{array}{c|c} \alpha & B \\ \hline & p^T \end{array}. \tag{1.5}$$

* Received by the editors March 18, 1980, and in final revised form March 16, 1981. This work was sponsored in part by the Department of Computer Science, Aarhus University, Aarhus, Denmark.

† Department of Computer Science, Aarhus University, Aarhus, Denmark. Currently at the National Agency of Environmental Protection, Air Pollution Laboratory, DK-4000 Roskilde, Denmark.

It should be mentioned here that the following equalities hold for the DIRKM's:

$$\alpha_i = \sum_{j=1}^{i} \beta_{ij}, \quad (\beta_{ii} = \gamma), \quad i = 1(1)m, \tag{1.6}$$

but for the methods considered in this paper (1.6) will not be satisfied.

It should be emphasized that the functions k_i depend not only on the stepsize but also on the independent variable t_n. Therefore, it is more correct to use the notation $k_i(t_n + \alpha_i h_{n+1})$. Abbreviating this notation to $k_i(h_{n+1})$, we want to underline the important fact that if the first j ($j < m$) functions (1.3) have already been computed by some iterative procedure and if the stepsize has to be changed because the iterative procedure fails to converge for k_{j+1}, then the first j functions (1.3) have to be recalculated.

The order of the method described by (1.3)–(1.4) or (1.5) can be defined as follows. Let

$$t_n = x, \quad h_{n+1} = h, \quad \Delta y = y(x+h) - y(x). \tag{1.7}$$

Assume that $y_n = y(x)$. Consider

$$\varphi_m(h) = \Delta y - h \sum_{i=1}^{m} p_i k_i(h). \tag{1.8}$$

Use the Taylor expansion of $\varphi_m(h)$ $(0 < \theta < 1)$

$$\varphi_m(h) = \sum_{j=0}^{p} \left(\frac{h^j}{j!}\right) \varphi_m^{(j)}(0) + \left(\frac{h^{p+1}}{(p+1)!}\right) \varphi_m^{(p+1)}(\theta h). \tag{1.9}$$

The order of the method is p when

$$\varphi_m^{(j)}(0) = 0 \quad \text{for } j = 1(1)p, \qquad \varphi_m^{(p+1)}(0) \neq 0. \tag{1.10}$$

Assume now that a DIRKM of order $p \geqq 1$ is used in the numerical integration of (1.1). In general, some iterative process must be used in the computation of $k_i(h)$, $i = 1(1)m$, because (1.3) are implicit. The quasi-Newton iterative process (QNIP) is commonly used in the integration codes. The use of QNIP in the solution of (1.3) is assumed from now on. Moreover, it is assumed that the simple Gaussian elimination (GE) is applied in the decomposition (the LU factorization) of the matrix $I - h\gamma f'_y$ (see § 2). It is well known that very often an old decomposition (obtained at some previous step j, $j < n$) can also be used at step n. Some problems where a new decomposition is normally computed only when the stepsize is changed can be constructed and arise in practice. Strategies which attempt to keep the old decomposition even after small changes in the stepsize have also been proposed, and it has been verified that they work perfectly for some problems (1.1). Unfortunately, there also arise situations where the old decomposition cannot be used during more than one step. For some problems (especially when a low degree of accuracy is required) even several decompositions per step are needed. This is true not only when DIRKM's are used, but also for many other implicit methods. Two examples are given below in order to show that the average number of decompositions per step can be larger than one.

In Table 1 the numerical results given in Enright et al. [13, p. 23] are used to compute the average numbers of decompositions per steps for 5 codes and for 3 values of the error tolerance. A wide range of test-problems is used in [13]. It should be mentioned that the numerical results for some of the test-problems are not taken into

TABLE 1

The average numbers of the decompositions per step for the 5 codes tested by Enright et al. [13, p. 23]. The codes are based on backward differentiation formulae (GEAR), *the trapezoidal rule with extrapolation* (TRAPEX), *second derivative multistep formulae* (SDBASIC), *a fully implicit Runge–Kutta method* (IMPRK) *and a generalized Runge–Kutta technique* (GENRK).

Tolerance	GEAR	SDBASIC	TRAPEX	IMPRK	GENRK
10^{-2}	0.27	1.47	1.72	6.67	2.67
10^{-4}	0.15	0.89	1.00	0.84	1.99
10^{-6}	0.09	0.61	0.55	0.23	1.87

account in Table 1. This is so, for example, for problem D6. The code IMPRK uses about 24.92 decompositions per step in the integration of D6 with $\varepsilon = 10^{-2}$ (see [13, p. 46]).

The numerical results obtained by SIRKUS (a code based on DIRKM's derived in [18]) in the integration of two chemical problems [2], [15] are shown in Table 2. Note that for the bigger problem ($s = 63$) the average numbers of decompositions per step are larger.

The results in Table 1 and Table 2 show that it is worthwhile attempting to answer the following questions. When can an old decomposition be used several times? If the problem is such that more than one decomposition per step will be needed when a DIRKM is used, what can be done in order to improve the performance of the DIRKM under consideration?

The following definitions will be useful in our efforts to answer the above questions.

TABLE 2

The average numbers of decompositions per step found in the integration of two chemical problems by the code SIRKUS *which is based on* DIRKM's.

Tolerance	$s = 15$	$s = 63$
10^{-1}	1.79	2.27
10^{-2}	0.53	1.75
10^{-3}	0.12	0.89

DEFINITION 1.1. The IVP1 has *property S* if $f(t, y)$ and $f'_y(t, y)$ are slowly varying in t and y.

DEFINITION 1.2. The IVP1 has *property* $\bar{S}$ if at least one of the functions $f(t, y)$ and $f'_y(t, y)$ is quickly varying in t and both functions are slowly varying in y.

DEFINITION 1.3. The IVP1 has *property* S^* if at least one of the functions $f(t, y)$ and $f'_y(t, y)$ is quickly varying in t and at least one of these functions is quickly varying in y.

In § 2 a theorem proved by Kantorovich in 1956 (see [16], [17]) is modified for the use of the QNIP in the solution of (1.3), when (1.1) is solved by a DIRKM. The theorem indicates that the QNIP can cause difficulties in the numerical integration when the IVP1 has not property S. Some modified DIRKM's (MDIRKM's) are derived in § 3. The stability properties of the MDIRKM's are discussed in § 4. An error estimation technique is proposed in § 5. Some applications of the MDIRKM's for

linear IVP's 1 are given in § 6. A brief discussion of the results is presented in the last section.

2. On the use of Newton's method in connection with DIRKM's. Assume that some approximations $k_i^0(h)$, $i=1(1)m$, to the solutions of (1.3) are available (only in this section the notation $k_i^*(h)$ will be used for the solution of the ith system (1.3)). Let (for $i=1(1)m$ and $q=0, 1, \cdots$)

$$\bar{f}'_y(\tau, \eta) \approx f'_y(t_n+\alpha_i h, y_n + h\sum_{j=1}^{i-1} \beta_{ij} k_j(h) + h\gamma k_i^q(h)). \tag{2.1}$$

Then the QNIP can be applied in the solution of (1.3) as follows:

$$[I - h\gamma \bar{f}'_y(\tau, \eta)][k_i^{q+1}(h) - k_i^q(h)] = P(k_i^q(h)), \tag{2.2}$$

$$P(k_i(h)) = k_i(h) - f\Big(t_n+\alpha_i h, y_n + h\sum_{j=1}^{i-1} \beta_{ij} k_j(h) + h\gamma k_i(h)\Big). \tag{2.3}$$

For the QNIP the following theorem holds.

THEOREM 2.1. *Assume that*

$$\Gamma = [I - h\gamma \bar{f}'_y(\tau, \eta)]^{-1} \tag{2.4}$$

exists. Let the following conditions be satisfied when $k_i^0(h) \in \Omega_i$ (*where* Ω_i *is the closed sphere defined by* $\|k_i(h) - k_i^0(h)\| < r_i$, $i=1(1)m$):

$$\|\Gamma P(k_i^0(h))\| \leqq \bar{\eta}_i, \qquad i=1(1)m; \tag{2.5}$$

$$\|I - \Gamma P'(k_i^0(h))\| \leqq \delta_i, \qquad i=1(1)m; \tag{2.6}$$

$$\|\Gamma P''(k_i(h))\| \leqq K_i, \qquad k_i(h) \in \Omega_i, \quad i=1(1)m. \tag{2.7}$$

Then we have:

(i) *Existence and uniqueness. If*

$$\bar{h}_i = \frac{K_i \bar{\eta}_i}{(1-\delta_i)^2} < 0.5, \qquad \delta_i < 1, \quad i=1(1)m, \tag{2.8}$$

$$r_i \geqq \frac{(1-\sqrt{1-2\bar{h}_i})(1-\delta_i)}{K_i}, \qquad i=1(1)m, \tag{2.9}$$

then for any $i \in \{1, 2, \cdots, m\}$ (1.3) *has a solution* $k_i^*(h) \in \Omega_i$, *which is unique if*

$$r_i < \frac{(1+\sqrt{1-2\bar{h}_i})(1-\delta_i)}{K_i}, \qquad i=1(1)m. \tag{2.10}$$

(ii) *Convergence. If* (2.5)–(2.10) *hold, then the* QNIP *is convergent* (*i.e.*, $k_i^q(h) \in \Omega_i$, $i=1(1)m$, $q=0, 1, \cdots$, *and* $k_i^q(h) \to k_i^*(h)$ *as* $q \to \infty$).

(iii) *Speed of convergence. If* $k_i^q(h)$ *is found by the* QNIP, *then* (*for* $i=1(1)m$ *and* $q=0, 1, \cdots$)

$$\|k_i^*(h) - k_i^q(h)\| \leqq \frac{[1-(1-\delta_i)\sqrt{1-2\bar{h}_i}]^{q+1}}{K_i}. \tag{2.11}$$

The above theorem is a modification of a result proved in [16] (see also [17, Chap. XVIII]). Similar results can be found in [19] (where some conditions containing

the eigenvalues of f'_y are used, see [19, p. 28]). We prefer the formulation given by Kantorovich because it is very simple and allows us immediately to draw some conclusions about the qualitative behavior of the QNIP. Indeed, note that (2.6) measures the failure of Γ to be a good approximation to $[P'(k_i^0(h))]^{-1}$, and (2.5) measures the failure of $k_i^0(h)$ to be a good starting approximation. When the problem has property S both δ_i and $\bar{\eta}_i$ will normally be small ($\bar{\eta}_i$ are small because the extrapolation rules which are commonly used for obtaining starting approximations $k_i^0(h)$ work in general well in this case; δ_i are small because Γ is a good approximation to $[P'(k_i^0(h))]^{-1}$ even if it is calculated in a previous step $j<n$). This means that the *strategy of keeping the old decomposition will work well when* (1.1) *has property S.* If the IVP1 has property $\bar{S}$ or property S^* and if, in addition, ε is large, then the above strategy may cause difficulties. The results will be poorer when an attempt to keep the old decomposition even after small changes of the stepsize is carried out (this leads to a large number of rejected steps). If one of the above strategies is combined with restrictions in the changes of the stepsize, then the algorithms so found may perform very badly. However, the same algorithm can be efficient if the IVP1 has property S and/or if ε is small. See, for example, the performance of IMPRK for problem D6 [13, p. 46]. When $\varepsilon = 10^{-2}$ the results are catastrophic: 5657 decompositions and 231 steps. When $\varepsilon = 10^{-6}$ the results are much better, 69 decompositions and 15 steps (note too that the computing time is reduced by a factor larger than 100).

The above analysis shows that Nørsett's condition $\beta_{ii} = \gamma$ for $i = 1(1)m$ [18] normally ensures that at most one decomposition per step is needed if the problem has property S. However, if the IVP1 has property $\bar{S}$ or S^* and if ε is large, then one should be prepared for an integration process where more than one decomposition per step will be needed even if the matrix $I - h\gamma f'_y(t_n + \alpha_1 h, y_n + h\gamma k_1^0(h))$ is decomposed at the beginning of each step. This is very unfortunate if the system is large (the computational cost per decomposition is $O(s^3)$ simple arithmetic operations, while the computational cost of the QNIP without the decompositions is $O(s^2)$). It should also be pointed out that the transformation of (1.1) to autonomous form will not change the situation. If the nonautonomous problem (1.1) has not property S, then the transformed autonomous problem has not property S either. Moreover, in many practical problems t has a special physical meaning, and it is desirable to keep t as independent variable. Finally, if the problem is linear, then the transformation to autonomous form may cause some extra computations (because the transformed problem will in general be nonlinear).

Theorem 2.1 and the above analysis indicate that an attempt to improve the performance of the DIRKM's in the case where the problem has not property S may be worthwhile.

3. Modified diagonally implicit Runge–Kutta methods. Replace (1.6) with the condition $\alpha_i = \gamma^*$ for $i = 1(1)m$. The method so found will be called a modified DIRKM (MDIRKM) if it has the same order as the *corresponding* DIRKM (the DIRKM which has the same coefficients γ, β_{ij} and p_i but the coefficients α_i satisfy (1.6)). An answer to the question whether MDIRKM's can be constructed is given by the following theorem.

THEOREM 3.1. MDIRKM's *of order up to* 2 *can be constructed.*

Proof. (a) *Order* 2 *is attainable.* Consider (1.10). It is obvious that $\varphi_m(0) = 0$ is satisfied. Since (see (1.3)–(1.10))

$$(\Delta y)' = \frac{d(\Delta y)}{dh} = y'(x+h) = f(x+h, y(x+h)), \tag{3.1}$$

$$\frac{dk_i(h)}{dh} = \gamma^* f'_t\Big(x+\gamma^* h,\, y_n + h\sum_{j=1}^{i}\beta_{ij}k_j(h)\Big)$$
$$+ f'_y\Big(x+\gamma^* h,\, y_n + h\sum_{j=1}^{i}\beta_{ij}k_j(h)\Big)\Big[\sum_{j=1}^{i}\beta_{ij}k_j(h) + h\sum_{j=1}^{i}\beta_{ij}\Big(\frac{dk_j(h)}{dh}\Big)\Big], \tag{3.2}$$

$$\varphi'_m(h) = f(x+h,\, y(x+h)) - \sum_{i=1}^{m} p_i k_i(h) - h\sum_{i=1}^{m} p_i\Big(\frac{dk_i(h)}{dh}\Big), \tag{3.3}$$

it is clear that

$$\varphi'_m(0) = \Big(1 - \sum_{i=1}^{m} p_i\Big) f(x, y(x)), \tag{3.4}$$

and therefore $\varphi'_m(0) = 0$ implies

$$\sum_{i=1}^{m} p_i = 1. \tag{3.5}$$

From

$$(\Delta y)'' = \frac{d^2(\Delta y)}{dh^2} = f'_t(x+h,\, y(x+h)) + f'_y(x+h,\, y(x+h)) f(x+h,\, y(x+h)), \tag{3.6}$$

$$\varphi''_m(h) = (\Delta y)'' - 2\sum_{i=1}^{m} p_i\Big(\frac{dk_i(h)}{dh}\Big) - h\sum_{i=1}^{m} p_i\Big(\frac{d^2k_i(h)}{dh^2}\Big), \tag{3.7}$$

it follows that

$$\varphi''_m(0) = \Big(1 - 2\gamma^* \sum_{i=1}^{m} p_i\Big) f'_t(x, y(x)) + \Big(1 - 2\sum_{i=1}^{m} p_i \sum_{j=1}^{i}\beta_{ij}\Big) f'_y(x, y(x)) f(x, y(x)), \tag{3.8}$$

and therefore $\varphi''_m(0) = 0$ implies

$$\gamma^* = 0.5 \quad \text{and} \quad \sum_{i=2}^{m} p_i \sum_{j=1}^{i-1}\beta_{ij} = 0.5 - \gamma. \tag{3.9}$$

It is readily seen that the coefficients of the method can be chosen so that (3.5) and (3.9) are satisfied (and the order is 2). If, for example, $m = 2$, then

$$\begin{array}{c|cc} 0.5 & \gamma & \\ 0.5 & (0.5-\gamma)/p & \gamma \\ \hline & 1-p & p \end{array} \tag{3.10}$$

can easily be found.

(b) *No* MDIRKM *of order* 3 *can be constructed.* This is trivial; no quadrature formula based on one point can be of order higher than 2. □

Assume that the IVP1 has property $\bar{S}$ and that ε is large. For all $\alpha_i = \gamma^*$, then one could expect $\bar{f}'_y(\tau, \eta)$ to be a good approximation to all matrices in the right-hand side of (2.1) and the QNIP will perform well at all stages during step n. If (1.6) are satisfied the above statements will often not hold. This shows that the MDIRKM's are introduced in an attempt to improve the performance of the QNIP when the IVP1 has property $\bar{S}$. However, if the IVP1 is linear, then the MDIRKM's are efficient not only when the problem has property $\bar{S}$ but also when the problem has property S^*, see § 6.

4. Stability properties of the MDIRKM's. Let q be a complex constant. Assume that $\operatorname{Re} q \leqq 0$ and consider the model-equation

$$y' = qy, \qquad y \in \mathbb{R}. \tag{4.1}$$

It is well known (see, e.g., [5]) that the use of any RK method in the solution of (4.1) leads to

$$y_{n+1} = R(z)y_n, \qquad (n = 1(1)N), \tag{4.2}$$

where (see (1.5))

$$R(z) = 1 + zp^T(I - zB)^{-1}e, \quad z = h_{n+1}q, \quad e = (1, 1, \cdots, 1)^T. \tag{4.3}$$

The method is A-stable [12] if

$$|R(z)| \leqq 1 \quad \forall \operatorname{Re}(z) \leqq 0. \tag{4.4}$$

Since (4.3) does not depend on α, the following result is clear.

THEOREM 4.1. *An* MDIRKM *is A-stable if and only if the corresponding* DIRKM *is A-stable.*

If, for example, $m = 2$, then it is well known that for the DIRKM's

$$R(z) = [1 + (1 - 2\gamma)z + (\gamma^2 - 2\gamma + 0.5)z^2]/(1 - \gamma z)^2 \tag{4.5}$$

and the methods are A-stable if $\gamma \geqq 0.25$. Theorem 4.1 shows that this result holds also for the corresponding MDIRKM's (with $m = 2$).

It has already been mentioned that the MDIRKM's are efficient in the solution of linear systems (1.1), i.e., when

$$f(t, y) = A(t)y + b(t). \tag{4.6}$$

Therefore, it seems to be useful to investigate the AN-stability of the MDIRKM's. The notion AN-stability was introduced by Burrage and Butcher in [5]. Let $q(t)$ be a continuous complex-valued function with $\operatorname{Re} q(t) \leqq 0$ for $t \in [a, b]$. Consider the nonautonomous model-equation

$$y' = q(t)y, \qquad q(t) \in \mathbb{R}. \tag{4.7}$$

The implementation of any RK method to (4.7) leads to

$$y_{n+1} = K(Z)y_n, \quad Z = \operatorname{diag}(z_1, z_2, \cdots, z_m), \quad z_i = h_{n+1}q(t_n + \alpha_i h_{n+1}), \quad i = 1(1)m, \tag{4.8}$$

where (see (1.5) again)

$$K(Z) = 1 + p^T Z(I - BZ)^{-1}e, \qquad e = (1, 1, \cdots, 1)^T. \tag{4.9}$$

The method is said to be AN-stable if

$$|K(Z)| \leqq 1 \quad \forall \operatorname{Re}(z_i) \leqq 0, \quad i = 1(1)m. \tag{4.10}$$

While AN-stability implies A-stability, the converse statement is, in general, not true (see the example given in [5, p. 49]). However, for the MDIRKM's the following result holds.

THEOREM 4.2. *AN-stability is equivalent to A-stability for the* MDIRKM's.

A remarkably simple criterion for AN-stability has been given by Burrage and Butcher [5]. This criterion can be described as follows. Consider the symmetric matrix M whose elements are $m_{ij} = p_i\beta_{ij} + p_j\beta_{ji} - p_ip_j$. For all $p_i \geqq 0$, if M is a semipositive definite matrix, then the RK method is said to be algebraically stable.

THEOREM 4.3 (Burrage and Butcher [5, *p.* 50]. *An algebraically stable* RK *method is AN-stable and, if* $\alpha_1, \cdots, \alpha_m$ *are distinct, it conversely holds that an AN-stable* RK *method is algebraically stable.*

The use of Theorem 4.3 to the 2-stage DIRKM's and the corresponding MDIRKM's gives

THEOREM 4.4. *The 2-stage* MDIRKM *described by* (3.10) *and the corresponding* DIRKM *are algebraically stable if* $p = 0.5$ *and* $\gamma \geqq 0.25$.

Proof. Matrix M is semipositive definite if (i) $\det(M) \geqq 0$, (ii) $m_{11} \geqq 0$ and (iii) $m_{22} \geqq 0$. Condition (i) leads after straightforward computations to $-4(0.5-\gamma)^2(0.5-p)^2 \leqq 0$ and (since $\gamma = 0.5$ produces a 1-stage method) $p = 0.5$. With $p = 0.5$, $m_{11} = m_{22} = \gamma - 0.25$ and (ii) and (iii) lead to $\gamma \geqq 0.25$. □

Theorem 4.3 and Theorem 4.4 give immediately the following result.

COROLLARY 4.1. *The 2-stage* DIRKM*'s are AN-stable if* $p = 0.5$ *and* $\gamma \geqq 0.25$.

By the use of Theorem 4.1 and the established fact about the A-stability of the 2-stage DIRKM's the following result can easily be obtained.

COROLLARY 4.2. *The 2-stage* MDIRKM*'s are AN-stable if* $\gamma \geqq 0.25$.

Corollary 4.1 and Corollary 4.2 show that if $p \neq 0.5$ and $\gamma \geqq 0.25$ then the 2-stage MDIRKM is AN-stable, while the corresponding DIRKM is only A-stable.

A situation where the use of an A-stable method in the solution of nonautonomous equations causes instability is given below. Consider the equation

$$y' = A \sin^2\left(\frac{\pi t}{c} - 3.430251901\right) y, \qquad A < 0, \quad c > 0. \tag{4.11}$$

Assume that the 2-stage DIRKM given by

$$\begin{array}{c|cc} 1-\sqrt{2}/2 & 1-\sqrt{2}/2 & \\ 27\sqrt{2}/2-18 & 14\sqrt{2}-19 & 1-\sqrt{2}/2 \\ \hline & (53-5\sqrt{2})/62 & (9+5\sqrt{2})/62 \end{array} \tag{4.12}$$

is applied in the solution of (4.11) with a constant stepsize $h = c$. A simple analysis shows that (4.12) will be unstable if $-Ac > 37.1$. The numerical results obtained for $A = -10{,}000$, $h = c = 0.1$, $y(0) = 1{,}000$ are given in Table 3. The results obtained with two other methods (which are AN-stable) are also given in Table 3. These methods are:

$$\begin{array}{c|cc} 1 & 1 & \\ 0 & -1 & 1 \\ \hline & 0.5 & 0.5 \end{array} \qquad \begin{array}{cc|c} 1-\sqrt{2}/2 & 1-\sqrt{2}/2 & \\ \sqrt{2}/2 & \sqrt{2}-1 & 1-\sqrt{2}/2 \\ \hline & 0.5 & 0.5 \end{array} \tag{4.13}$$

TABLE 3

The errors found in the numerical integration of (4.11) (E1 *is the error found by* (4.12), E2 *and* E3 *are the errors for the first and the second method* (4.13), *respectively*).

t	Number of steps	E1	E2	E3
1.0	10	5.95E+5	5.93E 0	6.11E−17
2.0	20	3.54E+7	3.52E−3	3.73E−37
3.0	30	2.11E+9	2.09E−6	2.28E−57
4.0	40	1.25E+11	1.24E−9	3.26E−66
5.0	50	7.45E+12	7.35E−13	0.0

For the 2-stage DIRKM's (when implemented in the solution of (4.7)),

$$K(Z)=\frac{[1+(1-p-\gamma)z_1+(p-\gamma)z_2+(\gamma^2-2\gamma+0.5)z_1z_2]}{(1-\gamma z_1)(1-\gamma z_2)}, \tag{4.14}$$

where (when a constant stepsize is used)

$$z_i = hq(t_n-\alpha_i h), \qquad i=1,2. \tag{4.15}$$

For the method (4.12), $\gamma^2-2\gamma+0.5=0$ and the choice $h=c$ gives $z_2\approx 0$ at each step. Therefore it must be emphasized that the above example is very artificially created. Nevertheless, the example shows that the use of *AN*-stable methods in the solution of nonautonomous problems should be preferred; see also the second experiment in § 6.

Compare the results for the two methods (4.13). The second method follows the behavior of the solution much better. This shows that it seems to be useful to introduce *LN*-stability (*L*-stability for nonautonomous problems).

DEFINITION 4.1. The notation $|Z|\to\infty$ will be used to express the fact that $|z_i|\to\infty$ for all i.

DEFINITION 4.2. The RK method is said to be *LN-stable* if it is *AN*-stable and if $|Z|\to\infty$ implies $K(Z)\to 0$.

The relations between the different kinds of stability for the RK methods when they are applied to (4.1) and (4.7) are seen in Fig. 4.1.

L-stability	$\Rightarrow$	*A*-stability
$\Uparrow$		$\Uparrow$
LN-stability	$\Rightarrow$	*AN*-stability

FIG. 4.1

The following corollaries can be formulated and proved.

COROLLARY 4.3. *For the* MDIRKM*'s LN-stability and L-stability are equivalent.*

COROLLARY 4.4. *The 2-stage* DIRKM*'s are LN-stable if* $p=0.5$ *and* $\gamma=1\pm\sqrt{2}/2$.

COROLLARY 4.5. *The 2-stage* MDIRKM*'s are LN-stable if* $\gamma=1\pm\sqrt{2}/2$.

As an example let us note that the MDIRKM corresponding to (4.12) is *LN*-stable, while (4.12) is not.

The results in this section show that not only a better performance of the QNIP for problems which have property $\bar{S}$ can be expected when the MDIRKM's are used (see § 3), but also, the classes of *AN*-stable and *LN*-stable 2-stage MDIRKM's are considerably larger than the classes of *AN*-stable and *LN*-stable 2-stage DIRKM's.

LN-stable implicit Runge–Kutta methods are also considered by Burrage [4]. This concept is extended for a wider class of methods in [6].

5. Error estimation technique. A device which can be used to control the local truncation error during the integration process performed by some 2-stage MDIRKM's of order 2 will be described in this section. The device is based on "embedding" which was first used by Fehlberg [14] for explicit RK methods. The following statements, which are well known and only slightly modified for our methods, are needed before the formulation of the main result in this section (Theorem 5.2).

DEFINITION 5.1. Consider the problem defined by

$$y'=f(t,y), \qquad y(t_n)=y_n. \tag{5.1}$$

Assume that an m-stage RK method (not necessarily an MDIRK or a DIRKM) of order p is used to find y_{n+1}. Then

$$T_{n+1}^{p} = (\varphi_m^{(p+1)}(0)/(p+1)!)h^{p+1}, \qquad h = h_{n+1} \tag{5.2}$$

will be called the *principal part* of the local truncation error.

THEOREM 5.1. *Assume that y_{n+1} is computed by a 2-stage* MDIRKM *of order 2. Consider another Runge–Kutta method of order 3 defined as follows: $k_1(h)$ and $k_2(h)$ are the vectors computed by the* MDIRKM *under consideration,*

$$k_i(h) = f\Big(t_n + \alpha_i h,\ y_n + h \sum_{j=1}^{m} \beta_{ij} k_j(h)\Big), \qquad i = 3(1)m, \tag{5.3}$$

$$\hat{y}_{n+1} = y_n + h \sum_{i=1}^{m} \hat{p}_i k_i(h). \tag{5.4}$$

Then if the terms which contain h^4 are neglected in (1.9) *the principal part of the local truncation error can be written in the following way:*

$$T_{n+1}^{2} = \hat{y}_{n+1} - y_{n+1} = h \sum_{i=1}^{2} (\hat{p}_i - p_i) k_i(h) + h \sum_{i=3}^{m} \hat{p}_i k_i(h). \tag{5.5}$$

The problem is how to choose the auxiliary method (5.3)–(5.4). It is not possible to construct an MDIRKM of order 3 (see § 3). It is not desirable to use implicit formulae in (5.3) (this may cause extra decompositions). Therefore the only choice which will ensure that the computational cost of the error estimator formulae (5.3)–(5.4) is $O(s^2)$ is $\beta_{ij} = 0$ for $i = 3(1)m$ and $j \geqq i$. By this choice the following theorem can be proved.

THEOREM 5.2. *The smallest number m which allows us to construct an error estimator* (5.3)–(5.4) *with explicit formulae* (5.3) *for a 2-stage* MDIRKM *is* 4.

Proof. The method (5.4) will be of order 3 if its coefficients satisfy the following conditions.

$$\sum_{i=1}^{m} \hat{p}_i = 1, \tag{5.6}$$

$$\sum_{i=1}^{m} \hat{p}_i \alpha_i = 0.5, \tag{5.7}$$

$$\sum_{i=1}^{m} \hat{p}_i \sum_{j=1}^{i} \beta_{ij} = 0.5, \tag{5.8}$$

$$\sum_{i=1}^{m} \hat{p}_i \alpha_i^2 = \frac{1}{3}, \tag{5.9}$$

$$\sum_{i=1}^{m} \hat{p}_i \alpha_i \sum_{j=1}^{i} \beta_{ij} = \frac{1}{3}, \tag{5.10}$$

$$\sum_{i=1}^{m} \hat{p}_i \sum_{j=1}^{i} \alpha_j \beta_{ij} = \frac{1}{6}, \tag{5.11}$$

$$\sum_{i=1}^{m} \hat{p}_i \sum_{j=1}^{i} \beta_{ij} \sum_{\nu=1}^{j} \beta_{j\nu} = \frac{1}{6}, \tag{5.12}$$

$$\sum_{i=1}^{m} \hat{p}_i \Big(\sum_{j=1}^{i} \beta_{ij}\Big)^2 = \frac{1}{3}. \tag{5.13}$$

(a) Let us choose $m=3$. Then it is easily seen that the system (5.6)–(5.13) has no solution (consider (5.6), (5.7) and (5.9) and take into account that (3.5) and (3.9) must also be satisfied).

(b) Let us choose $m=4$. Assume that $\gamma, \beta_{21}, \beta_{31}, \beta_{32}, \alpha_3$ and α_4 are chosen so that (5.13) is satisfied. Then the solution of (5.6)–(5.12) can be found (for $\beta_{21}\neq 0$, $\alpha_3\neq\alpha_4$, $\alpha_3\neq 0.5$ and $\alpha_4\neq 0.5$) by the use of the following formulae (and the notation $\bar{\alpha}=(2\alpha_4-1)/(2\alpha_3-1)$, $\bar{\beta}=\beta_{31}+\beta_{32}-\gamma$, $\bar{\gamma}=1-6\gamma+6\gamma^2$):

$$\beta_{42}=\{(\alpha_4-\alpha_3)[\bar{\gamma}(2\alpha_4-1)+\gamma]+\bar{\alpha}(\beta_{32}\beta_{21}-\bar{\beta}\gamma)+[\bar{\alpha}(\alpha_4-\alpha_3)-\gamma]\bar{\beta}\}/\beta_{21}, \tag{5.14}$$

$$\beta_{43}=-\bar{\alpha}(\alpha_4-\alpha_3), \tag{5.15}$$

$$\beta_{41}=\alpha_4-\alpha_3-\beta_{42}-\beta_{43}+\beta_{31}+\beta_{32}, \tag{5.16}$$

$$\hat{p}_4=\tfrac{1}{6}(2\alpha_4-1)(\alpha_4-\alpha_3), \tag{5.17}$$

$$\hat{p}_3=-\bar{\alpha}\hat{p}_4, \tag{5.18}$$

$$\hat{p}_2=[0.5-\gamma-(\hat{p}_3+\hat{p}_4)\bar{\beta}-\hat{p}_4(\alpha_4-\alpha_3)]/\beta_{21}, \tag{5.19}$$

$$\hat{p}_1=1-\hat{p}_2-\hat{p}_3-\hat{p}_4. \tag{5.20}$$

Straightforward calculations show that (5.13) can be rewritten as

$$2\bar{\beta}(2\alpha_3-1-\bar{\beta}+\beta_{21})=(2\alpha_3-1)\{[1+\bar{\gamma}-(3-6\gamma)\beta_{21}](2\alpha_4-1)-\alpha_4+\alpha_3+\beta_{21}\}. \tag{5.21}$$

It is easy to see that (5.21) can be satisfied (for example by the choice of $\beta_{32}=0$, $\beta_{31}=\gamma$, $\alpha_3=\alpha_4-\beta_{21}-[1+\bar{\gamma}-(3-6\gamma)\beta_{21}](2\alpha_4-1)$). □

Consider now the integration method as a combination of a basic 2-stage MDIRKM of order 2 and a 4-stage error estimator (5.3)–(5.4) of order 3. If (1.1) is linear, then the requirement (5.13) is not necessary (this requirement appears when the coefficient in the term containing f''_{yy} is equated to zero). In this case the six parameters $\gamma, \beta_{21}, \beta_{31}, \beta_{32}, \alpha_3$ and α_4 could be used in order to construct an integration method which is optimal with regard to some of the following requirements: *accuracy, stability, computational work per step and simple implementation.* If (1.1) is not linear then at least one of the parameters must be used in order to satisfy (5.13). The other parameters can again be used in an attempt to optimize the integration method.

6. Application of MDIRKM's in the solution of linear systems. Assume that: (i) *the* IVP1 *is linear,* (ii) *the* IVP1 *has property* $\bar{S}$ *or* S^*, (iii) ε *is large.* In this section we shall show that in this situation the QNIP can successfully be replaced by the use of GE and, moreover, if the use of GE is assumed, then it is preferable to apply MDIRKM's. Some numerical results obtained by Y12NBF will be used to illustrate our conclusions. Therefore, before the discussion, a brief description of this code is needed (some more details are given in [21]). The integration method implemented in the code is given by

$$\begin{array}{c|cccc} 0.5 & 1-\sqrt{2}/2 & & & \\ 0.5 & \sqrt{2}-1 & 1-\sqrt{2}/2 & & \\ 0 & 0 & 0 & 0 & \\ 1 & -\sqrt{2}+1 & \sqrt{2}-1 & 1 & 0 \\ \hline p_i & 0.5 & 0.5 & & \\ \hat{p}_i & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \frac{1}{6} \end{array} \tag{6.1}$$

By this choice of the parameters γ, β_{21}, β_{31}, β_{32}, α_3 and α_4, an attempt to construct an integration scheme which is optimal with regard to the computational work per step and to the simplicity of implementation has been carried out. Since the QNIP is replaced by GE, the most expensive parts of the computational work per successful step are one decomposition, two function calls and two back substitutions. The computational work needed to obtain the coefficient matrix for $k_1(i)$ and $k_2(i)$ is reduced from $O(s^2)$ to $O(4s)$ arithmetic operations by the use of matrix $\bar{A} = (1-\sqrt{2}/2)^{-1}h^{-1}A$ instead of $A = I - h(1-\sqrt{2}/2)A(t_n + h/2)$. Vector $k_3(h)$ is computed before the computation of $k_1(h)$ and $k_2(h)$. In this way it is not necessary to recompute $k_3(h)$ when the step is rejected and has to be repeated with a smaller stepsize. The step is not rejected immediately after obtaining $\|\hat{y}_{n+1} - y_{n+1}\|_2 > \varepsilon$. First the code will attempt to perform extrapolation as follows. Vectors y_{n+2} and y^*_{n+2} are calculated using starting values y_{n+1} and y_n and stepsizes h and $2h$, respectively. Only the basic method (the 2-stage MDIRKM) is used in these calculations. The approximation y_{n+2} is accepted if $\|y_{n+2} - y^*_{n+2}\|_2/7 \leqq \varepsilon$ and in the next several steps only this extrapolation rule is used. If the extrapolation rule fails, then the stepsize $h_{n+1} = h$ is reduced and y_{n+1} is recomputed by (6.1). During the use of the extrapolation rule, the code takes 2 small steps and 1 large step. When we count the steps we give the number of the small steps. The code uses 1.5 decompositions, 3 back substitutions and 1.5 function calls per successful small step when the extrapolation rule is used.

Now we are ready to present some numerical results. Two very simple but illustrative examples are given below.

Example 6.1. Consider the problem

$$y' = A \sin^2\left(\frac{t}{40}\right)(y - 0.01t) + 0.01, \qquad y(0) = 1, \quad t \in [0, 100], \quad A < 0. \tag{6.2}$$

The exact solution of the problem is $y(t) = 0.01t + \exp[A(t/2 - 10 \sin(t/20)]$. If $-A$ is small, then f and f'_y are not very quickly varying in t, but when $-A$ is large they are.

The results given in Table 4 show that the computational work in the integration process practically does not depend on the magnitude of parameter A.

Example 6.2. Let us consider a more stringent problem:

$$y' = A \sin^2\left(\frac{t}{2}\right)(y - t) + 1, \qquad y(0) = 1, \quad t \in [0, 100], \quad A < 0. \tag{6.3}$$

TABLE 4

Numerical results obtained in the integration of (6.2). *Error tolerance* $\varepsilon = 10^{-1}$.

$-A$	Steps	Decompositions	Function calls	Substitutions	Accuracy
0.5	15	15	31	30	2.5E−2
50	18	24	35	59	5.0E−2
500	18	27	32	54	7.4E−2
10^5	19	29	33	58	5.1E−2
10^{10}	17	27	29	54	5.0E−2

The exact solution of the problem is $y(t) = t + \exp[A(t/2 - \sin t/2)]$. If $-A$ is large the functions f and f'_y are varying very quickly in t; moreover, f is also varying in y.

It is seen from Table 5 that the computational work increases when $-A$ becomes large, but not very fast.

TABLE 5
Numerical results obtained in the integration of (6.3). *Error tolerance* $\varepsilon = 10^{-1}$.

$-A$	Steps	Decompositions	Function calls	Substitutions	Accuracy
0.01	15	15	31	30	$2.2E-2$
50	28	42	48	84	$1.0E-1$
10^3	31	47	52	94	$1.1E-1$
10^6	39	57	66	114	$1.0E-1$

An MDIRKM (as implemented in Y12NBE) has also been used in the solution of some problems of chemical origin ([21], [22]) for which all assumptions made at the beginning of this section hold. The problems and some previous experiments concerning these chemical problems are described in [20].

An implementation of this MDIRKM for linear problems with large and sparse matrices $A(t)$ has also been developed [22]. The sparse matrix algorithm is based on ideas described in [24], [25], [26], [28]. Numerical examples, with s up to 255, are given in [22].

A code based on the use of MDIRKM's and designed for nonlinear problems which have property $\bar{S}$ is under preparation at RECKU (the Regional Computing Centre at the University of Copenhagen).

7. Some concluding remarks. It is necessary to emphasize that the MDIRKM's will be efficient only when the IVP1 has property $\bar{S}$ (also property S^* if the problem is linear) and when ε is large. If this is not so then the DIRKM's of order $p \geqq 2$ may perform better. If the error tolerance is stringent then the code STRIDE [7], [10], [11], which is based on singly-implicit Runge–Kutta methods ([3], see also [9]; these methods are derived by the use of a transformation proposed in [8]) implemented in a variable stepsize variable formula manner, will work much better than any MDIRKM (whose order cannot exceed 2). This means that the MDIRKM's *must be used carefully*. If the problem is large and the user can establish that the nonlinear problem which has to be solved has property $\bar{S}$, then the use of MDIRKM's will normally be efficient (and sometimes very efficient). Often the problem has property $\bar{S}$ only on a part of the integration interval. If this is so the MDIRKM's should be used in conjunction with some other methods (STRIDE). The use of MDIRKM's with linear problems which have property $\bar{S}$ and even S^* is also very efficient. If the problem is not large (say, $s \leqq 10$), if ε is large and if the QNIP is replaced by GE (as in Y12NBF), then the MDIRKM's will be efficient also for linear problems which have property S; the computational cost of the decompositions is not very large (in comparison with the computational cost for the back substitutions) and the extra decompositions will be compensated by a great reduction of the numbers of function calls and back substitutions. If the problem is linear and large, then matrix $A(t)$ is usually sparse and some sparse technique can easily be implemented, and the use of MDIRKM's is again efficient if the above conditions are satisfied. Note that large linear problems (1.1) arise often in practice (e.g., in the solution of some parabolic partial differential equations [27], or in chemistry [22], and an investigation of the properties of the problem may result in a considerable improvement of the efficiency of the numerical integration when the right method is chosen.

Acknowledgments. Section 4 has been written following the constructive comments on a previous version of this paper made by an unknown referee. The author would like to thank him very much for the helpful suggestions.

REFERENCES

[1] R. ALEXANDER, *Diagonally implicit Runge–Kutta methods for stiff* ODE's, SIAM J. Numer. Anal., 14 (1977), pp. 1006–1021.

[2] H. BILDSØE, J. P. JACOBSEN AND K. SCHAUMBURG, *Application of density matrix formalism in* NMR *spectroscopy* I. *Development of a calculation scheme and some simple examples*, J. Magnet. Resonance, 23 (1976), pp. 137–151.

[3] K. BURRAGE, *A special family of Runge–Kutta methods for solving stiff differential equations*, BIT, 18 (1978), pp. 22–41.

[4] ———, *Stability and efficiency properties of implicit Runge–Kutta methods*, Ph.D. Thesis, Mathematics Department, Auckland University, Auckland, New Zealand, 1978.

[5] K. BURRAGE AND J. C. BUTCHER, *Stability criteria for implicit Runge–Kutta methods*, SIAM J. Numer. Anal., 16 (1979), pp. 46–57.

[6] ———, *Nonlinear stability of a general class of differential equation methods*, BIT, 20 (1980), pp. 185–203.

[7] K. BURRAGE, J. C. BUTCHER AND F. H. CHIPMAN, *An implementation of singly-implicit Runge–Kutta methods*, BIT, 20 (1980), pp. 326–340.

[8] J. C. BUTCHER, *On the implementation of implicit Runge–Kutta methods.* BIT, 16 (1976), pp. 237–240.

[9] ———, *A transformed implicit Runge–Kutta method*, J. Assoc. Comput. Mach., 26 (1979), pp. 731–738.

[10] J. C. BUTCHER, K. BURRAGE AND F. H. CHIPMAN, STRIDE-*stable Runge-Kutta integrator for differential equations*, Computational Mathematics Report, March 1979, Department of Mathematics, University of Auckland, Auckland, New Zealand.

[11] F. H. CHIPMAN, *Some experiments with* STRIDE, Working papers for the 1979 SIGNUM Meeting on Numerical Ordinary Differential Equations, R. D. Skeel, ed., Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 1979.

[12] G. DAHLQUIST, *A special stability problem for linear multistep methods*, BIT, 3 (1963), pp. 27–43.

[13] W. H. ENRIGHT, T. E. HULL AND B. LINDBERG, *Comparing methods for stiff systems of* ODE's, BIT 15 (1975), pp. 10–48.

[14] E. FEHLBERG, *Klassische Runge–Kutta–Formeln vierter und niedrigerer Ordnung mit Schrittweiten-Kontrolle und ihre Anwendung auf Wärmeleitungsprobleme*, Computing, 6 (1970), pp. 61–71.

[15] J. P. JACOBSEN, H. K. BILDSØE AND K. SCHAUMBURG, *Application of density matrix formalism in* NMR *spectroscopy* II. *The one-spin-*1 *case in anisotropic phase.* J. Magnet. Resonance, 23 (1976), pp. 153–164.

[16] L. V. KANTOROVICH, *On integral equations*, Uspekhi Mat. Nauk, 11 (1956), pp. 3–29.

[17] L. V. KANTOROVICH AND G. P. AKILOV, *Functional Analysis in Normed Spaces*, Pergamon Press, Oxford, 1964.

[18] S. P. NØRSETT, *Semiexplicit Runge–Kutta methods*, Mathematics and Computation, No. 6/74, Department of Mathematics, University of Trondheim, Norway.

[19] H. H. ROBERTSON AND J. WILLIAMS, *Some properties of algorithms for stiff differential equations*, J. Inst. Math. Appl., 16 (1975), pp. 23–34.

[20] K. SCHAUMBURG AND J. WASNIEWSKI, *Use of a semiexplicit Runge–Kutta algorithm in a spectroscopic problem*, Comput. Chem., 2 (1978), pp. 19–25.

[21] K. SCHAUMBURG, J. WASNIEWSKI AND Z. ZLATEV, *Solution of ordinary differential equations with time dependent coefficients. Development of a semiexplicit Runge–Kutta algorithm and application to a spectroscopic problem*, Comput. Chem., 3 (1979), pp. 57–63.

[22] 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.

[23] H. J. STETTER, *Analysis of discretization methods for ordinary differential methods.* Springer, Berlin, 1973.

[24] Z. ZLATEV, *Use of iterative refinement in the solution of sparse linear systems*, Report 1/79, Institute of Mathematics and Statistics, The Royal Veterinary and Agricultural University, Copenhagen, 1979, SIAM J. Numer. Anal., to appear.

[25] ———, *On some pivotal strategies in Gaussian elimination by sparse technique*, SIAM J. Numer. Anal., 17 (1980), pp. 18–30.

[26] 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.

[27] Z. ZLATEV AND P. G. THOMSEN, *Application of backward differentiation methods to the finite element solution of time dependent problems*, Int. J. Num. Meth. Engng, 14 (1979), pp. 1051–1061.

[28] 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.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0007 $01.00/0

AN INTERVAL ANALYSIS APPROACH TO RANK DETERMINATION IN LINEAR LEAST SQUARES PROBLEMS*

THOMAS A. MANTEUFFEL†

Abstract. The linear least squares problem $A\mathbf{x} \cong \mathbf{b}$ has a unique solution only if the matrix A has full column rank. Numerical rank determination is difficult, especially in the presence of uncertainties in the elements of A. This paper proposes an interval analysis approach. We define a set of matrices A^I that contains all possible perturbations of A due to uncertainties and say that A^I is rank deficient if any member of A^I is rank deficient. A modification to the QR decomposition method of solution of the least squares problem allows a determination of the rank of A^I and a partial interval analysis of the solution vector $\mathbf{x}$. This procedure requires the computation of R^{-1}. Another modification is proposed which determines the rank of A^I without computing R^{-1}. The additional computational effort is $O(n^2)$, where n is the column dimension of A.

Key words. linear least squares, numerical rank, QR decomposition

1. Introduction.

The linear least squares problem

$$A\mathbf{x} \cong \mathbf{b} \tag{1.1}$$

has a unique solution only if the matrix A is of full column rank. Numerical determination of rank is not always easy, especially in light of uncertainties in the elements of A. It is often the case that $A = (a_{ij})$ is known to be correct only within certain tolerances, and the desired but unknown $\hat{A} = (\hat{a}_{ij})$ satisfies

$$\hat{a}_{ij} = a_{ij} + \psi_{ij}, \qquad |\psi_{ij}| \leqq \max(\gamma_{ij}|a_{ij}|, \zeta_{ij}), \tag{1.2}$$

where γ_{ij}, ζ_{ij} are relative and absolute error bounds. The uncertainty in $\hat{A}$ may cause rather large uncertainties in the solution $\mathbf{x}$. If any $\hat{A}$ in (1.2) is rank deficient, then a numerical solution may be completely unreasonable.

It is essential that algorithms designed to solve linear least squares problems address the question of rank deficiency. A complete answer would require an extensive interval analysis and the arrays of bounds (γ_{ij}) and (ζ_{ij}) (cf. Hanson and Smith, [7], [8]). For the most part this extra computation is unwarranted. In this paper, we address the situation where the matrix is assumed to be of full rank. The most efficient algorithm for this case is the QR decomposition of A based upon Householder reflections (Golub [5]). We will show that a minor modification of this algorithm will yield bounds that help determine the rank of the numerically uncertain matrix A.

The algorithms proposed here yield as much information about the possible rank deficiency of A as a singular value decomposition (Golub and Kahan [4]). The computational cost of a singular value decomposition is difficult to predict because of the iterative part of the algorithm (cf. Lawson and Hanson [10, p. 122]). As a general rule, the total cost is 2 to 8 times that of the fixed part of the algorithm. The algorithms proposed here are less costly than even the fixed portion of the singular value decomposition.

1.1. Notation.

We will use the following notation. Capital letters will denote matrices, boldface small letters will denote vectors, and small letters will denote scalars.

* Received by the editors July 8, 1980.
† Los Alamos National Laboratory, Los Alamos, New Mexico 87545.

The $m \times n$ $(m \geqq n)$ real-valued matrix

$$A = (a_{ij}) = \begin{pmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & & | \end{pmatrix} .$$

has a singular value decomposition

$$A = U\Sigma V^*, \qquad \Sigma = \operatorname{diag}(\sigma_1, \cdots, \sigma_n), \qquad 0 \leqq \sigma_n \leqq \cdots \leqq \sigma_1,$$

where $U_{m\times n}$ and $V_{n\times n}$ are unitary matrices. We will use the vector norms

$$\|\mathbf{v}\|_\infty = \max_i |v_i|,$$

$$\|\mathbf{v}\|_1 = \sum_i |v_i|,$$

$$\|\mathbf{v}\|_2 = \left(\sum_i |v_i|^2\right)^{1/2},$$

and matrix norms

$$\|A\|_\infty = \max_i \sum_j |a_{ij}|,$$

$$\|A\|_1 = \max_j \sum_j |a_{ij}| = \max_j \|\mathbf{a}_j\|_1,$$

$$\|A\|_2 = \max_{\|\mathbf{x}\|_2 = 1} \|A\mathbf{x}\|_2 = \sigma_1,$$

$$\|A\|_F = \left(\sum_{i,j} (a_{ij})^2\right)^{1/2} = \left(\sum_i \sigma_i^2\right)^{1/2}.$$

The 2-norm may appear without subscript. The vector $\mathbf{e}_j$ will denote the jth basis vector and $\mathbf{e} = (1, 1, \cdots, 1)^T$. We will let $C_i(A) = \|A\|_i \|A^+\|_i$, where A^+ is the pseudoinverse of A (Moore–Penrose generalized inverse), be the condition of A in the appropriate norm.

1.2. Problem definition. Consider the set of matrices that includes all perturbations in (1.2). Let

$$A_0^I = \{A + \Psi : |\psi_{ij}| \leqq \max(\gamma_{ij}|a_{ij}|, \zeta_{ij})\}. \tag{1.3}$$

We will say that A_0^I is of full rank if every $\hat{A} \in A_0^I$ is of full rank. A subset of columns of A_0^I will be said to be linearly independent if there exists no $\hat{A} \in A_0^I$ whose corresponding columns are linearly dependent. An independent set of interval columns will be said to be maximal if no larger independent set includes it. The Rank (A_0^I) will be the cardinality of the largest maximal independent set of columns of A_0^I. This yields

$$\operatorname{Rank}(A_0^I) \leqq \min_{\hat{A} \in A_0} \operatorname{Rank}(\hat{A}). \tag{1.4}$$

Notice that $\{\mathbf{a}_i^I, \mathbf{a}_j^I\}$ and $\{\mathbf{a}_i^I, \mathbf{a}_k^I\}$ may both be dependent sets, while $\{\mathbf{a}_j^I, \mathbf{a}_k^I\}$ may be an independent set. In fact, one can construct matrices for which the inequality in (1.4) is strict. However, if the right-hand side equals n, then equality holds.

A desirable result would be to find a largest maximal independent set of columns of A_0^I and produce an interval solution $\mathbf{x}^I$ that contains all possible solutions based upon this maximal set. Such a result is possible but requires an inordinate amount of

computation and storage (cf. Moore [13], Hanson and Smith [7], [8]). The goal of this paper is to obtain as much of this information as possible at a moderate increase in computation and storage over the Businger–Golub QR decomposition method (Businger and Golub [2]).

Consider a larger interval matrix in which the error bounds are the same throughout each column. Let

$$\gamma_j = \max_i |\gamma_{ij}|, \qquad \zeta_j = \max_i |\zeta_{ij}|,$$
$$A_1^I = \{A + \Psi : |\psi_{ij}| \leqq \max(\gamma_j |a_{ij}|, \zeta_j)\}. \tag{1.5}$$

One can picture A_1^I as a matrix each of whose columns is represented by an m-dimensional box in R^m with sides parallel to the axes. The Householder reflections used in the QR decomposition rotate the vector about the origin. The sides of the box may no longer be parallel to the axes. For that reason we enclose the box in a ball. Let

$$\varepsilon_j = \left(\sum_{i=1}^{m} \max(\gamma_j |a_{ij}|, \zeta_j)^2 \right)^{1/2},$$
$$A_2^I = \{A + \Psi : \|\psi_j\|_2 \leqq \varepsilon_j,\ j = 1, \cdots, n\}. \tag{1.6}$$

Finally, consider an interval definition based upon the Frobenius norm. Let

$$\varepsilon = \left(\sum_{j=1}^{n} \varepsilon_j^2 \right)^{1/2},$$
$$A_F^I = \{A + \Psi : \|\Psi\|_F \leqq \varepsilon\}. \tag{1.7}$$

This is the interval matrix underlying methods based upon singular values.

Notice that

$$A_0^I \subseteq A_1^I \subseteq A_2^I \subseteq A_F^I. \tag{1.8}$$

In this paper we will use A_2^I. It has the property that the set of perturbations is invariant to multiplication by a unitary matrix. While rank determination based upon A_2^I is not as precise as that based upon A_0^I, it is preferable to rank based upon A_F^I.

1.3. Overview. We will show in § 2 that, while singular value decomposition yields a partial answer to the rank of A_2^I, an equally definitive answer can be garnered from a QR decomposition and inspection of quantities related to $\|R^{-1}\|_1$. The latter can be established with less computation effort. In § 3, an algorithm will be proposed that constructs a maximal set of independent columns of A_2^I. Further, a partial interval analysis of $\mathbf{x}^I$ will be presented. This algorithm requires the construction of R^{-1} as the QR decomposition proceeds but remains computationally more efficient than singular value decomposition. In § 4, it will be shown that a maximal set of columns may be determined without construction of R^{-1}. The additional computation over the QR decomposition is $O(n^2)$. Section 5 will discuss numerical results.

This work is similar in nature to that of Golub, Klema and Stewart [6]. There the goal is to find an acceptable reduced rank solution based upon A_F^I. Their algorithm compares solutions based upon numerical pseudoinverses of different ranks. Here we assume full rank and only consider reduced rank solutions when rank deficiency is detected. No comparison between reduced rank solutions is presented. Their algorithm requires more work and assumes less information about the uncertainties in A.

The bounds established in § 4 are similar in nature to those of Cline, Moler, Stewart and Wilkinson [3] and the $O(n)$ bounds of Anderson and Karasalo [1] and Karasalo [9]. These papers deal with lower and upper bounds of $C_1(R)$ respectively. The bounds of § 4 also yield upper and lower bounds of $C_1(R)$. The lower bound is similar to that of Cline et al., and the upper bound will be better than that of Anderson and Karasalo. Moreover, upper and lower bounds are constructed for each column. These bounds are used to determine the independence of the columns. When the gap between the upper and lower bounds leaves independence in question, the exact value is computed.

In [1], [3], [9], the upper and lower bounds are used to estimate the smallest singular value, which is in turn used to define numerical rank. One aim of this work is to show that these bounds are as viable as the smallest singular value in detecting rank deficiency. Indeed, the singular value bounds must be based upon A_F^I which may be considerably larger than A_2^I.

This analysis precludes an accurate rank determination when a weighted least squares solution is sought in which the weights vary greatly in magnitude. Singular value analysis fails here as well. This case has been examined by Powell and Reid [15] and Manteuffel [12].

2. Rank of A_2^I. For convenience we will first assume that the columns of the matrix A have been scaled so that the uncertainty is the same in each. This does not affect the solution of (1.1). Later we will lift this restriction. We have

$$A_2^I = \{A + \Psi : \|\psi_j\|_2 \leqq \varepsilon, j = 1, \cdots, n\}. \tag{2.1}$$

Let

$$\Psi_2^I = \{\Psi : \|\psi_j\|_2 \leqq \varepsilon, j = 1, \cdots, n\}. \tag{2.2}$$

2.1. Tests based on singular values. While singular values can be used to accurately determine Rank (A_F^I), they are imprecise for determining the more desirable Rank (A_2^I). We have the following well-known result.

THEOREM 2.1. *If* $\sigma_k \leqq \varepsilon$, *then* Rank $(A_2^I) < k$. *If* $\sigma_k > \varepsilon\sqrt{n}$, *then* Rank $(A_2^I) \geqq k$.

Proof. Notice that if $\Psi \in \Psi_2^I$, then $\|\Psi\|_F \leqq \varepsilon\sqrt{n}$. Further, notice that if $\|\Psi\|_F \leqq \varepsilon$, then $\Psi \in \Psi_2^I$. The theorem follows from well-known results on singular values (cf. Stewart [14, p. 320)]). □

Thus, the singular values will determine the rank deficiency of A_2^I to within $\sqrt{n}$. In general, the $\sqrt{n}$ cannot be omitted.

2.2. Tests based on R^{-1}. Rank tests based upon $\|R^{-1}\|_1$ give similar results. First, let

$$R^{-1} = \Delta = (\delta_{ij}),$$
$$\rho_j = \|\delta_j\|_1 = \sum_{i=1}^{j} |\delta_{ij}|; \tag{2.3}$$

then

$$\rho = \|R^{-1}\|_1 = \max_j \rho_j. \tag{2.4}$$

THEOREM 2.2. *Let* $\{p_{j_l}\}_{l=1}^k$ *be a subset of the* ρ_j*'s. If*

$$\left(\sum_{l=1}^{k} \rho_{j_l}^2\right)^{1/2} < \frac{1}{\varepsilon},$$

then Rank $(A_2^I) \geqq k$.

Proof. Let $A = QR$ be the QR decomposition of A. If we assume R is of full rank, then Rank $(A+\Psi)$ = Rank $(I + Q^*\Psi R^{-1})$. If $k = n$, we have

$$\|\Psi R^{-1}\|_F = \left(\sum_{j=1}^{n} \left\| \sum_{i=1}^{n} \boldsymbol{\psi}_i \delta_{ij} \right\|_2^2\right)^{1/2}$$

$$\leqq \left(\sum_{j=1}^{n} \left(\sum_{i=1}^{n} \|\boldsymbol{\psi}_i\| |\delta_{ij}|\right)^2\right)^{1/2}$$

$$\leqq \varepsilon \left(\sum_{j=1}^{n} \rho_j^2\right)^{1/2} < 1.$$

If $k < n$, consider the submatrix of ΨR^{-1} consisting of those columns associated with the ρ_j's in the subset. Denote this by $(\Psi R^{-1})_k$. Again we have

$$\|(\Psi R^{-1})_k\|_F < 1,$$

which implies that these k columns of $(I + Q^*\Psi R^{-1})$ are linearly independent for every $\Psi \in \Psi_2^I$. The case in which R is singular is proved similarly. □

The next result yields a more convenient form for rank determination.

THEOREM 2.3. *If $\rho \geqq 1/\varepsilon$, then A_2^I is rank deficient. If $\rho < 1/\varepsilon\sqrt{n}$, then A_2^I is of full rank.*

Further, if $k = \text{card}\,\{\rho_j\colon \rho_j < 1/\varepsilon\sqrt{n}\}$, then Rank $(A_2^I) \geqq k$.

Proof. If $A = QR$ is the QR decomposition of A, then the matrix $\hat{A} = A + \Psi$, $\Psi \in \Psi_2^I$, has the same linearly independent sets of columns as the matrix $Q^*(A+\Psi) = R + \Phi$. Note that $\Phi \in \Psi_2^I$.

Assume $\rho_k \geqq 1/\varepsilon$ for some k. Define Ψ such that $\boldsymbol{\psi}_i = -\text{sign}\,(\delta_{ik})/\rho_k Q\mathbf{e}_k$. We have $\Psi \in \Psi_2^I$, and it is easy to show that $(R+\Phi)(R^{-1}\mathbf{e}_k) \equiv \mathbf{0}$. Thus, A_2^I is rank deficient.

The last two results follow from Theorem 2.2. □

This result is sharp in that the $\sqrt{n}$ cannot be omitted. Suppose A is such that $R^{-1} \geqq 0$ and $\rho_j = -1/\varepsilon\sqrt{n}$, $j = 1, \cdots, n$. Then let Ψ be such that $\boldsymbol{\psi}_i = (-\varepsilon/\sqrt{n})Q\mathbf{e}$, $(\mathbf{e} = \sum_{i=1}^{n} \mathbf{e}_i)$. We have $\Psi \in \Psi_2^I$, and

$$(R + Q^*\Psi)(R^{-1}\mathbf{e}) = 0.$$

Thus, A is rank deficient. On the other hand, suppose

$$R^{-1} = \rho \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ 0 & \frac{1}{2} & -\frac{1}{3} \\ 0 & 0 & \frac{1}{3} \end{pmatrix}.$$

One can show that for $1/\varepsilon\sqrt{n} \leqq \rho < 3/\varepsilon\sqrt{22}$ we have Rank $(A_2^I) = n$.

2.3. Implicit scaling. Consider again the interval matrix (1.6) with perhaps different bounds for each column. A singular value analysis without first scaling the columns yields no information about A_2^I. This is not a major drawback, in that column scaling so that the uncertainty is equal in each column is a standard practice. However, this scaling can be handled implicitly when bounds are based upon R^{-1}. Suppose $A \in A_2^I$. Let $E = \text{diag}\,(\varepsilon_1, \cdots, \varepsilon_n)$; then $AE^{-1} \in A_2^I$, where now $\varepsilon = \varepsilon_j = 1$, $j = 1, \cdots, n$. The results of § 2.2 apply with R replaced RE^{-1} and ε replaced by 1. The ρ_j's must be replaced by

$$\rho_j = \|E\boldsymbol{\delta}_j\|_1 = \sum_{i=1}^{j} |\delta_{ij}|\varepsilon_i. \tag{2.5}$$

2.4. Operation count. A singular value decomposition involves an iteration and so it is difficult to accurately predict how much computational effort is required. If the algorithm of Golub and Kahan [4] is used, the fixed computational effort is approximately $2mn^2$ floating point operations.[1] If a QR decomposition is followed by a singular value decomposition, the fixed computation is $mn^2+5n^3/6$ (Lawson and Hanson [10, p. 122]). The QR decomposition requires $mn^2-n^3/3$ operations. Once R is known, the quantities ρ_j in (2.5) may be computed by successive back solves requiring $n^3/6$ operations and n words of storage. Thus, the QR algorithm with rank determination based upon $\|R^{-1}\|_1$ is computationally more efficient than singular value decomposition for determining the rank of A_2^I.

There are other advantages to computing R^{-1}. These will be discussed in § 3. In § 4, bounds will be developed that yield much of the same information with only $O(n^2)$ extra computation.

3. Constructing a maximal set. In this section an algorithm will be presented which constructs a maximal set of independent columns. Also, bounds on the interval solution based upon this maximal set are established.

3.1. The algorithm. Consider the QR decomposition of A. As in the Businger–Golub algorithm (Businger and Golub [2]), the upper triangular matrix $R=A^{(n)}$ is computed in n steps, where

$$A^{(k+1)}=H^{(k)}P_r^{(k)}A^{(k)}P_c^{(k)}, \tag{3.1}$$

$P_r^{(k)}$, $P_c^{(k)}$ are row and column permutation matrices, and $H^{(k)}$ is the Householder reflection matrix that zeros out the subdiagonal elements of the $(k+1)$st column of $P_r^{(k)}A^{(k)}P_c^{(k)}$.

Let us denote

$$A^{(k)}=\begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & R_{22}^{(k)} \end{pmatrix}, \tag{3.2}$$

where $R_{11}^{(k)}$ is $k\times k$ and upper triangular. The elements of $R_{11}^{(k)}$ and $R_{12}^{(k)}$ are fixed (up to column interchanges in $R_{12}^{(k)}$) for step $k+1,\cdots,n$. We can construct the inverse of R as we proceed. Consider the matrix

$$\Delta^{(k)}=(\delta_{ij}^{(k)}), \tag{3.3}$$

where the elements are updated at each step according to the formulas

$$\delta_{kk}^{(k)}=\frac{1}{r_{kk}}; \tag{3.4}$$

for $i=1,\cdots,k-1$,

$$\delta_{ik}^{(k)}=-\delta_{kk}^{(k)}\delta_{ik}^{(k-1)}; \tag{3.5}$$

for $j=k+1,\cdots,n$, for $i=1,\cdots,k-1$,

$$\delta_{ij}^{(k)}=\delta_{ij}^{(k-1)}+\delta_{ik}^{(k)}r_{kj}^{(k)}, \qquad \delta_{kj}^{(k)}=\delta_{kk}^{(k)}r_{kj}^{(k)}. \tag{3.6}$$

We may write

$$\Delta^{(k)}=\begin{pmatrix} \Delta_{11}^{(k)} & \Delta_{12}^{(k)} \\ 0 & \Delta_{22}^{(k)} \end{pmatrix}, \tag{3.7}$$

[1] Here, a floating point operation consists of one addition and one multiplication.

where

$$\Delta_{11}^{(k)} = (R_{11}^{(k)})^{-1}, \qquad \Delta_{12}^{(k)} = (R_{11}^{(k)})^{-1} R_{12}^{(k)} \tag{3.8}$$

and $\Delta_{22}^{(k)}$ is as yet undefined.

After k steps, the first k columns of Δ are fixed. We may compute ρ_j, $j = 1, \cdots, k$ using (2.5). Further, we may examine the columns $k+1, \cdots, n$ in order to determine which column should be ordered next. In the Businger–Golub algorithm the column permutation is chosen to make $r_{k+1,k+1}$ as large as possible. Here we choose column $k+1$ to make ρ_{k+1} as small as possible. Let $\mathbf{s}^{(k)}$ be the vector such that $s_j^{(k)}$ contains the $\|\cdot\|_2$ of the corresponding column of $R_{22}^{(k)}$. For $j = k+1, \cdots, n$ let

$$\rho_j^{(k+1)} = \frac{1}{s_j^{(k)}} \left(\varepsilon_{k+1} + \sum_{i=1}^{k} |\sigma_{ij}^{(k)}| \varepsilon_i \right). \tag{3.9}$$

If column j were to be chosen to be permuted to the $(k+1)$st position, the resulting ρ_{k+1} would be given by $\rho_j^{(k+1)}$. We then choose the column order so that

$$\rho_{k+1}^{(k+1)} \leqq \rho_j^{(k+1)}, \qquad j = k+1, \cdots, n. \tag{3.10}$$

(It should be noted that, even though ρ_k is chosen to be as small as possible at each step, the inequality $\rho_k \leqq \rho_{k+1}$ might not necessarily hold.)

This strategy has two motivations. The first is that it facilitates the construction of a maximal set. Suppose we let $k_2 = n$ and $k_1 = 0$ before the first step. Let us assume that $\rho_j < 1$, $j \leqq k$. If $\rho_j^{(k+1)} > 1$ for some $k < j \leqq k_2$, then the interval columns $\{\mathbf{a}_1^I, \cdots, \mathbf{a}_k^I, \mathbf{a}_j^I\}$ are linearly dependent. Column j may not be used in computing a maximal set. This column should be put in position k_2 and k_2 set to $k_2 - 1$. Only the first k_2 columns need be considered further. Now suppose $\rho_j^{(k+1)} < 1$, $j = k+1, \cdots, k_2$. The column with the smallest of these is placed in the $k+1$st position. Let k_1 be the number of elements of the largest subset of the first $k+1$ columns such that $\sum \rho_j^2 < 1$ for this subset. By Theorem 2.2 and § 2.3, we have $k_1 \leqq \text{Rank}\,(A_2^I)$.

If this process is continued until $k = k_2$, we will have constructed a set of columns to which no other column may be added. If $k_1 = k_2$, this is a maximal set. If $k_1 < k_2$, the set of columns contains a maximal set which in turn contains the subset of columns used to determine k_1. In practice, $k_1 = k_2$ with few exceptions (see § 5).

The second motivation for choosing ρ_{k+1} as small as possible at step $k+1$ stems from a forward analysis of the growth of uncertainty and roundoff error. In Manteuffel [11] it is shown that the column with the smallest relative uncertainty should be reflected out first. The theorems of § 2 may be used to show that the ρ's provide upper and lower bounds on the relative uncertainty. If $\hat{A} = A + \Psi$, $\Psi \in \Psi_2^I$ is decomposed using the same row and column pivot order as A, we will have $\hat{R} = R + \Omega$, where Ω is also upper triangular. From Theorem 2.2 and § 2.3, we have

$$|\omega_{kk}| \leqq \left(\sum_{j=1}^{k} \rho_j^2 \right)^{1/2} |r_{kk}|, \qquad k = 1, \cdots, n, \tag{3.11}$$

where ω_{kk} is the kth diagonal element of Ω. From Theorem 2.3 and § 2.3, we know that for each k there is some $\Psi \in \Psi_2^I$ such that $|\omega_{kk}| \geqq \rho_k |r_{kk}|$. Notice that if the absolute uncertainty were the same in each column, then the column with the smallest relative uncertainty would correspond to the longest column, as in the Businger–Golub algorithm. The choice of the first column affects the size of the uncertainty in the columns of $R_{22}^{(1)}$. If one were to rescale at each step so that the absolute uncertainty

were the same for each column of $R_{22}^{(k)}$, then the algorithm presented here would be similar to that of Businger and Golub.

It is also shown in Manteuffel [11] (see also Powell and Reid [15]) that the growth of uncertainty and roundoff errors may be reduced by choosing $H^{(k+1)}$ to reflect the first column of $R_{22}^{(k)}$ onto the nearest basis element. This may be accomplished by ordering the rows so that the element of the first column of $R_{22}^{(k)}$ with the largest absolute value appears at the top of the column.

The overall cost of this algorithm is mn^2 arithmetic operations. The QR decomposition requires $mn^2 - n^3/3$ operations, construction of Δ requires $n^3/6$ operations and the selection of $\boldsymbol{\rho}$ requires $n^3/6$ operations. This is still less than the fixed number of operations required for a singular value decomposition, which is the smaller of $mn^2 + 5n^3/3$ or $2mn^2$. Note that Δ may be constructed over R so that the only extra storage needed is for the vector $\boldsymbol{\rho}$. The extra computation required to select the column pivot may be unwarranted. In § 4, upper and lower bounds on ρ_k are constructed at a cost of $O(n^2)$ operations and the pivot strategy may be based upon these. In fact, these bounds do not require the construction of Δ. However, Δ may provide information about $\mathbf{x}^I$, the interval solution.

3.2. Bounds on uncertainty. Once $\Delta = R^{-1}$ has been constructed, both $\|R^{-1}\|_1$ and $\|R^{-1}\|_\infty$ may also be computed. The norms $\|R\|_1$ and $\|R\|_\infty$ can be computed during the decomposition. With these and the bound $C_2(A) \leqq \sqrt{C_1(A)C_\infty(A)}$, one can analyze the relative uncertainty in the various norms; that is, the quantity

$$\frac{\|\mathbf{x} - \hat{\mathbf{x}}\|}{\|\mathbf{x}\|},$$

where $\hat{\mathbf{x}}$ is the solution of a perturbed system $\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$, can be bounded (cf. Stewart [14, p. 223]).

We can also look at the term by term uncertainty to bound $\mathbf{x}^I$. Let $\hat{A} = A + \Psi$, $\Psi \in \Psi_2^I$, $\hat{\mathbf{b}} = \mathbf{b} + \boldsymbol{\psi}_{N+1}$, $\|\boldsymbol{\psi}_{N+1}\|_2 \leqq \varepsilon_{N+1}$.

If we multiply both equations by Q we have

$$R\mathbf{x} = \mathbf{g}, \qquad (R + \Phi)\hat{\mathbf{x}} = \mathbf{g} + \mathbf{f},$$

where $\Phi \in \Psi_2^I$, $\|\mathbf{f}_2\| \leqq \varepsilon_{N+1}$. We can write

$$(\mathbf{x} - \hat{\mathbf{x}}) = R^{-1}(\mathbf{f} - \Phi\hat{\mathbf{x}})$$

and

$$\begin{aligned} |x_i - \hat{x}_i| &= \left| \sum_k \delta_{ik} f_k - \sum_j \left(\sum_k \delta_{ik} \phi_{kj} \right) \hat{x}_j \right| \\ &\leqq \sum_k |\delta_{ik}||f_k| + \sum_j \left(\sum_k |\delta_{ik}||\phi_{kj}| \right) |\hat{x}_j| \\ &\leqq \|\boldsymbol{\delta}_i^T\|_2 \left(\varepsilon_{N+1} + \sum_j \varepsilon_j |\hat{x}_j| \right), \end{aligned} \tag{3.12}$$

where $\boldsymbol{\delta}_i^T$ represents the ith row of $\Delta = R^{-1}$. For small perturbations we may replace $|\hat{x}_j|$ by $|x_j|$ on the right-hand side. This interval solution is fairly sharp in that we can always construct a perturbation such that

$$|x_i - \hat{x}_i| \geqq \|\boldsymbol{\delta}_i^T\|_2 \left(\varepsilon_{N+1} + \left| \sum_j \varepsilon_j \hat{x}_j \right| \right). \tag{3.13}$$

Of course, (3.13) cannot be made to hold for all i simultaneously.

4. $O(n^2)$ bounds. In this section, we develop an algorithm that determines a maximal linearly independent set of columns of A_2^I at an additional cost of $O(n^2)$ operations over the QR decomposition.

4.1. Upper and lower bounds. Let us again assume that the columns of A have been scaled so that $\varepsilon = \varepsilon_j$; $j = 1, \cdots, n$. We will lift this restriction later. Since R is upper triangular, it is an H-matrix.[2] The corresponding M-matrix[3] is

$$\bar{R} = (\bar{r}_{ij}) \quad \bar{r}_{ii} = |r_{ii}|, \quad \bar{r}_{ij} = -|r_{ij}|, \qquad i \neq j. \tag{4.1}$$

Let us write

$$\begin{aligned} R &= D(I - T), D = \operatorname{diag}(r_{11}, \cdots, r_{nn}), \\ \bar{R} &= \bar{D}(I - \bar{T}), \bar{D} = \operatorname{diag}(\bar{r}_{11}, \cdots, \bar{r}_{n_n}). \end{aligned} \tag{4.2}$$

Then

$$\begin{aligned} R^{-1} &= (I + T + \cdots + T^{n-1})D^{-1}, \\ \bar{R}^{-1} &= (I + \bar{T} + \cdots + \bar{T}^{n-1})\bar{D}^{-1}, \end{aligned} \tag{4.3}$$

from which we see that

$$\bar{R}^{-1} \geqq |R^{-1}| \geqq 0. \tag{4.4}$$

If we denote $\bar{R}^{-1} = (\bar{\delta}_{ij})$, then

$$\bar{\rho}_j = \sum_{i=1}^{j} \bar{\delta}_{ij} \geqq \sum_{i=1}^{j} |\delta_{ij}| = \rho_j. \tag{4.5}$$

Since $\bar{R}^{-1} \geqq 0$, the $\bar{\rho}_j$'s can be computed with a single back solve. Let $\bar{\boldsymbol{\rho}} = (\bar{\rho}_1, \cdots, \bar{\rho}_n)^T$; then, if $\mathbf{e} = \sum_{i=1}^{n} \mathbf{e}_i$, we have

$$\bar{R}^{-T}\mathbf{e} = \bar{\boldsymbol{\rho}} \tag{4.6}$$

and

$$\bar{\rho}_j = \frac{1}{|r_{jj}|}\left(1 + \sum_{i=1}^{j-1} |r_{ij}|\bar{\rho}_i\right). \tag{4.7}$$

The computation of $\bar{\rho}$ can be carried out during the QR decomposition and, in fact, $\bar{\rho}_k$ will be known after the kth step of the QR algorithm.

The bounds in Anderson and Karasalo [1] and Karasalo [9] are less expensive to compute, but also less accurate. There, $\bar{R} = \bar{D}(I - \bar{T})$ is replaced by a matrix, say $\tilde{R} = \bar{D}(I - \tilde{T})$, where the nonzero elements of each row of $\tilde{T} \geqq 0$ all have the same value, and $\tilde{R}^{-1} \geqq \bar{R}^{-1} \geqq |R^{-1}|$. Bounds similar to $\bar{\boldsymbol{\rho}}$ can be computed in $O(n)$ arithmetic, but they will, in general, be less accurate. In light of the fact that the QR decomposition requires $O(mn^2)$ arithmetic, the $O(n^2)$ arithmetic required to compute $\bar{\boldsymbol{\rho}}$ is not a great burden.

Now, consider (4.6) with R^{-T} in place of $\bar{R}^{-T}$. If we let

$$R^{-T}\begin{pmatrix} \pm 1 \\ \vdots \\ \pm 1 \end{pmatrix} = \begin{pmatrix} \tilde{\rho}_1 \\ \vdots \\ \tilde{\rho}_n \end{pmatrix}, \tag{4.8}$$

[2] A matrix $A = (a_{ij})$ is an H-matrix if the matrix $\bar{A} = (\bar{a}_{ij})$ with $a_{ii} = |a_{ii}|$, $\bar{a}_{ij} = -|a_{ij}|$ $i \neq j$ is an M-matrix.

[3] A matrix $A = (a_{ij})$ is an M-matrix if $a_{ij} \leqq 0$ for $i \neq j$, A is nonsingular, and $A^{-1} \geqq 0$.

where the signs are to be chosen later, then we have

$$|\tilde{\rho}_j| = \left| \sum_{i=1}^{j} \pm \delta_{ij} \right| \leqq \sum_{i=1}^{j} |\delta_{ij}| = \rho_j.$$

Since R^T is lower triangular, we may write

$$R^T \tilde{\boldsymbol{\rho}} = \begin{pmatrix} \pm 1 \\ \vdots \\ \pm 1 \end{pmatrix} \tag{4.9}$$

and choose the sign of the jth element of the right-hand side after $\tilde{\rho}_i$, $i < j$, have been computed. We may write

$$\tilde{\rho}_j = \frac{1}{r_{jj}} \left(\pm 1 - \sum_{i=1}^{j=1} r_{ij} \tilde{\rho}_i \right) \tag{4.10}$$

and choose the sign so that $|\tilde{\rho}_j|$ is as large as possible. Again, $\tilde{\rho}_k$ will be known after the kth step of the QR decomposition. We have then

$$|\tilde{\rho}_j| \leqq \rho_j \leqq \bar{\rho}_j, \qquad j = 1, \cdots, n. \tag{4.11}$$

The lower bounds $\tilde{\boldsymbol{\rho}}$ are computed in much the same manner as certain quantities in Cline, Moler, Stewart and Wilkinson ([3, p. 371]). There they are concerned with what corresponds to $\|\boldsymbol{\rho}\|_2$, while here we are concerned with each individual $|\tilde{\rho}_j|$.

Following their development, we can enhance the lower bounds by looking ahead before choosing the sign. Suppose $\tilde{\rho}_j$, $j < k$ have been chosen. Each $\tilde{\rho}_j$, $j \geqq k$ is determined by

$$r_{jj}\tilde{\rho}_j = (-r_{1j}\tilde{\rho}_1 - \cdots - r_{k-1j}\tilde{\rho}_{k-1}) + (-r_{kj}\tilde{\rho}_k - \cdots - r_{j-1j}\tilde{\rho}_{j-1} \pm 1), \tag{4.12}$$

where the right-hand side has been split into two parts. The first part is determined once $\tilde{\rho}_j$, $j < k$ have been chosen. Denote this part by $\tilde{\rho}_j^{(k-1)}$. The two values of $\tilde{\rho}_k$ are

$$\tilde{\rho}_k = \frac{1}{r_{kk}} (\tilde{\rho}_k^{(k-1)} \pm 1). \tag{4.13}$$

We denote these by $\tilde{\rho}_k^+$ and $\tilde{\rho}_k^-$. We know that both values satisfy

$$|\tilde{\rho}_k^+| \leqq \rho_k, \qquad |\tilde{\rho}_k^-| \leqq \rho_k, \tag{4.14}$$

and so may choose the larger in absolute value for our lower bound. However, we are still free to choose either $\tilde{\rho}_k^+$ or $\tilde{\rho}_k^-$ for computation of $\rho_j^{(k)}$, $j > k$ as long as we are consistent throughout. Let

$$\begin{aligned} \tilde{\rho}_j^{(k)+} &= \tilde{\rho}_j^{(k-1)} - r_{kj}\tilde{\rho}_k^+, \\ \tilde{\rho}_j^{(k)-} &= \tilde{\rho}_j^{(k-1)} - r_{kj}\tilde{\rho}_k^-, \end{aligned} \tag{4.15}$$

and choose the sign to maximize

$$\sum_{j>k} |\tilde{\rho}_j^{(k)}|. \tag{4.16}$$

There is about twice as much work in this algorithm, but the overall work to compute $\tilde{\boldsymbol{\rho}}$ remains $O(n^2)$. Notice that the vector of bounds $\tilde{\boldsymbol{\rho}}$ does not necessarily satisfy (4.9) but will satisfy (4.11).

4.2. Implicit scaling. As in § 2.3, these bounds can incorporate implicit scaling. If we again consider A_2^I as in (1.5), then the formulas corresponding to (4.7) and (4.10) become

$$\bar{\rho}_j = \frac{1}{|r_{jj}|}\left(\varepsilon_j + \sum_{i=1}^{j-1} |r_{ij}|\bar{\rho}_i\right), \tag{4.17}$$

$$\tilde{\rho}_j = \frac{1}{r_{jj}}\left(\pm\varepsilon_j - \sum_{i=1}^{j-1} r_{ij}\tilde{\rho}_i\right). \tag{4.18}$$

These bounds can be accumulated as a sequence of partial sums. Let

$$\bar{\rho}_j^{(0)} = \varepsilon_j,$$
$$\tilde{\rho}_j^{(0)} = 0, \qquad j = 1, \cdots, n,$$

and at step k update $\bar{\boldsymbol{\rho}}^{(k-1)}$, $\tilde{\boldsymbol{\rho}}^{(k-1)}$ by the formulas

$$\bar{\rho}_k^{(k)} = \frac{1}{|r_{kk}|}\bar{\rho}_k^{(k-1)}, \tag{4.19}$$

$$\tilde{\rho}_k^{(k)} = \frac{1}{r_{kk}}(\pm\varepsilon_k - \tilde{\rho}_k^{(k-1)}), \tag{4.20}$$

and for $j > k$

$$\bar{\rho}_j^{(k)} = \bar{\rho}_j^{(k-1)} + |r_{kj}|\bar{\rho}_k^{(k)}, \tag{4.21}$$

$$\tilde{\rho}_j^{(k)} = \tilde{\rho}_j^{(k-1)} + r_{kj}\tilde{\rho}_k^{k}. \tag{4.22}$$

4.3. Constructing a maximal set. These bounds may be used in the pivot strategy to construct a maximal set. Let $k_1 = 0$, $k_2 = n$ at the outset, and assume that after k steps we have $\bar{\rho}_j < 1$, $j = 1, \cdots, k$. We now choose column $k+1$ to make $\bar{\rho}_{k+1}$ as small as possible. First, we can use $\tilde{\boldsymbol{\rho}}$ to discard columns. Let $\mathbf{s}^{(k)}$ be the vector such that $s_j^{(k)}$ contains the $\|\cdot\|_2$ of the corresponding column of $R_{22}^{(k)}$. If column j were to be shifted to the $(k+1)$st position, then $\bar{\rho}_{k+1}$ and $|\tilde{\rho}_{k+1}|$ would have the values

$$\bar{\rho}_{k+1} = \frac{1}{s_j^{(k)}}\bar{\rho}_j^{(k)}, \tag{4.23}$$

$$|\tilde{\rho}_{k+1}| = \frac{1}{s_j^{(k)}}|\operatorname{sign}(\tilde{\rho}_j^{(k)})\varepsilon_j + \tilde{\rho}_j^{(k)}|, \tag{4.24}$$

where $\bar{\rho}_j^{(k)}$, $\tilde{\rho}_j^{(k)}$ are computed by (4.21) and (4.22). If, for some j, $k < j \leqq k_2$, the absolute value of the quantity in (4.24) is greater than or equal to 1.0, then the interval columns $\{\mathbf{a}_1^I, \cdots, \mathbf{a}_k^I, \mathbf{a}_j^I\}$ are dependent. Column j should be put in position k_2 and k_2 set to $k_2 - 1$.

After as many columns as possible are discarded, column $k+1$ is chosen from among the remaining columns to make $\bar{\rho}_{k+1}$ as small as possible. If $\bar{\rho}_{k+1} < 1$, we proceed. If not, then $|\tilde{\rho}_{k+1}| < 1 \leqq \bar{\rho}_{k+1}$. In order to insure that the first $k+1$ columns are not dependent, we must compute the exact value of ρ_{k+1}. This can be accomplished by a single back solve. If $\rho_{k+1} \geqq 1$, this column must be discarded. If $\rho_{k+1} < 1$, the column is accepted. Since ρ_{k+1} is a better bound than $\bar{\rho}_{k+1}$, we may set $\bar{\rho}_{k+1}^{(k+1)} = \rho_{k+1}$ for computation of $\bar{\rho}_j^{(k+1)}$ $j = k+2, \cdots, k_2$ as in (4.21). This does not affect (4.11).

In this manner, we can construct a set of columns to which no column may be added. Let k_1 be the number of columns in the largest subset for which $\sum \bar{\rho}_j^2 < 1$;

then, by Theorem 2.2 and (4.11) $k_1 \leqq \text{Rank}\,(A_2^I)$. If $k_1 = k_2$, we have a maximal set. If $k_1 < k_2$, then the set contains a maximal set which in turn contains the subset of columns used to compute k_1.

If A_2^I is assumed to be of full rank, that is, if it is not clearly rank deficient from the outset, then the bounds $\bar{\rho}_k$ will most likely suffice for all but at most a few columns. In some of these columns $\tilde{\rho}_k$ may provide enough information. The exact value ρ_k will be computed $O(1)$ times for all but the most pathological problems. This yields an $O(n^2)$ algorithm that produces a maximal set of independent columns of A_2^I.

5. Numerical results. In this section we will compare the performance of the three algorithms described above, the algorithm of § 3 that computes R^{-1} (we shall call it M1), the algorithm of § 4 that creates upper and lower bounds (M2) and the algorithm of § 4 that chooses the lower bound by looking ahead at partial sums (M3).

The ability of the three algorithms to determine numerical rank has been spelled out in the discussion above. However, several questions about their comparative performance must be answered by numerical tests.

Methods M2 and M3 will yield essentially the same pivot order, based upon the upper bounds $\bar{\boldsymbol{\rho}}$. The difference between M2 and M3 lies in the computation of the lower bounds $\tilde{\boldsymbol{\rho}}$. While M3 generally produces better bounds, the cost of computing those bounds is higher. The advantage to having better bounds lies in less frequent calculations of exact values of $\boldsymbol{\rho}$. While M3 did compute fewer exact values, the difference was not sufficient to warrant the extra arithmetic as will be shown below.

In M1, the pivot order is determined by $\boldsymbol{\rho}$, and so, it may differ from the pivot order produced by M2, M3. While each algorithm will find a maximal set of independent columns, the numerical rank produced by M1 may differ from that produced by M2, M3. Even if the rank is the same, the maximal sets may contain different columns.

Four tests were performed. In each trial an $m \times n$ matrix $A = (a_{ij})$ was constructed with $n \in [10, 50]$, $m \in [n, 2n]$, and $a_{ij} \in [-1, 1]$ (each uniformly distributed). In the first test 2,000 trials were run with relative uncertainty $\gamma_j = 10^{-4}$, $j = 1, \cdots, n$ and absolute uncertainty $\zeta_j = 10^{-4}$, $j = 1, \cdots, n$. In the second test, 500 matrices were generated. Each matrix was given the four values $\gamma_j = \zeta_j = 10^{-2}$, 10^{-4}, 10^{-6}, 10^{-8}, $j = 1, \cdots, n$ for a total of 2,000 trials. In the third test, after n and m were chosen, an integer k was chosen so that $n - k \leqq 5$ and A was constructed so that the last $n - k$ columns were linear combinations of the first k columns. Then a matrix $\Psi = (\psi_{ij})$ with $\psi_{ij} \in [-10^{-5}, 10^{-5}]$ was added to A. Each matrix was given the four values $\gamma_j = \zeta_j = 10^{-2}$, 10^{-4}, 10^{-6}, 10^{-8}, $j = 1, \cdots, n$ as in the second test. A total of 2,000 trials were run. In the fourth test A was chosen as in the first two tests, but relative and absolute uncertainties were chosen so that $\gamma_j = 10^{-\text{NEX}}$, $\text{NEX} \in [0, 14]$ and $\zeta_j \in [0, 1]$ (uniformly distributed).

The first test is perhaps the most realistic set of problems. These algorithms were designed to handle problems that are assumed to be of full rank, with proper detection of that rare instance when rank deficiency occurs. Tests 2, 3 and 4 were constructed to produce increasingly more trials which were rank deficient and rank deficiencies of higher orders. In test 4, the absolute uncertainty was of the same magnitude as the elements of A.

Table 5.1 shows how many of the trials were of full rank, in how many trials M1 produced the same rank as M2, M3, and in how many trials M1 produced the same maximal set as M2, M3. Also shown is the total number of exact values of $\boldsymbol{\rho}$ that M2 and M3 were required to compute. Figures 5.1–5.4 show the distribution of the upper and lower bounds, $\bar{\boldsymbol{\rho}}$ and $\tilde{\boldsymbol{\rho}}$, for those trials in which M1 produced the same column

TABLE 5.1

Test	Trials	Full Rank	Same Rank	Same Max Set	Exact Values M2	Exact Values M3
1	2,000	2,000	2,000	2,000	4	4
2	2,000	1,977	1,998	1,983	625	620
3	2,000	896	1,974	1,857	1,495	1,313
4	2,000	101	1,446	868	17,465	16,711

order as M2, M3. The left half of each figure shows the percent of the total with $|\tilde{\rho}_j|/\rho_j \leqq x$ and the right half shows the percent of the total with $\rho_j/\bar{\rho}_j \leqq x$. Here, x is the value indicated on the abscissa.

We may conclude that there is enough similarity between the performance of algorithms M1 and M2, M3, in terms of finding a maximal set, that the extra computation required by M1 may be unwarranted. Method M1 does provide exact values for $C_1(R)$ and $C_\infty(R)$, but Figs 5.1–5.4 show that M2 and M3 also provide good bounds for $C_1(R)$. These bounds are best on the trials that are rank deficient or nearly rank deficient. Method M1 does provide bounds on $\mathbf{x}^I$ which cannot be extracted from M2 or M3.

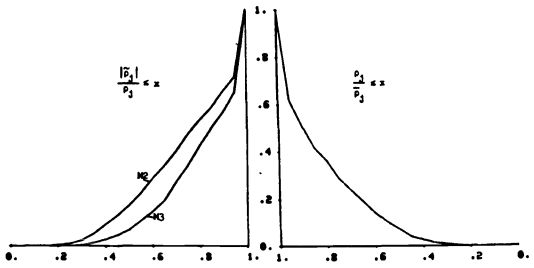


FIG. 5.1. *Test* 1.

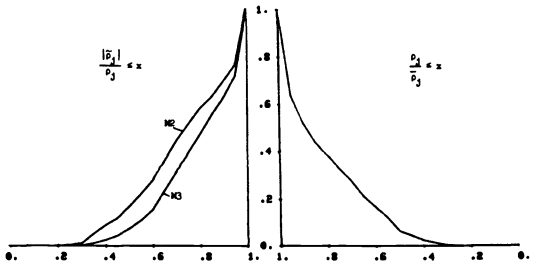


FIG. 5.2. *Test* 2.

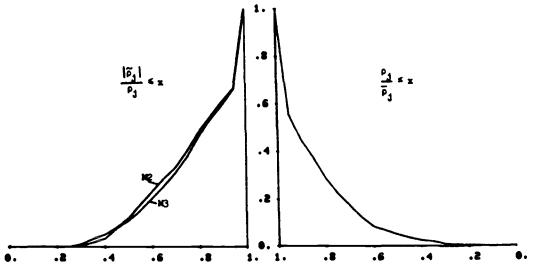


FIG. 5.3. *Test* 3.

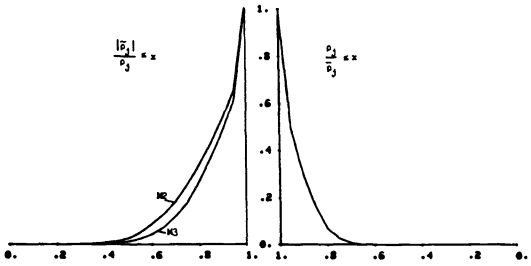


FIG. 5.4. *Test* 4.

The test results also show that M2 is more economical than M3. While Figs. 5.1–5.4 show that M3 yields better lower bounds, Table 5.1 shows that more arithmetic was required to achieve the same results in terms of finding a maximal set.

In every trial in the first three tests we had $k_1 = k_2$ so there was no doubt about the rank of A_2^I. In the fourth test there was uncertainty as to the rank in some trials but $k_2 - k_1$ never exceeded 3.

Acknowledgment. I would like to thank the referees for their careful reading and helpful suggestions and Ilkka Karasalo for pointing out several errors in the original manuscript.

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] P. BUSINGER AND G. GOLUB, *Linear least squares by Householder transformations*, Numer. Math. 7, (1965), pp. 269–276.

[3] 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.

[4] G. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965) pp. 205–224.

[5] G. GOLUB, *Numerical methods for solving linear least squares problems*, Numer. Math., 7 (1965), pp. 206–216.

[6] G. GOLUB, V. KLEMA AND G. W. STEWART, *Rank degeneracy and least squares problems*, Stanford Report Stan-CS-76-559, Stanford University, (1976).

[7] E. HANSON, *Interval arithmetic in matrix computations, Part* I, SIAM J. Numer. Anal., 2 (1965) pp. 308–320.

[8] E. HANSON AND R. SMITH, *Interval arithmetic in matrix computations, Part* II, SIAM J. Numer. Anal., 4 (1967), pp. 1–9.

[9] I. KARASALO, *A criterion for truncation of the QR decomposition algorithm for the singular least squares problem*, BIT 14, (1974), pp. 156–166.

[10] C. LAWSON AND R. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[11] T. A. MANTEUFFEL, *Numerical rank determination in linear least squares problems*, Sandia Laboratories Report SAND 79-8243, Albuquerque, NM, August 1979.

[12] ———, *The weighted linear least squares problem; an interval analysis approach*, Sandia Laboratories Report SAND 80-1260, Albuquerque, NM August, 1980.

[13] R. E. MOORE, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[14] M. J. D. POWELL AND J. K. REID, *On applying Householder transformations to linear least squares problems*, Proc. IFIP 1968, vol. 1, 1969, pp. 122–126.

[15] G. W. STEWART, *Introduction to Matrix Computation*, Academic Press, New York, 1973.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0008 $01.00/0

OPTIMAL SMOOTHING OF NOISY DATA USING SPLINE FUNCTIONS*

FLORENCIO UTRERAS D.†

Abstract. We consider the problem of approximating a function f supposed to be "smooth", given its values known with error at n different points of a real interval $[a, b]$. To approximate f we use the natural smoothing spline of order q and parameter τ. For choosing τ, the method of generalized cross validation, proposed by Wahba and others, has very interesting theoretical properties, but requires expensive calculations.

The asymptotic behavior of the eigenvalues associated with the spline functions provides a practical method for calculating the GCV function which reduces the computation time by a factor of n.

Key words. splines, smoothing, cross validation, noisy data.

1. Introduction. We consider the problem of approximating the unknown function $f:[a, b]\to\mathbb{R}$, supposed to be "smooth", from the values $f(t_i)$ measured with error at n distinct points of $[a, b]$. More precisely, let

$$z_i = f(t_i)+\varepsilon_i, \qquad i=1, 2, \cdots, n, \tag{1.1}$$

where ε_i are random errors satisfying

$$\begin{aligned} &\mathcal{E}[\varepsilon_i]=0,\\ &\mathcal{E}[\varepsilon_i\varepsilon_j]=0, \qquad i\neq j,\\ &\mathcal{E}[\varepsilon_i^2]=v_i^2, \end{aligned} \tag{1.2}$$

and $f\in H^q[a, b]$, defined by $H^q[a, b]=\{g:[a, b]\to\mathbb{R} \mid g, g', \cdots, g^{(q-1)}$ are absolutely continuous and $\int_a^b [g^{(q)}(t)]^2\,dt<+\infty\}$. For $n\geqq q$, our estimate for f is the polynomial smoothing spline $\sigma_{n,\tau}$ defined by

$$\begin{aligned} &\tau\int_a^b [\sigma_{n,\tau}^{(q)}(t)]^2\,dt+\frac{1}{n}\sum_{i=1}^n \alpha_{in}^2(z_i-\sigma_{n,\tau}(t_i))^2\\ &\quad=\min_{g\in H^q[a,b]}\left\{\tau\int_a^b [g^{(q)}(t)]^2\,dt+\frac{1}{n}\sum_{i=1}^n \alpha_{in}^2(z_i-g(t_i))^2\right\} \end{aligned} \tag{1.3}$$

where the weights α_{in}^2 are defined by

$$\alpha_{in}^2=\frac{\dfrac{1}{v_i^2}}{\dfrac{1}{n}\displaystyle\sum_{j=1}^n \frac{1}{v_j^2}}. \tag{1.4}$$

The α_{in} are completely determined if we know the v_i up to a multiplicative constant. Also, $\alpha_{in}=1$ for $i=1, 2, \cdots, n$ if $v_i=v$ for $i=1, 2, \cdots, n$.

Let us denote $Y_i^\tau=\sigma_{n,\tau}(t_i)$, $i=1, 2, \cdots, n$. It is well known that

$$z\to Y^\tau \tag{1.5}$$

is a linear one-to-one transformation from $\mathbb{R}^n$ into itself, for each $\tau>0$. We call the associated matrix $A_n(\tau)$.

---

* Received by the editors August 14, 1979, and in final revised form March 25, 1981.
† Departamento de Matemáticas, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile, Casilla 5272-Correo 3, Santiago, Chile.

The main subject of this paper is the practical choice of τ, the tradeoff between the "roughness" of the solution, measured by $\int_a^b [g^{(q)}(t)]^2\,dt$, and the infidelity to the data, measured by $(1/n)\sum_{i=1}^n \alpha_{in}^2 (z_i - g(t_i))^2$.

Wahba and others have proposed choosing τ as the minimizer of the (generalized cross validation) function:

$$V_n(\tau) = \frac{\frac{1}{n}\sum_{i=1}^{n} \alpha_{in}^2 (z_i - \sigma_{n,\tau}(t_i))^2}{\left(1 - \frac{1}{n}\sum_{i=1}^{n} \operatorname{Tr}(A_n(\tau))\right)^2}. \tag{1.6}$$

For theoretical properties of this estimator see [18], [5] and [16].

Wahba [18], [19], Craven & Wahba [5] and others have shown that the minimizer of (1.6) provides a very good estimate of the optimal parameter, even for relatively small samples, but their method for performing this calculation is very expensive.

The algorithms that we develop here allow us to treat large samples at a very reduced cost. For cubic or quintic splines the cost is $1/n$ times the cost of actual algorithms ([5], [20]) in the case of equally spaced data.

Early, Utreras [14] gave the theorem on the asymptotic behavior of the eigenvalues of Ω for the equally-spaced-data case. The present paper generalizes that result for unequally-spaced data and general standard deviations, and it also proposes a practical method for using this theorem in the general case and provides a comparison with other methods.

2. Mathematical background. Let $[a, b]$ be a real interval and $a < t_1 < t_2 < \cdots < t_n < b$ be n different points belonging to $[a, b]$. If $Y_1, Y_2, \cdots, Y_n$ are n real numbers ($n > 2$), it is well known that there exists one and only one solution to the problem

$$\underset{g \in H^q[a,b]}{\text{Minimize}} \left\{ \int_a^b [g^{(q)}(t)]^2\,dt \right\}, \qquad g(t_i) = Y_i, \quad i = 1, 2, \cdots, n. \tag{2.1}$$

The solution σ of this problem is called the interpolating spline of order q. We also know that σ has the following properties:

(i) σ is a polynomial of degree $\leqq q-1$ in the two intervals

$$[a, t_1], \qquad [t_n, b];$$

(ii) σ is a polynomial of degree $\leqq 2q-1$ in each interval $[t_i, t_{i+1}]$, $i = 1, 2, \cdots, n-1$;

(iii) $\sigma^{(j)}$ is continuous, $j = 0, 1, \cdots, 2q-2$.

Let S_n^q be the linear space of all real-valued functions defined on $[a, b]$ satisfying (i), (ii) and (iii). We define a basis on S_n^q in the following way. Let σ_i $i = 1, \cdots, n$ be the interpolating spline satisfying

$$\sigma_i(t_j) = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases} \tag{2.2}$$

Thus, if σ is the solution to the problem (2.1), then

$$\sigma = \sum_{i=1}^{n} Y_i \sigma_i. \tag{2.3}$$

Let s be an element of S_n^q whose values at $t_1, \cdots, t_n$ are given by

$$s(t_j) = x_j, \qquad j = 1, \cdots, n. \tag{2.4}$$

We then have

$$\begin{aligned} \int_a^b \sigma^{(q)}(t) s^{(q)}(t)\, dt &= \sum_{i=1}^n \sum_{j=1}^n x_i Y_j \int_a^b \sigma_i^{(q)}(t) \sigma_j^{(q)}(t)\, dt \\ &= \sum_{i=1}^n \sum_{j=1}^n x_i Y_j \omega_{ij}^q, \end{aligned} \tag{2.5}$$

with

$$\omega_{ij}^q = \int_a^b \sigma_i^{(q)}(t) \sigma_j^{(q)}(t)\, dt,$$

and we define Ω_n^q as the matrix whose elements are ω_{ij}^q. Thus we have

$$\int_a^b \sigma^{(q)}(t) s^{(q)}(t)\, dt = \langle x, \Omega_n^q Y \rangle, \tag{2.6}$$

where $\langle \cdot, \cdot \rangle$ denotes the scalar product in $\mathbb{R}^n$.

Finally, let $\tau > 0$ be a given real number and $z_1, z_2, \cdots, z_n$ be n real numbers. We define the natural polynomial smoothing spline as the solution of the problem

$$\underset{u \in H^q[a,b]}{\text{Minimize}} \left\{ \tau \int_a^b [u^{(q)}(t)]^2\, dt + \frac{1}{n} \sum_{i=1}^n \alpha_{in}^2 (u(t_i) - z_i)^2 \right\}.$$

This problem is equivalent to

$$\underset{Y \in \mathbb{R}^n}{\text{Minimize}} \left\{ \underset{\substack{u \in H^q[a,b] \\ u(t_i) = Y_i, i = 1, \cdots, n}}{\text{Minimize}} \left\{ \tau \int_a^b [u^{(q)}(t)]^2 + dt + \frac{1}{n} \sum_{i=1}^n \alpha_{in}^2 (u(t_i) - z_i)^2 \right\} \right\}$$

$$= \underset{Y \in \mathbb{R}^n}{\text{Minimize}} \left\{ \tau \langle Y, \Omega_n^q Y \rangle + \frac{1}{n} \sum_{i=1}^n \alpha_{in}^2 (Y_i - z_i)^2 \right\},$$

and the solution Y^τ satisfies

$$\tau \Omega_n^q Y^\tau + \frac{1}{n} D_n Y^\tau = \frac{1}{n} D_n z, \tag{2.7}$$

where D_n is the diagonal matrix with elements α_{in}^2, $i = 1, \cdots, n$. This equality allows us to write

$$\begin{aligned} Y^\tau &= (1 + n\tau D_n^{-1} \Omega_n^q)^{-1} z \\ &= A_n(\tau) z, \end{aligned} \tag{2.8}$$

where

$$A_n(\tau) = (1 + n\tau D_n^{-1} \Omega_n^q)^{-1}. \tag{2.9}$$

Let $\lambda_{1n}, \lambda_{2n}, \lambda_{3n}, \cdots, \lambda_{nn}$ be the n eigenvalues of $nD_n^{-1} \Omega_n^q$ in increasing order. Then

$$\operatorname{Tr}(A_n(\tau)) = \sum_{i=1}^n \frac{1}{1 + \tau \lambda_{in}}. \tag{2.10}$$

In the following section we study the behavior of the λ_{in} and get an asymptotic expression for (2.10).

3. The asymptotic behavior of the eigenvalues. Let us consider the following bilinear forms defined and continuous on $H^q[a, b]\times H^q[a, b]$:

$$\mathscr{B}(u, v)=\int_a^b u^{(q)}(t)v^{(q)}(t)\,dt, \tag{3.1}$$

$$\mathscr{A}(u, v)=\int_a^b \omega(t)u(t)v(t)\,dt, \tag{3.2}$$

where ω is a positive function such that there exist positive constants c_1, c_2 satisfying

$$0<c_1\leqq\omega(t)\leqq c_2 \quad \text{for all } t\in[a, b]. \tag{3.3}$$

Let $\mu_1, \mu_2, \mu_3, \cdots$ be the eigenvalues of the problem

$$\mathscr{B}(u, v)=\mu\mathscr{A}(u, v) \quad \text{for all } v\in H^q[a, b]. \tag{3.4}$$

Our aim in this section is to study the relationship between these eigenvalues and those associated with $nD_n^{-1}\Omega_n$. This relationship will be an asymptotic one, that is to say, it will hold for large n. Therefore in order to clarify the presentation, we introduce the superscript n on the abcissae $t_1, \cdots, t_n$; they will now be denoted by $t_1^n, t_2^n, \cdots, t_n^n$.

Now, we introduce a numerical quadrature formula which approximates $\mathscr{A}$, say $\mathscr{A}_n$:

$$\mathscr{A}_n(u, v)=\sum_{i=1}^{n} u(t_i^n)v(t_i^n)\omega(t_i^n)d_i^n, \tag{3.5}$$

where the weights $\{d_i^n\}_1^n$ are defined by

$$\begin{aligned} d_1^n &= t_1^n-a+\tfrac{1}{2}h_1^n, \\ d_i^n &= \frac{h_i^n+h_{i+1}^n}{2}, \qquad i=2, \cdots, n-1, \\ d_n^n &= b-t_n^n+\tfrac{1}{2}h_{n-1}^n \end{aligned} \tag{3.6}$$

and

$$h_i^n=t_{i+1}^n-t_i^n, \qquad i=1, 2, \cdots, n-1.$$

With this definition, it is straightforward to prove the following result.

LEMMA 1. *If $u, v\in H^q[a, b]$ $(q>2)$ and $\omega\in H^2[a, b]$, we have the following error bound*:

$$|\mathscr{A}(u, v)-\mathscr{A}_n(u, v)|\leqq\|u\|_q\cdot\|v\|_q\|\omega\|_2\cdot\delta_n^{-2}, \tag{3.7}$$

where

$$\delta_n=\max\{t_1^n-a, b-t_n^n, h_1^n, \cdots, h_{n-1}^n\}.$$

Proof. See Utreras [17].

Consider now the eigenvalue problem

$$\mathscr{B}(u_n, v)=\mu_n\mathscr{A}_n(u_n, v) \quad \text{for all } v\in H^q[a, b]. \tag{3.8}$$

In the following lemma, we prove that the eigenvalues of (3.8) are those of $nD_n^{-1}\Omega_n$ and conversely, we also state that the eigenfunctions of (3.8) are polynomial spline functions belonging to S_n^q.

LEMMA 2. *Let $\mu_{1n}, \mu_{2n}, \cdots, \mu_{nn}$ be the eigenvalues of* (3.8) *in increasing order, and suppose that there exists a function $\omega \in H^2[a, b]$ satisfying*

$$\omega(t_i^n) = \frac{1}{nd_i^n}\alpha_{in}^2, \qquad i = 1, 2, \cdots, n.$$

Then, we have

(i) *u_n is a solution of* (3.8) *implies that $u_n \in S_n^q$;*

(ii) *$\mu_{in} = \lambda_{in}$, $i = 1, 2, \cdots, n$.*

Proof. The eigenvalues of (3.8) are the solutions of

$$\mathcal{B}(u_n, v) = \mu_n \mathcal{A}_n(u_n, v) \quad \text{for all } v \in H^q[a, b].$$

Thus,

$$\int_a^b u_n^{(q)}(t)v^{(q)}(t)\,dt = \mu_n \sum_{i=1}^n u_n(t_i^n)v(t_i^n)\omega(t_i^n)\,d_i^n$$

$$= \sum_{i=1}^n [\mu_n u_n(t_i^n)\omega(t_i^n)d_i^n]v(t_i^n) \quad \text{for all } v \in H^q[a, b].$$

So we have

$$\int_a^b u_n^{(q)}(t)v^{(q)}(t)\,dt = \sum_{i=1}^n \xi_i^n v(t_i^n) \quad \text{for all } v \in H^q[a, b].$$

Then, using a well-known theorem on the characterization of spline functions (see [8, pp. 219–221]) we conclude that $u_n \in S_n^q$. Problem (3.8) is thus equivalent to

$$u_n \in S_n^q, \qquad \int_a^b u_n^{(q)}(t)v^{(q)}(t)\,dt$$

$$= \mu_n \sum_{i=1}^n u_n(t_i^n)v(t_i^n)\omega(t_i^n)d_i^n \quad \text{for all } v \in S_n^q.$$

Let $x_n \in \mathbb{R}^n$ be the vector of values of u_n at the knots, and $y \in \mathbb{R}^n$ the vector of values of v at the knots. So, using (2.6), we can write

$$\langle \Omega_n^q x_n, y\rangle = \mu_n \sum_{i=1}^n x_{ni}y_i\omega(t_i^n)d_i^n \quad \text{for all } y \in \mathbb{R}^n. \tag{3.9}$$

Now, from the hypothesis, this problem is equivalent to

$$\langle \Omega_n^q x_n, y\rangle = \mu_n \sum_{i=1}^n \frac{1}{n}\alpha_{in}^2 x_{ni}y_i$$

$$= \mu_n \frac{1}{n}\langle D_n x_n, y\rangle \quad \text{for all } y \in \mathbb{R}^n,$$

and so

$$\Omega_n^q x_n = \mu_n \frac{1}{n} D_n x_n, \qquad nD_n^{-1}\Omega_n^q x_n = \mu_n x_n. \tag{3.10}$$

Conversely, if x_n is an eigenvector of (3.10), we can define as u_n the spline function having x_n as the vector of values at the knots, and the result holds.

Now, we can state the main result of this section.

THEOREM 1. *Let λ_{in}, $i = 1, 2, \cdots, n$ be the eigenvalues associated with the matrix $nD_n^{-1}\Omega_n^q$, and suppose that there exists a positive function $\omega \in H^2[a, b]$ such that*

(i) $$\omega(t_i^n) = \frac{1}{nd_i^n}\alpha_{in}^2, \qquad i = 1, 2, \cdots, n;$$

(ii) *there exist $C_1, C_2 > 0$ satisfying $0 < C_1 \leqq \omega(t) \leqq C_2$;*

(iii) $\lim_{n\to\infty} \delta_n = 0$.

Then the λ_{in} converge to the eigenvalues of the problem (3.4). *That is to say,*

$$\lim_{n\to\infty} \lambda_{in} = \mu_i \quad \textit{for each fixed } i.$$

Proof. To show this result, we apply a lemma by Fix (see [6, Lemma 4.1]). We have that $\sup_{\|u\|=1} \inf_{\|v\|=1} |\mathscr{A}(u, v) - \mathscr{A}_n(u, v)| =: \eta_{\mathscr{A}}^n$ goes to zero as n increases because

$$\eta_{\mathscr{A}}^n = \sup_{\|u\|=1} \inf_{\|v\|=1} |\mathscr{A}(u, v) - \mathscr{A}_n(u, v)| \leqq \sup_{\substack{u,v \\ \|u\|=1 \\ \|v\|=1}} |\mathscr{A}(u, v) - \mathscr{A}_n(u, v)| \leqq \delta_n^2 \cdot \|\omega\|_2^2.$$

Also, we have that $\mathscr{B}$, $\mathscr{A}$, $\mathscr{A}_n$ are symmetric and positive semidefinite, so the eigenvalues are simple, and we can apply the theorem by Fix, which states that

$$\lim_{n\to\infty} |\lambda_{in} - \mu_i| = 0 \quad \text{for each fixed } i.$$

The following result tells us something about the eigenvalues $\{\mu_i\}_{i\in N}$.

COROLLARY 1. *The following two problems are equivalent.*

(V.P.) $$\mathscr{B}(u, v) = \mu\mathscr{A}(u, v) \quad \textit{for all } v \in H^q[a, b];$$

(D.P.) $$(-1)^q D^{2q} u = \mu\omega u,$$
$$u^{(j)}(a) = u^{(j)}(b) = 0, \qquad j = q, \cdots, 2q-1.$$

Proof. It suffices to integrate by parts.

Note. It is necessary to discuss the hypothesis

$$\omega(t_i^n) = \frac{1}{nd_i^n}\alpha_{in}^2, \qquad i = 1, 2, \cdots, n.$$

This function has a simple form in two important cases.

(1) When the data are equally spaced and the standard deviations are equal, ω becomes a constant.

(2) When we can choose the data points in such a way that the distance between consecutive points is proportional to the reciprocal of the variance of the errors, ω becomes a constant.

Generally, this hypothesis implies the existence of a relation between the choice of the data points and the variance of the errors. This relation must be invariant with n. We believe that it could be possible to show the result using a hypothesis like

$$0 < C_1 \leqq \omega_n(t_i^n) = \frac{1}{nd_i^n}\alpha_{in}^2 \leqq C_2,$$

with $\lim_{n\to\infty} \omega_n \to \omega$, the convergence being taken in $H^2[a, b]$.

Now we are interested in the behavior of Tr $(A_n(\tau))$ as n increases. We have

$$\frac{1}{n} D_n A_n(\tau) z + \tau \Omega_n^q A_n(\tau) z = \frac{1}{n} D_n z;$$

this equality is equivalent to

$$\frac{1}{n} \langle D_n A_n(\tau) z, x\rangle + \tau \langle \Omega_n^q A_n(\tau) z, x\rangle = \frac{1}{n} \langle D_n z, x\rangle \quad \text{for all } x \in \mathbb{R}^n. \tag{3.11}$$

If we denote by $r, s \in S_n^q$ the spline functions satisfying

$$\begin{aligned} s(t_i^n) &= x_i, \qquad i = 1, 2, \cdots, n, \\ r(t_i^n) &= z_i, \qquad i = 1, 2, \cdots, n, \end{aligned} \tag{3.12}$$

we have

$$\mathcal{A}_n(\sigma_{n,\tau}, s) + \tau \mathcal{B}(\sigma_{n,\tau}, s) = \mathcal{A}_n(r, s) \quad \text{for all } s \in S_n^q. \tag{3.13}$$

This equality leads us to study the operator $R_n(\tau)$ defined on $H^q[a, b]$, with values on $H^q[a, b]$ as

$$\mathcal{A}_n(R_n(\tau)g, u) + \tau \mathcal{B}(R_n(\tau)g, u) = \mathcal{A}_n(g, u) \quad \text{for all } u \in H^q[a, b]. \tag{3.14}$$

It is easy to see that $R_n(\tau)$ is a finite-range operator, and its trace is given by

$$\text{Tr } (R_n(\tau)) = \text{Tr } (A_n(\tau)). \tag{3.15}$$

Let us define $R(\tau)$ from $H^q[a, b]$ into $H^q[a, b]$ by the variational equality

$$\mathcal{A}(R(\tau)g, u) + \tau \mathcal{B}(R(\tau)g, u) = \mathcal{A}(g, u) \quad \text{for all } u \in H^q[a, b]. \tag{3.16}$$

The eigenvalues associated with $R(\tau)$ are $\{1/(1+\tau\mu_i)\}_{i\in N}$, and the μ_i satisfy the following inequality

$$\alpha(i)^{2q} \leqq \mu_{i+q} \leqq \beta(i)^{2q}, \qquad i = 1, 2, \cdots, \tag{3.17}$$

for some $\alpha, \beta > 0$. (For a proof of this result, see Utreras [17].) Then we have that

$$\sum_{i=1}^{\infty} \frac{1}{1+\tau\mu_i} < +\infty \quad \text{for each fixed } \tau > 0.$$

We conclude that the self-adjoint operator $R(\tau)$ belongs to the trace class, and

$$\text{Tr } (R(\tau)) = \sum_{i=1}^{\infty} \frac{1}{1+\tau\mu_i}. \tag{3.18}$$

The next theorem gives us a relationship between Tr $(R(\tau))$ and Tr $(A_n(\tau))$.

THEOREM 2. *Under the same hypotheses as in Theorem* 1, *and if* $\lim_{n\to\infty} n\delta_n^2 = 0$, *we have*

(i) $\lim_{n\to\infty} \text{Tr } (A_n(\tau)) = \text{Tr } (R(\tau))$ *for each fixed* $\tau > 0$;

(ii) $\lim_{n\to\infty} |\text{Tr } (A_n(\tau)) - \sum_{i=1}^{n} 1/(1+\tau\mu_i)| = 0$.

Proof. Let $\bar{R}_n(\tau)$ be the self-adjoint operator defined by the variational equality

$$\bar{R}_n(\tau)\phi \in S_n^q, \qquad \tau \mathcal{B}(\bar{R}_n(\tau)\phi, \sigma) + \mathcal{A}(\bar{R}_n(\tau)\phi, \sigma) = \mathcal{A}(\phi, \sigma) \quad \text{for all } \phi \in S_n^q,$$

and call $\{1/(1+\tau\xi_{in})\}_1^n$ the corresponding eigenvalues. Then we have

(a) $$\frac{1}{1+\tau\xi_{in}} \leqq \frac{1}{1+\tau\mu_i}, \qquad 1 \leqq i \leqq n,$$

(b) $$\|R(\tau) - \bar{R}_n(\tau)\| = O(\delta_n^2).$$

Inequality (a) comes from the monotonicity of the Rayleigh–Ritz–Galerkin approximation, and (b) comes from the error estimate (see [2], [3]). Let $\eta > 0$ be a given positive number. We have

$$|\mathrm{Tr}\,(\bar{R}_n(\tau)) - \mathrm{Tr}\,(R(\tau))| = \left| \sum_{i=1}^{n} \left(\frac{1}{1+\tau\xi_{in}} - \frac{1}{1+\tau\mu_i} \right) - \sum_{i=n+1}^{\infty} \frac{1}{1+\tau\mu_i} \right|.$$

Using the inequality $\mu_{i+q} \geqq \alpha i^{2q}$, we have

$$\sum_{i=J}^{\infty} \frac{1}{1+\tau\mu_i} = O(J^{-2q}),$$

so we can choose $J \in N$ such that $\sum_{i=J}^{\infty} 1/(1+\tau\mu_i) < \eta$.
Also, we can choose $N(J)$ such that for $n \geqq N(J)$ we have

$$\left| \sum_{i=1}^{J} \frac{1}{1+\tau\mu_i} - \sum_{i=1}^{J} \frac{1}{1+\tau\xi_{in}} \right| < \eta.$$

Then we obtain

$$\begin{aligned} |\mathrm{Tr}\,(R(\tau)) - \mathrm{Tr}\,(\bar{R}_n(\tau))| &\leqq \left| \sum_{i=1}^{J} \frac{1}{1+\tau\mu_i} - \sum_{i=1}^{J} \frac{1}{1+\tau\xi_{in}} \right| + \left| \sum_{i=J}^{\infty} \frac{1}{1+\tau\mu_i} + \sum_{i=J}^{n} \frac{1}{1+\tau\xi_{in}} \right| \\ &\leqq 3\eta. \end{aligned}$$

Also, we have

$$\begin{aligned} |\mathrm{Tr}\,(\bar{R}_n(\tau)) - \mathrm{Tr}\,(R_n(\tau))| &\leqq \sum_{i=1}^{n} |\langle s_i, (R_n(\tau) - \bar{R}_n(\tau)) s_i \rangle| \\ &\leqq n \|R_n(\tau) - \bar{R}_n(\tau)\| \\ &\leqq n O(\bar{\delta}_n^2). \end{aligned}$$

This last inequality is obtained using the theorem by Fix [6]. Thus we get the desired result.

This theorem allows us to replace $\mathrm{Tr}\,(A_n(\tau))$ by $\sum_{i=1}^{n} 1/(1+\tau\mu_i)$ in the computation of the generalized cross validation function. It is important to remember that we have only pointwise convergence. However, the evidence from numerical experiments is very convincing, and the gain in computing speed is enormous. In the following sections we describe some numerical experience. For a more complete set of tests and comparisons, see Utreras [17].

4. The case of equally-spaced data and constant standard deviation. In this section we particularize the preceding results when the t_i are defined by

$$t_i^n = a + (b-a)\frac{2i-1}{2n}, \qquad i = 1, 2, \cdots, n. \tag{4.1}$$

We assume too, that the errors are independent random variables satisfying

$$\mathscr{E}[\varepsilon_{in}^2] = v^2. \qquad i = 1, 2, \cdots, n. \tag{4.2}$$

Thus, we assume all variances to be equal to the unknown constant, v^2. In this case the discrete bilinear form defined by (3.5) becomes

$$\mathscr{A}_n(u, v) = \frac{b-a}{n} \sum_{i=1}^{n} u(t_i^n) v(t_i^n) \omega(t_i^n), \tag{4.3}$$

and the weight function becomes the constant $1/(b-a)$. So the differential problem is

$$(4.4)\qquad \begin{aligned} (-1)^q D^{2q}\phi &= \frac{1}{b-a}\,\phi, \\ \phi^{(j)}(a) = \phi^{(j)}(b) &= 0, \qquad j = q, \cdots, 2q-1. \end{aligned}$$

It is easy to see that $\mu_1 = \mu_2 = \cdots = \mu_q = 0$. Define $\lambda_i = \mu_{i+q}$ for $i \in N$. It is well known that the λ_i are the eigenvalues of the problem

$$(4.5)\qquad \begin{aligned} (-1)^q D^{2q}\phi &= \frac{1}{b-a}\,\phi, \\ \phi^{(j)}(a) = \phi^{(j)}(b) &= 0, \qquad j = 0, 1, \cdots, q-1 \end{aligned}$$

These eigenvalues are well known for $q = 1$, which corresponds to the harmonic oscillator or the vibrating string. For $q = 2$ we find the problem of the vibrating rod (see Courant and Hilbert [4]). We have tabulated these eigenvalues in this case and also in the case $q = 3$ (see Utreras [17]). Let us return to our initial problem, that is, to calculate the GCV function $V_n(\tau)$,

$$(4.6)\qquad V_n(\tau) = \frac{1}{n}\,\frac{\|z - A_n(\tau)z\|^2}{\left(1 - \frac{1}{n}\operatorname{Tr}(A_n(\tau))\right)^2}$$

To calculate the denominator we use the approximate expression given by Theorem 2, that is,

$$(4.7)\qquad \operatorname{Tr}(A_n(\tau)) \cong \sum_{i=1}^{n} \frac{1}{1+\tau\mu_i}.$$

To calculate the numerator, we use an algorithm by Paihua [10] which performs the computations in $30n$ operations.

Table 1 gives the run times (in seconds) on an IBM 360/67 computer. As is easily seen, time increases almost linearly with n.

TABLE 1

n	secs.	n	secs.
50	0.7993	250	3.1795
100	1.1376	350	3.7913
150	1.7564	350	4.4460
200	2.4703	450	4.8975

To illustrate the numerical behavior of the method, we performed the calculations on artificial data, $z_i = f(t_i) + \varepsilon_i$, $i = 1, 2, \cdots, n$. The ε_i's are pseudorandom normal numbers with standard deviation equal to 0.1. Because f is a known function, in this example we use $f(x) = \cos 4\pi x$. In the same graph, we plot the function (F), the spline calculated by GCV with the proposed algorithm (S) and the data (x). We have repeated the experiment for $n = 50,100,200,400$ to illustrate the remarkable "convergence" properties (which we cannot prove). See Figs. 1–4 for results.

The reader interested in computer programs for cubic or quintic splines is referred to [15]. Copies of this publication are available by request.

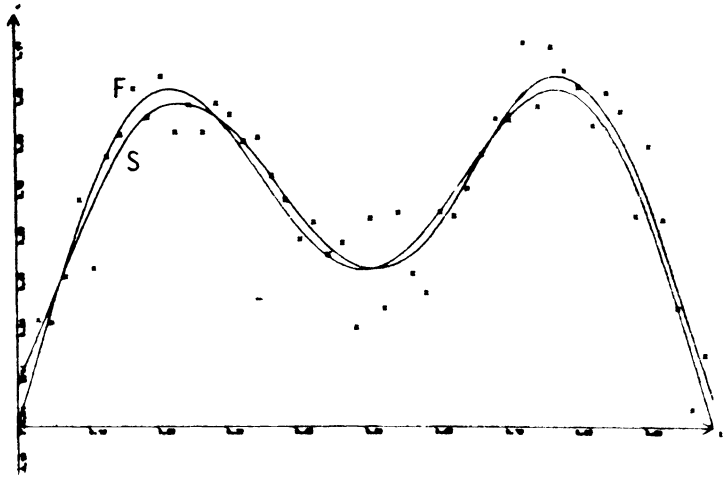


FIG. 1. 50 *points.*

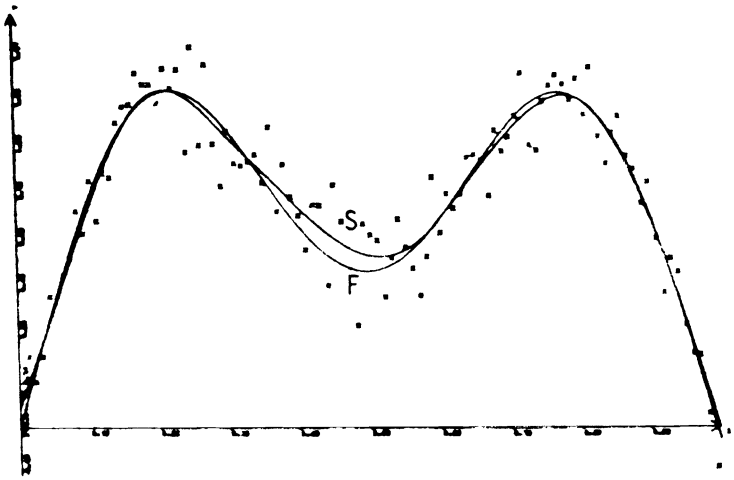


FIG. 2. 100 *points.*

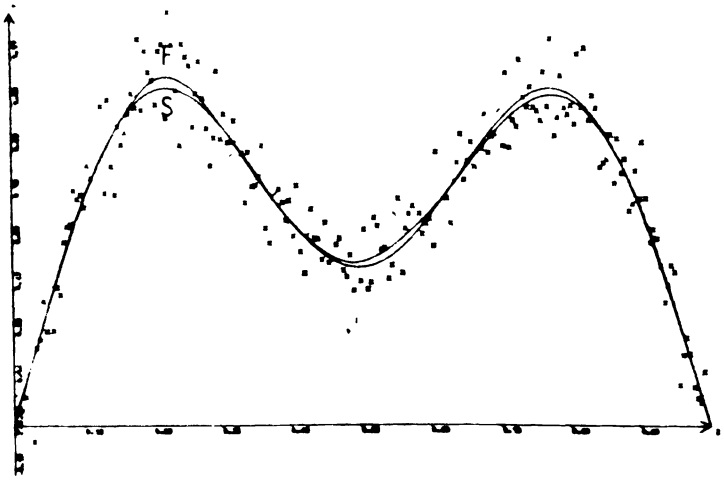


FIG. 3. 200 *points.*

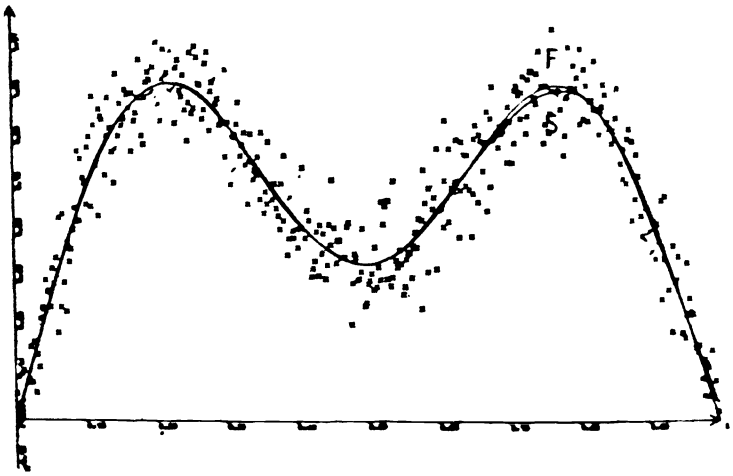


FIG. 4. 400 *points.*

5. A practical method in the general case. We return now to the general problem, that is, with general function ω. It is clear that we cannot calculate the eigenvalues in a simple way for general ω, so we propose a practical method based on a finite-difference approximation of the differential problem.

More explicitly, let $m > n$ and consider a finite-difference method with step $(b-a)/m$ to approximate the eigenvalues of the associated differential problem. The convergence of such methods has been studied by Kreiss [7]. If m is large, we can consider the first n eigenvalues of the finite-difference approximation as the exact ones and use them to calculate $\mathrm{Tr}\,(A_n(\tau))$. It might be possible to show that this approximation is convergent, but we do not establish this here.

With this method, we reduce our eigenvalue problem with a full symmetric matrix, to the following one:

$$\left(\frac{m+1}{b-a}\right)^{2q} Fx = \theta \bar{D} x, \qquad m \gg n, \tag{5.1}$$

where F is the discretization matrix and $\bar{D}$ is a diagonal matrix containing the values of ω at the discretization points.

We can transform this generalized eigenvalue problem to a simple eigenvalue problem without changing the band structure of the matrix. The new problem becomes

$$\left(\frac{m+1}{b-a}\right)^{2q} \bar{D}^{-1/2} F \bar{D}^{-1/2} v = \theta v, \qquad v \in \mathbb{R}^m. \tag{5.2}$$

The matrix $\bar{D}^{-1/2} F \bar{D}^{-1/2}$ is also symmetrical, so we can apply the Schwarz method (see [13]) to reduce it to tridiagonal form in $O(m^2)$ operations. In practical computations we have observed that, taking $m = 2n$, the approximations are correct (at least for our calculations).

In this way, we get an $O(n^2)$ method to compute the required eigenvalues. We have tested this method successfully with cubic splines. In this case, F becomes

$$F = \begin{bmatrix} -6 & 4 & -1 & & & \\ 4 & -6 & 4 & -1 & & \\ -1 & 4 & -6 & 4 & -1 & \\ & & & -1 & 4 & -6 \end{bmatrix}. \tag{5.3}$$

To calculate $\bar{D}$, we use the values of ω obtained by interpolation to the data

$$\omega(t_i^n) = \frac{1}{nd_i^n}\alpha_{in}^2. \tag{5.4}$$

The interpolation performed is piecewise linear.

To illustrate the numerical behavior of the method, we show two tests realized on artificial data. We use the test function $f(t) = \sin 2\pi t$ in the interval $[0, 1]$. We calculate the data points using the relation

$$t_i = \left[\int_0^{i/n} \omega(t)\,dt\right] \Big/ \left[\int_0^1 \omega(t)\,dt\right]$$

where $\omega(t) = 1 + \cos \pi t$. Then, we introduce a noise in the values of the function at $t_1, \cdots, t_n$ by adding pseudorandom normal numbers $N(0, (0{\cdot}1)^2)$. We do this for $n = 40,100$. The results are given in Figs. 5 and 6.

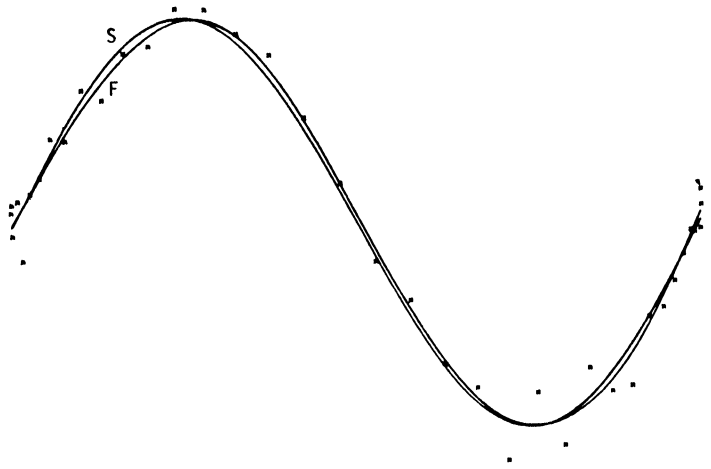


FIG. 5. 40 *points.*

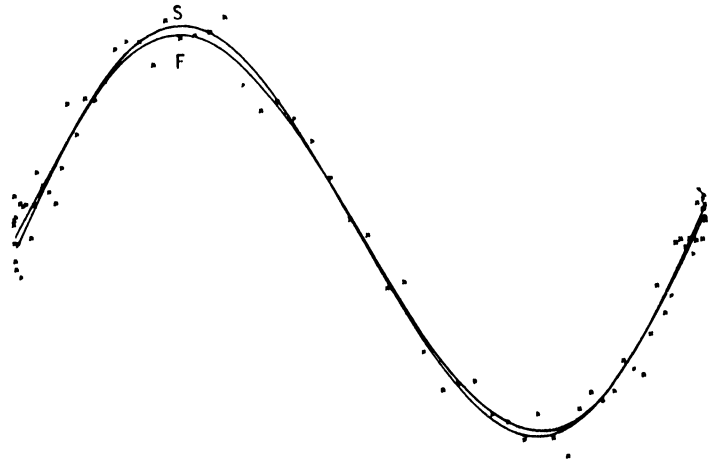


FIG. 6. 100 *points.*

6. A comparison with Reinsch's method. Our purpose in this section is to compare the method of GCV to Reinsch's suggestion [11] from the point of view of efficiency as defined by Wahba [19].

Define $T_n(\tau)$ as

$$T_n(\tau)=\frac{1}{n}\sum_{i=1}^{n}\alpha_{in}^2(f(t_i^n)-\sigma_{n,\tau}(t_i^n))^2. \tag{6.1}$$

$T_n(\tau)$ is a "mean" of squared differences between the approximation $\sigma_{n,\tau}$ and the true function f. As is easily seen, $T_n(\tau)$ cannot be computed for a real problem. In [16], we show that minimizing $V_n(\tau)$ is asymptotically the same as minimizing $T_n(\tau)$, and we say that the GCV method is asymptotically optimal. This demonstration is based on a conjecture by Craven and Wahba [5], and uses a bound on the eigenvalues proved in [17] which will be published elsewhere.

Define now the inefficiency of an approximation by smoothing spline as

$$I_n(\tau)=\frac{T_n(\tau)}{\min_\tau T_n(\tau)}. \tag{6.2}$$

This quantity is always greater than or equal to 1. The closer it is to one, the better is the approximation of f, at least from this error criterion point of view. So, to compare the two methods mentioned here, we use this parameter as the measure of goodness. A more complete comparison appears in [17].

We chose a large set of test functions defined on [0, 1] and performed the calculations as in the numerical examples of § 4. The computations have been repeated for various n and v. The results are given in Table 2. We observe that the GCV method is best, followed by Reinsch's method, but it is important to note that Reinsch requires the knowledge of v^2, which is not always known.

TABLE 2

			$v=0.100$					
Method	20	40	60	100	150	200	300	400
GCV	1.596	1.285	1.046	1.032	1.045	1.048	1.104	1.105
REINSCH	1.560	1.254	1.099	1.830	2.565	3.036	3.160	3.231
			$v=0.050$					
Method	20	40	60	100	150	200	300	400
GCV	1.130	1.025	1.027	1.029	1.061	1.140	1.071	1.027
REINSCH	1.133	1.283	2.701	2.402	1.544	1.299	1.220	1.061
			$v=0.010$					
Method	20	40	60	100	150	200	300	400
GCV	1.648	1.518	1.015	1.030	1.030	1.040	1.012	1.013
REINSCH	1.054	1.896	2.049	1.168	1.124	1.086	1.728	1.575
			$v=0.005$					
Method	20	40	60	100	150	200	300	400
GCV	1.320	1.258	1.296	1.013	1.019	1.026	1.016	1.042
REINSCH	1.726	1.801	1.216	1.447	1.350	1.261	1.294	1.455

Acknowledgments. This work was performed during the author's stay at the Institut de Recherche en Mathématiques Appliquées at Grenoble, and is a part of the author's doctoral thesis, which was directed by Professor P. J. Laurent, to whom the author wishes to express his sincere appreciation.

The author also wishes to thank Dr. David Hoaglin for his help in polishing the author's English.

REFERENCES

[1] M. ATTEIA, *Théorie et applications de fonctions spline en analyse numérique*, Doctoral Thesis, University of Grenoble, Grenoble, 1966.

[2] I. BABUSKA AND A. K. AZIZ, *Lecture on the finite element method*, in Foundations of the Finite Element Method with Applications to Partial Differential Equations. A. K. Aziz, ed., Academic Press, New York, 1972.

[3] F. CHATELIN, *Théorie de l'approximation des opérateurs linéaires. application au calcul des valeurs propres d'operateurs differentiels et integraux*, Cours de DEA d'Analyse Numérique, Université Scientifique et Médicale de Grenoble, 1976.

[4] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics*, I, Interscience, New York, 1953.

[5] P. CRAVEN AND G. WAHBA, *Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross validation*, Numer. Math., 31 (1979), pp. 317–403.

[6] G. FIX, *Effects of quadrature errors in finite element approximation of steady state, eigenvalue and parabolic problems*, in Foundations of the Finite Element Method with Applications to Partial Differential Equations, A. K. Aziz, ed., Academic Press, New York, 1972.

[7] H. O. KREISS, *Difference approximations for boundary and eigenvalue problems for ordinary differential equations*. Math. Comp., 26 (1972), pp. 605–624.

[8] P. J. LAURENT, *Approximation et optimisation*, Hermann, Paris, 1972.

[9] C. L. MALLOWS, *Some comments on* C_p. Technometrics, 15, (1973) pp. 661–675.

[10] L. PAIHUA, *Quelques méthodes numériques pour les fonctions spline a une et deux variables*, Doctoral thesis, University of Grenoble, 1978.

[11] C. M. REINSCH, *Smoothing by spline functions*, Numer. Math. 10, (1967) pp. 177–183.

[12] ———, *Smoothing by spline functions*, II. Numer. Math. 16, (1971), pp. 451–454.

[13] H. SCHWARZ, *Tridiagonalization of a symmetric band matrix*, in Linear Algebra, J. H. Wilkinson and C. Reinsch, eds., Springer-Verlag, Berlin, 1971.

[14] F. UTRERAS, *Sur le choix du paramètre d'ajustement dans le lissage par fonctions spline*, Exposé au Colloque National d'Analyse Numérique de France, Giens, May 21, 1978; Numer. Math., 34, (1980) pp. 15–28.

[15] ———, *A package for smoothing noisy data with splines*. Technical Report MA-80-B-209, Department of Mathematics, University of Chile, Santiago, 1980.

[16] ———, *Quelques résultats d'optimalité pour la méthode de validation croissé*, Séminaire d'Analyse Numérique, 301, Grenoble November, 1978.

[17] ———, *Utilisation de la méthode de validation croisée pour le lissage par functions spline à une ou deux variables*. Doctoral thesis, Université Scientifique et Médicale de Grenoble, Grenoble, May, 1979.

[18] G. WAHBA, *Practical approximate solutions to linear operator equations when the data are noisy*. SIAM J. Numer. Anal., 14 (1977), pp. 651–667.

[19] ———, *Smoothing noisy data with spline functions*. Numer. Math. 24, pp. 383–393 (1975).

[20] G. WAHBA AND S. WOLD, *A completely automatic French curve: fitting spline functions by cross-validation*. Comm. Statist., 4 (1975), pp. 1–7.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 3, September 1981

0196-5204/81/0203-0009 $01.00/0

ANALYSIS OF MEASUREMENTS BASED ON THE SINGULAR VALUE DECOMPOSITION*

RICHARD J. HANSON† AND MICHAEL J. NORRIS‡

Abstract. The problem of maintaining quality control of manufactured parts is considered. This involves matching points on the parts with corresponding points on a drawing. The difficulty in this process is that the measurements are often in different coordinate systems. Using the assumption that the relation between the two sets of coordinates is a certain rigid transformation, an explicit least squares solution is obtained. This solution requires the singular value decomposition of a related matrix.

Other topics in the paper include an appropriate angular representation of the resulting orthogonal transformation matrix, and a computational algorithm for the various required quantities.

Key words. orthogonal matrices, singular value decomposition, orthogonal Procrustes problem, analysis of measurements, quality control, part matching, least squares

1. Introduction.

The following problem arises in the analysis of measurements made on manufactured parts for the purpose of quality control.

A part is to be machined in accordance with specifications on drawings. To check conformance of this part with the drawings, a set of distinguished points is taken from the drawings and a corresponding set of points is taken from the part. If discrepancies between the two sets of coordinates are all within tolerance, the part is acceptable. Otherwise the part is deemed unacceptable and is discarded.

Discrepancies between the two sets of corresponding points can come from inaccuracies in manufacturing the part, or inaccuracy in measurements with respect to coordinate systems for the part or the drawing. Furthermore, it may be necessary to measure the set of points on the part in a different coordinate system than the points on the drawing. This introduces (possibly) large discrepancies between the points on the part and the points on the drawing.

We will not discuss in detail the problem of where or how the distinguished set of measurements should be taken on both the drawing and the part. This is a difficult matter that involves judgment from the individual responsible for quality control of the manufacturing process.

To salvage acceptable parts, one must account for the different coordinate system, in the two sets of measurements. It seems plausible that (to within error whose effect is negligible relative to tolerances) the coordinate axes of the set of drawings are orthogonal. Also, errors in any measurements made on the drawings are similarly negligible. One also expects that the measuring device, to within acceptable errors, yields measurements relative to a set of orthogonal coordinate axes. However, in addition to any intended change of coordinate system, the part undergoing the test may be improperly positioned relative to the coordinate system of the measuring device. This could result from some designated point on the part not being set at its prescribed location or from the entire part being misoriented relative to the coordinate axes. To accommodate these possibilities the relationship between the coordinate

* Received by the editors August 18, 1980. This work was sponsored by the U.S. Department of Energy under contract DE-AC04-76DP00789.

† Numerical Mathematics Division, Sandia National Laboratories, Albuquerque, New Mexico 87185 (a U.S. Department of Energy facility).

‡ Applied Mathematics Department, Sandia National Laboratories, Albuquerque, New Mexico 87185 (a U.S. Department of Energy facility).

systems should have the general form of a rigid transformation

$$\text{drawing points} = \text{orthogonal matrix} \times \text{part points} + \text{fixed translation}.$$

(This form of transformation also seems consistent with what one might do to fit two parts together without forcing. In fact, one could consider one of the parts as the "drawing." Then the transformation can be used to mate the two parts.) It is possible to minimize this corresponding discrepancy between drawing points and transforms of part points in the least squares sense. One can then base the accept or reject tests on the residuals of these equations. Perhaps the most natural choice for an approximation criterion would be to minimize the maximum residual. No direct solution to this "min-max" problem, with the simplicity of the solution for the least squares criterion, is known to the authors.

Using least squares, it is possible to give a simple algorithm for computing the (unknown) orthogonal matrix and the fixed translation of this transformation. The expression for this orthogonal matrix involves quantities that derive from the singular value decomposition (SVD) [1, pp. 238–239] of a related matrix. This is fully developed in § 2.

Section 3 outlines the computations to be performed and § 4 discusses the numerical analysis of the process and describes a computer program. Section 5 presents an appropriate angular representation for the transformation matrix. Although the original problem was in two or three space dimensions (and it is difficult to contemplate the problem in higher dimensions) the treatment in §§ 2 and 3 is applicable to any finite-dimensional space.

2. Analysis. In the following development the vectors are real, n-dimensional, and are also considered as $n \times 1$ matrices. For our derivation of the optimal transformation, we use the functional

$$(M, N) \equiv \text{trace } (M^T N).$$

For $M = N$, this dot (or inner) product defines the Frobenius or Euclidean norm in the linear vector space of $\hat{m} \times \hat{n}$ real matrices. (Here M^T = transpose of M, and trace $(M^T N)$ = sum of the diagonal terms of $M^T N$.)

The dot product has a number of easily proved properties stated as (1)–(5). Those properties allow for various formal interchanges in terms involving dot products of matrices. These rules are used freely within this section.

$$(M, N) = (N, M), \tag{1}$$

$$(M, N) = (N^T, M^T), \tag{2}$$

$$(MN, PQ) = (NQ^T, M^T P). \tag{3}$$

In (3), the indicated products must be defined in order for this to make sense. Also,

$$(M, N + P) = (M, N) + (M, P), \tag{4}$$

$$(M, \alpha N) = \alpha (M, N) = (\alpha M, N) \tag{5}$$

for scalar values of α.

The set of drawing points and the set of part points are respectively denoted by $\{Y_i \mid 1 \leqq i \leqq K\}$ and $\{X_i \mid 1 \leqq i \leqq K\}$. Since the coordinate systems for the drawing points

Y and part points X are related by a rigid transformation we have

$$Y = AX + B. \tag{6}$$

In (6), the $n \times n$ matrix A is orthogonal, $A^T A = I$, while B is an n-vector.

Other authors have considered problems and used methods that are closely related to ours. For example, Green [5], Bar-Itzhack [6] and especially Schönemann [7] have solved the problem of minimizing $(A\tilde{X} - \tilde{Y}, A\tilde{X} - \tilde{Y})$ where $\tilde{X}$ and $\tilde{Y}$ are $N \times K$ real matrices and A is an $N \times N$ orthogonal matrix. Here we solve the problem of minimizing $(A\tilde{X} + B\mathbf{e}^T - \tilde{Y}, A\tilde{X} + B\mathbf{e}^T - \tilde{Y})$ where $\tilde{X}$, $\tilde{Y}$ and A are as stated above, B is an N-vector, and determinant of $A = +1$. (The vector $\mathbf{e} = (1, \cdots, 1)^T$.)

The use of the singular value decomposition of square matrices, as utilized in this paper, simplifies derivations of the formulas for A and the required computations. The methods we present also apply to the above-mentioned related problems where A is only required to be orthogonal.

To obtain the best fit in the least squares sense, we want to minimize

$$f(A, B) = \sum_{i=1}^{K} (B - (Y_i - AX_i), B - (Y_i - AX_i)).$$

This expression can be simplified by eliminating B as an unknown. To do this we use Lemma 1.

LEMMA 1. *If $\{R_i \mid 1 \leqq i \leqq K\}$ is a set of n-vectors, then $\phi(R) = \sum_{i=1}^{K} (R - R_i, R - R_i)$ is minimized if and only if $R = \bar{R} = (\sum_{i=1}^{K} R_i)/K$.*

This lemma is easily proved by noting that, for all R, $\phi(R) = \sum_{i=1}^{K} [(R - \bar{R}, R - \bar{R}) + (R_i - \bar{R}, R_i - \bar{R})]$.

For any specific choice of A, including the optimal value of A which we do not know yet, Lemma 1 shows that $f(A, B)$ is minimized with

$$B = \bar{B} = \bar{Y} - A\bar{X}, \tag{7}$$

where

$$\bar{Y} = \left(\sum_{i=1}^{K} Y_i \right) \Big/ K$$

and

$$\bar{X} = \left(\sum_{i=1}^{K} X_i \right) \Big/ K.$$

Substituting this expression for $B = \bar{B}$ into $f(A, B)$ and defining the two sets of points (each with mean zero)

$$x_i = X_i - \bar{X}, \qquad y_i = Y_i - \bar{Y}, \qquad i = 1, \cdots, K \tag{8}$$

leads to the expression

$$f(A, \bar{B}) = \sum_{i=1}^{K} (Ax_i - y_i, Ax_i - y_i).$$

We want to choose an orthogonal $n \times n$ matrix A (possibly with the additional restriction $\det(A) = 1$), to minimize $f(A, \bar{B})$. Using (1), (4) and (5),

$$f(A, \bar{B}) = \sum_{i=1}^{K} [(Ax_i, Ax_i) - 2(y_i, Ax_i) + (y_i, y_i)].$$

An application of (3)–(5), together with the fact that A is an orthogonal matrix, shows that

$$f(A, \bar{B}) = -2 \sum_{i=1}^{K} (y_i, Ax_i) + \sum_{i=1}^{K} [(x_i, x_i) + (y_i, y_i)].$$

Finding A to minimize $f(A, \bar{B})$ requires finding an orthogonal matrix that maximizes the related functional

$$g(A) = \sum_{i=1}^{K} (y_i, Ax_i).$$

Again, (3) and (4) show that, with $C = \sum_{i=1}^{K} y_i x_i^T$,

$$g(A) = \sum_{i=1}^{K} (y_i x_i^T, A) = \left(\sum_{i=1}^{K} y_i x_i^T, A \right) \equiv (C, A). \tag{9}$$

It is worthwhile to point out that the inclusion of *weights* in the definition of $f(A, B)$ adds no essential complication or change to the development given here. In fact, if

$$f(A, B) = \sum_{i=1}^{K} \gamma_i^2 (B - (Y_i - AX_i), B - (Y_i - AX_i)), \qquad \gamma_i \neq 0,$$

then an obvious modification of Lemma 1 shows that

$$B = \bar{B} = \bar{Y} - A\bar{X},$$

$$\bar{Y} = \sum_{i=1}^{K} \gamma_i^2 Y_i \Big/ \sum_{i=1}^{K} \gamma_i^2,$$

and

$$\bar{X} = \sum_{i=1}^{K} \gamma_i^2 X_i \Big/ \sum_{i=1}^{K} \gamma_i^2.$$

Equation (8) is still valid, and minimizing

$$f(A, \bar{B}) = \sum_{i=1}^{K} \gamma_i^2 (Ax_i - y_i, Ax_i - y_i)$$

is equivalent to maximizing

$$g(A) = \left(\sum_{i=1}^{K} \gamma_i^2 y_i x_i^T, A \right) = (C, A).$$

Our point here is that the use of weights merely changes the computation of $\bar{X}$, $\bar{Y}$ and C in a trivial way. The arguments that follow remain unchanged.

To aid in the choice of A that maximizes $g(A)$, we introduce the singular value decomposition, SVD, of the $n \times n$ matrix C,

$$C = USV^T. \tag{10}$$

In (10) the $n \times n$ matrices U and V are orthogonal. Further, the $n \times n$ diagonal matrix $S = \text{diag}(s_1, \cdots, s_n)$ has the singular values of C as its diagonal terms with $s_1 \geqq s_2 \geqq \cdots \geqq s_n \geqq 0$. Substituting the SVD of C from (10) into $g(A)$ of (9) results in

$$g(A) = (USV^T, A) = (S, U^T A V) = (S, W),$$

where $W = U^T A V = \{w_{ij}\}$ is an $n \times n$ orthogonal matrix.

We now consider two cases for the optimal choice of A. In the first case A may be any orthogonal matrix, whereas in the second case A must also have determinant equal to the value 1.

Case I. *Determinant of A has either sign.* Since W is an $n \times n$ orthogonal matrix, each diagonal term satisfies $|w_{ii}| \leqq 1$, and thus

$$(S, W) = \sum_{i=1}^{n} s_i w_{ii} \leqq \sum_{i=1}^{n} s_i.$$

This inequality becomes an equality if and only if $w_{ii} = 1$ for each $s_i > 0$. If r is the number of nonzero singular values, then $g(A) = (S, W)$ is maximized precisely when

$$W = \begin{bmatrix} I & 0 \\ 0 & Z \end{bmatrix}.$$

Here Z is an arbitrary $(n-r) \times (n-r)$ orthogonal matrix. In case $r = n$ the matrix Z is absent, and then $W = I$. One is always free to choose $W = I$ for all values of r. This is effectively the results obtained by Schönemann [7].

Case II. *Determinant of* $A = +1$. The required condition is equivalent to $\det(W) = \det(U^T V)$. If either $\det(U^T V) = 1$ or $s_n = 0$, then the maximum of (S, W) is $\sum_{i=1}^{n} s_i$. An optimal choice of W is the same as in Case I with Z essentially arbitrary except for the requirement $\det(Z) = \det(U^T V)$. We will need the following lemma for the remaining development.

LEMMA 2. *Suppose that W is an* $n \times n$ *real orthogonal matrix with* $\det(W) = -1$. *Then*

(a) $\operatorname{trace}(W) = \operatorname{tr}(W) \leqq n - 2$;

(b) $\operatorname{tr}(W) = n - 2$ *if and only if*

$$W = P \operatorname{diag}(1, \cdots, 1, -1) P^T$$

for some $n \times n$ *orthogonal matrix P.*

Proof. Partition the eigenvalues of W into three groups

(i) $\lambda_1, \bar{\lambda}_1, \cdots, \lambda_p, \bar{\lambda}_p$ ($2p$ complex roots; $|\lambda_i|^2 = 1$),

(ii) $1, \cdots, 1$ (q roots, real and positive),

(iii) $-1, \cdots, -1$ (r roots, real and negative).

Since $\det(W) = |\lambda_1|^2 \cdots |\lambda_p|^2 (-1)^r = (-1)^r = -1$, we see that r is an odd integer with $r \geqq 1$.

Using Theorem 3.9 of [2, p. 285], we note that there is a real orthogonal matrix P such that $P^T W P$ is the direct sum of p individual 2×2 matrices

$$\begin{pmatrix} \cos(\theta_i) & \sin(\theta_i) \\ -\sin(\theta_i) & \cos(\theta_i) \end{pmatrix}$$

and the scalar matrices I_q followed by $-I_r$. Each angle θ_i is a principal argument of the complex eigenvalue λ_i of W, $1 \leqq i \leqq p$. Now computing $\operatorname{tr}(W) = \operatorname{tr}(PP^T W) = \operatorname{tr}(P^T W P)$ shows that

$$\operatorname{tr}(W) = 2 \sum_{i=1}^{p} \cos(\theta_i) + q - r.$$

The facts that $r \geqq 1$, $\cos(\theta_i) \leqq 1$, and $2p + q = n - r$ then show that

$$\operatorname{tr}(W) \leqq n - 2r \leqq n - 2.$$

This completes the proof of part (a) of the conclusion. The "if" clause of part (b) is

obviously true. To prove the "only if" clause of part (b), suppose that $\text{tr}(W) = n-2$. This can occur only if $r = 1$ and

$$2\sum_{i=1}^{p} \cos(\theta_i) + q = n-1.$$

The last of these equations says that $p = 0$ and $q = n-1$, so

$$P^T W P = \text{diag}(1, \cdots, 1, -1)$$

and the condition on W follows. This completes the proof of part (b) of the lemma. □

Suppose then that $\det(U^T V) = -1$ and $s_n > 0$. In this case we show that the maximum of (S, W) is $\sum_{i=1}^{n-1} s_i - s_n$ and that W is essentially the matrix $\text{diag}(1, \cdots, 1, -1)$.

To this end we write the inequalities

$$\begin{aligned}\sum_{i=1}^{n-1} s_i - s_n - (S, W) &= \sum_{i=1}^{n-1} (1-w_{ii})s_i - (1+w_{nn})s_n \\ &\geqq \sum_{i=1}^{n-1} (1-w_{ii})s_n - (1+w_{nn})s_n \\ &= [(n-2) - \text{tr}(W)]s_n \\ &\geqq 0.\end{aligned}$$

The first inequality here follows from the elementary fact that no entry of W exceeds one, and that the nonnegative singular values, $\{s_i\}$, are ordered. The second inequality uses Lemma 2, part (a).

Suppose now that equality holds throughout these relations. If j is the smallest index such that $s_j = s_n$, then equality holds throughout if and only if $w_{ii} = 1$ for $i < j$ and $\text{tr}(W) = n-2$. Thus W maximizes (S, W) if and only if $W = \text{diag}(I_{j-1}, W_{n-j+1})$ where W_{n-j+1} is orthogonal, $\det(W_{n-j+1}) = -1$, and $\text{tr}(W_{n-j+1}) = (n-j+1)-2 = n-j-1$. By Lemma 2, part (b), W maximizes (S, W) if and only if

$$W = \begin{bmatrix} I_{j-1} & 0 \\ 0 & P\,\text{diag}(1, \cdots, 1, -1)P^T \end{bmatrix}$$

for some orthogonal matrix P.

A particular choice of W that always results in the maximum of $g(A)$ for Case II is

$$W = \text{diag}(1, \cdots, 1, \det(U^T V)).$$

With this particular choice we can evaluate the orthogonal matrix

$$A = UWV^T$$

that maximizes $g(A)$ or, equivalently, minimizes $f(A, \bar{B})$.

The phenomenon of "ill-conditioning" is present in the computation of A in a subtle form. The dimensions of the arbitrary matrices Z and P of Cases I and II are related to the multiplicity of the smallest singular value. If s_n is the only zero singular value then the extra condition $\det(A) = 1$ fixes the previously arbitrary scalar orthogonal matrix Z. Multiple zero singular values may occur when a poor choice of coordinate pairs are chosen on the part and drawing. This in turn may lead to discontinuous dependence of the matrix A on the data in the sense that small changes in the data yield large (but bounded) changes in the elements of A.

A similar type of discontinuity could occur in Case II when $s_n > 0$ is a multiple singular value. These remarks should be considered with the disclaimer in the Introduction regarding the choice of a distinguished set of points. First, the pair $\{A, B\}$ may reduce the size of the residuals and hence be of value even when a large variation in A may result from small variations in the data. Next, it is possible to evaluate the choice of distinguished points under certain circumstances that we now describe.

Suppose the part is expected to match the drawing rather closely so that x_k is close to y_k. One can take the matrix G based only on drawing points

$$G = \sum_{k=1}^{K} y_k y_k^T,$$

and obtain the singular values (= eigenvalues) of G. Let δ be the minimum distance between two singular values of G. Now $C = G + H$ with $H = \sum_{k=1}^{K} (x_k - y_k) y_k^T$. If $\|H\| < \delta/2$ then the singular values of C differ from those of G by less than $\delta/2$ and also are distinct, [1, *p.* 25]. Here $\|\cdot\|$ = largest singular value.)

Should the choice of a distinguished set be a poor one on the basis of the suggested evaluation, it might be possible to change to a completely different distinguished set or to significantly modify the matrix C by adding points to the distinguished set. Such possibilities depend of course on the nature of the part and its intended usage.

Frequently a drawing for a part is in a different scale than the part itself. This means that the set of part points $\{X_i\}$ may be derived from a set of actual measurements on the part $\{\hat{X}_i\}$, by a scale factor, $X_i = \alpha \hat{X}_i$, $i = 1, \cdots, K$.

To provide a check for consistency of these measurements, one can slightly extend the type of transformation allowed, namely, a rigid transformation combined with a change of scale, or

$$Y = \alpha AX + B.$$

Here $\alpha \geqq 0$ is a scale factor to be determined, A is orthogonal, and B is an n-vector. The optimal value of α does not affect the optimal choice for A. We will not prove this here, but one chooses A, as outlined above but using the data $\{\hat{X}_i\}$ instead of $\{X_i\}$.

The optimal choice for α can be shown to be based on the quantity $\tilde{g} = g(A) = \sum_{i=1}^{K} (Ax_i, y_i)$ as follows:

$$\alpha = \begin{cases} \tilde{g} \Big/ \sum_{i=1}^{K} (x_i, x_i), & \tilde{g} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Now, for example, one can examine the size of both sets of residuals, $1 \leqq i \leqq K$,

$$Y_i - AX_i - B, \qquad B = \bar{Y} - A\bar{X},$$

and

$$Y_i - \alpha AX_i - \hat{B}, \qquad \hat{B} = \bar{Y} - \alpha A\bar{\hat{X}}.$$

This consistency check on the scaling of measurements can catch the following type of blunder: A part is manufactured half as large as the drawing scale when it should have been *twice* as large as the drawing scale.

Incidentally, it is not necessarily true that the residuals using α will be smaller than the residuals with $\alpha = 1$. What is true, of course, is that the sums of squares of residuals is not larger with the optimal α than with $\alpha = 1$.

Monitoring of the value of α can provide information of potential use for process control. If α is usually significantly above (below) unity, then there is probably a bias in the process. If in addition, the residuals using the optimum value of α are significantly smaller than those for $\alpha = 1$, then there is possibly a removable source of error in the process.

3. Computational steps. The given data are the set $\{Y_i\}$ of drawing points and the set $\{X_i\}$ of part points. An algorithm is outlined for determining the orthogonal matrix A, the n-vector B, and the optimum scale factor, α.

Step *Operation*

1. Compute the means
$$\bar{Y} = \left(\sum_{i=1}^{K} Y_i\right)\Big/K, \qquad \bar{X} = \left(\sum_{i=1}^{K} X_i\right)\Big/K.$$
2. Compute the differences
$$y_i = Y_i - \bar{Y}, \quad x_i = X_i - \bar{X}, \quad i = 1, \cdots, K.$$
3. Compute the matrix
$$C = \sum_{i=1}^{K} x_i y_i^T.$$
4. Compute the SVD of C,
$$C = USV^T.$$
5. Compute the product U^TV.
6. Compute the value
$$\sigma = \det(U^TV).$$
7. Compute the orthogonal matrix
$$A = U \operatorname{diag}(1, \cdots, 1, \sigma) V^T.$$
8. Compute the points Ax_i and y_i, $i = 1, \cdots, K$.
9. Compute the scalar $g = \sum_{i=1}^{K} (Ax_i, y_i)$.
10. Compute the optimum scale factor α:
$$\alpha = \begin{cases} g\Big/\sum_{i=1}^{K} (x_i, x_i), & g > 0 \\ 0, & \text{otherwise.} \end{cases}$$
11. Compute the translation vector
$$B = \bar{Y} - A\bar{X}.$$
12. Compute the residuals
$$r_i = y_i - Ax_i, \qquad i = 1, \cdots, K$$
$$(= Y_i - AX_i - B).$$

Following Step 12, the components of the residual vector r_i are checked for size. If they are all smaller in magnitude than a specified tolerance, the part is accepted. Otherwise the part is rejected.

The optimum scale factor can be used to generate another set of residuals $\hat{r}_i = Y_i - \alpha A\hat{X}_i - \hat{B}$, where $\hat{B} = \bar{Y} - \alpha A\bar{\bar{X}}$. These components should usually be smaller in magnitude than the specified tolerance for acceptance of the part. The (rare) situation may occur where some $\hat{r}_i$ are larger than the corresponding r_i. In the case where the r_i are within tolerances but the $\hat{r}_i$ are not, it seems that the part should still be accepted. Our justification for this statement is that the user of this analysis (most likely) has the two sets of points in the same scale, yielding residuals r_i. Thus some rigid transformation of one set of points to the other has been shown to exist with sufficiently small tolerances. This is, we think, the important point behind this type of analysis in the first place.

4. Numerical analysis. In § 3 the main steps of an algorithm are given that allows one to compute the orthogonal matrix A, the vector B, and the scalar α of the transformation $Y = \alpha AX + B$. The computation of A, B and the residuals $Y_I - (AX_I + B)$ are done in the provided subprogram MAP(·) for the special cases of $n = 2$ or $n = 3$. Each of these steps is straightforward with the exception of Step 4. In that step the SVD of an $n \times n$ matrix C is required. For general values of n, one can use the Golub–Reinsch algorithm as implemented in subroutine SVDRS(·), [1, p. 250]. However, as we noted in the Introduction, this particular problem is most likely to occur for values of $n \leqq 3$. For this reason a special and succinct version of the Golub-Reinsch algorithm was written for a 3×3 matrix. This subroutine, SVD3B3 (·), is called by MAP(·) and implements that algorithm in this special case. By avoiding the most expensive loops and using plane rotations computed by SROTG (·), [3, p. 314], for all the elimination steps, a robust and efficient code was obtained. The exclusive use of SROTG(·) was made because of the few elimination steps required for a 3×3 matrix. This version of the SVD for 3×3 matrices is at least a factor of five times faster than SVDRS (·) using the CDC 6600 computer. The package of subroutines and a sample test program are available directly from the authors. The primary subroutine of the package is MAP (·). This subroutine accepts data triples $\{X_I\}$ and $\{Y_I\}$ as input and returns the matrix A and the vector B, and the residuals $R_I = Y_I - AX_I - B$ as output. There are approximately 796 card images in this package. The program units are all written in highly portable Fortran. We believe that it can be used on most computers that support Fortran, with no changes required. It is anticipated that the primary interest in this package is with MAP (·). The usage instructions for this subroutine are as follows.

The user must have the following declaration statement in a program that calls MAP (·):

DIMENSION $X(3, K)$, $Y(3, K)$, $R(3, K)$, $A(3, 3)$, $B(3)$.

Assign values to K, $X(*, *)$, and $Y(*, *)$ and execute the subroutine call

CALL MAP (K, X, Y, R, A, B).

The parameters in the calling sequence fall into two classes: *Input* and *Output.*

Input

$K, X(3, K), Y(3, K)$ The integer K contains the number of triples of points, $\{X_I\}$ and $\{Y_I\}$. The arrays $X(3, K)$ and $Y(3, K)$ contain the X_I and Y_I, respectively. The first coordinate of X_I is stored in $X(1, I)$; the second is stored in $X(2, I)$; the third is stored in $X(3, I)$. The same convention holds for Y_I. (Note that for $n = 2$ variables, the third coordinate of both the X_I and Y_I should be fixed at the same value, say zero.)

Output

$R(3, K)$ The residuals $R_I = Y_I - AX_I - B$ are returned in the array $R(3, K)$. The coordinates of R_I are stored in the same convention as the coordinates of the X_I and Y_I.

$A(3, 3), B(3)$ The optimal orthogonal transformation matrix, A, and translation vector, B, are returned in the arrays $A(3, 3)$ and $B(3)$. The transformation has the form $Y = AX + B$. The elements a_{IJ} of A and the entries b_I of B are returned in $A(I, J)$ and $B(I)$, respectively.

As an illustration of the process we used MAP $(\cdot)$ to compute A and B from a set of data for both $\{X_I\}$ and $\{Y_I\}$.

TABLE 1
An example, for $n = 3$, computed using MAP $(\cdot)$.

I	X_I	Y_I	$1000 \times (R_I = Y_I - AX_I - B)$
	0.99962512	1.0	−0.82456
	1.00049650	1.0	0.28771
1	0.99903218	1.0	−0.23266
	0.99837196	1.0	0.93430
	0.99947869	1.0	−0.17797
2	−1.00200510	−1.0	0.80444
	−1.00184080	−1.0	1.39634
	−1.00025510	−1.0	0.28408
3	1.00015700	1.0	−0.38086
	−0.99843296	−1.0	−1.50609
	−1.00105830	−1.0	−0.39382
4	−1.00003210	−1.0	−0.19092
	0.99999990	0.00037760	0.00025253
A:	−0.00037741	0.99999965	−0.00074112
	−0.00025281	0.00074102	0.99999969
B:	0.00056948	0.00033381	0.00071211

5. Angular representation of the orthogonal matrix A for $n = 3$. Occasionally one needs to represent a 3×3 orthogonal matrix as a product of plane rotations or by the angles of these rotations. In a sense, the $3 \times 3 = 9$ pieces of data can be represented with just three pieces of data. (Often machines are manufactured so that achieving a rigid coordinate transformation can *only* be done by rotating about one coordinate axis, then another, etc.) It seems to the authors that there are compelling reasons to suggest that the use of Euler angles [4, p. 259–270] is the wrong approach for representing a broad class of orthogonal matrices. Briefly, the Euler angles are

derived from the three plane rotation matrices that diagonalize the orthogonal matrix A, determinant of $A = 1$. The $c_i = \cos(\theta_i)$ and $s_i = \sin(\theta_i)$, $i = 1, 2, 3$. First, c_1 and s_1 are chosen to eliminate the entry at the intersection of row 1 and column 3 of A. This rotation is applied to A to form the matrix product R_1A. Next, c_2 and s_2 are chosen to eliminate row 2, column 3 of R_1A. This rotation is applied to R_1A to form the product R_2R_1A. Finally, c_3 and s_3 are chosen to eliminate row 1, column 2 of R_2R_1A. The product $R_3R_2R_1A$ is the identity matrix by virtue of the fact that the R_i and A are orthogonal, and determinant of $A = 1$.

The central problem with this representation for A in our application is that the angles θ_i may not depend continuously on the data. This is true because typically A = Identity matrix + small terms. These small terms uniquely determine the θ_i, but an arbitrarily small change to these terms can make a change in the θ_i as much as 2π.

$$\begin{bmatrix} c_3 & s_3 & 0 \\ -s_3 & c_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_2 & s_2 \\ 0 & -s_2 & c_2 \end{bmatrix}\begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

FIG. 1. *Deriving Euler angles from an orthogonal matrix.*

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & c_3 & s_3 \\ 0 & -s_3 & c_3 \end{bmatrix}\begin{bmatrix} c_2 & 0 & s_2 \\ 0 & 1 & 0 \\ -s_2 & 0 & c_2 \end{bmatrix}\begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

FIG. 2. *Deriving stable set of angles from an orthogonal matrix.*

Our suggestion for a set of angle coordinates for A in our application is based on triangularizing A as shown in Fig. 2. The first plane rotation is chosen to eliminate row 2, column 1. This plane rotation is applied to A to form P_1A. Next, the second plane rotation is determined that eliminates row 3, column 1 of P_1A. This is applied to P_1A to form P_2P_1A. Finally, the third plane rotation is applied to eliminate row 3, column 2 of P_2P_1A. This yields the (continuous) representation $A = P_1^TP_2^TP_3^T$.

The principal angles θ_i satisfying $c_i = \cos(\theta_i)$, $s_i = \sin(\theta_i)$, $i = 1, 2, 3$ are determined using the two-argument arctangent function. This elementary function is available in most Fortran systems as ATAN2$(\cdot, \cdot)$. One can also use the idea of Stewart [3, p. 314] to essentially store just one number for the pair (c_i, s_i), $i = 1, 2, 3$. This is equivalent to storing the angles θ_i, but it avoids computing the arctangent, sine and cosine functions to reconstruct the rotation matrices.

REFERENCES

[1] C. L. LAWSON AND R. J. HANSON, *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[2] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York–London, 1973.

[3] C. L. LAWSON, R. J. HANSON, F. T. KROGH AND D. R. KINCAID, *Basic linear algebra subprograms for Fortran usage*, ACM Trans. Math. Software, 5 (1979), pp. 308–323.

[4] I. N. BRONSHTEIN AND K. A. SEMENDYAYEV, *A Guide-book to Mathematics for Technologists and Engineers*, Pergamon Press, New York, 1964.

[5] B. F. GREEN, *The orthogonal approximation of an oblique structure in factor analysis*, Psychometrika, 17 (1952), pp. 429–440.

[6] I. Y. BAR-ITZHACK, *Iterative optimal orthogonalization of the strapdown matrix*, IEEE Trans. Aero. and Electronic Systems, 11 (1975), pp. 30–37.

[7] P. H. SCHÖNEMANN, *A generalized solution of the orthogonal Procrustes problem*, Psychometrika, 31 (1966), pp. 1–10.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0001 $01.00/0

SOLUTION OF HOMOGENEOUS SYSTEMS OF LINEAR EQUATIONS ARISING FROM COMPARTMENTAL MODELS*

R. E. FUNDERLIC† AND J. B. MANKIN†

Abstract. Systems of linear differential equations with constant coefficients, $Ax = \dot{x}$, with the matrix A having nonnegative off-diagonal elements and zero column sums, occur in compartmental analysis. The steady-state solution leads to the homogeneous system of linear equations $Ax(\infty) = \dot{x}(\infty) = 0$. LU-factorization, the Crout algorithm, error analysis and solution of a modified system are treated.

Key words. M-matrix, homogeneous linear system, compartmental model, queueing networks, LU-factorization, tracer methods

Introduction. The mathematical theory of systems of linear, ordinary differential equations with constant coefficients was applied to tracer experiments to explain and analyze data consisting of concentrations in various organs [13], [14], [16], [19]. Mathematically, this corresponds exactly with the compartmental models later applied in ecology [9], [10], [11]. O'Neill [12] presented a survey paper on the use of compartmental models in ecology, and he dates their initial applications from about 1960.

A compartmental model is one in which the system is conceptualized as being n distinct, interconnected compartments between which usually either mass or energy is transported. Ideally, the compartments are assumed to be homogeneous and uniform with no internal gradients, but, in reality, all that is necessary is that the modeler conceptualize them mathematically as distinct.

A fundamental aspect of the model is the form of the transfer function between compartments. O'Neill [12] reviewed the conditions under which such transfer functions reasonably represent real systems. We will confine ourselves to the usual transfer functions that are a linear combination of the state variables. With this assumption, the dynamics of a compartmental model can be expressed by a system of first-order linear ordinary differential equations with constant coefficients.

The increase of atmospheric carbon dioxide as a result of increased fossil fuel combustion has stimulated the development and study of global carbon models. Many of the models of the carbon cycle are formulated as closed systems. A closed system is one in which there is no flow of mass across the boundaries of the system. In other words, the amount of mass within the system remains constant. This means that many of the carbon models may be considered as described by the following set of linear, ordinary differential equations with constant coefficients:

$$\dot{x} = Ax, \qquad x(t_0) = x_0, \tag{1}$$

where the elements of x represent concentrations of carbon in a compartment, e.g., troposphere, and the elements of the matrix A represent the transfer coefficients from one compartment to another. The solution of the steady state of this system then

* Received by the editors November 18, 1980.

† Union Carbide Corporation Nuclear Division, Computer Sciences Division, PO Box Y, Oak Ridge, Tennessee 37830. 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, and by the National Science Foundation Ecosystem Studies Program under Interagency Agreement DEB 77-25781 with the U.S. Department of Energy.

amounts to solving the homogeneous system of linear algebraic equations:

$$Ax = 0, \tag{2}$$

which has a nontrivial vector solution x if and only if A is singular (i.e., $\det A = 0$). Since the system is closed, the elements of the matrix A obey the following relationships:

$$a_{ij} \geqq 0, \quad i \neq j, \qquad a_{ii} = -\sum_{j \neq i} a_{ji}, \quad i = 1, 2, \cdots, n. \tag{3}$$

That is, the column sums of A are zero and therefore A is singular. Matrices with properties (3) will be called *compartmental matrices* or *C-matrices for brevity*. A more thorough derivation of the differential equation and a discussion of many of the fundamental properties of C-matrices are given in [3]. The negative of a C-matrix is in the class of M-matrices. A frequently used definition of an M-matrix is that its off-diagonal elements are nonpositive and the real parts of the eigenvalues are nonnegative. A comprehensive exposition on M-matrices has been given by Berman and Plemmons [1]. The conservation property of C-matrices

$$\sum_{i=1} x_i(t) = c \qquad \text{(a constant)}, \tag{4}$$

is assumed for all time, t. The problem, then, is to solve (2) for the state variables at steady state.

Reduction to irreducible systems. A compartmental system is said to be irreducible if there is a way, perhaps indirectly, for "material" to flow from each compartment to each of the other compartments. Compartmental matrices associated with irreducible systems are thus irreducible, and have rank one less than their order [2]. The system on the left in Fig. 1 gives rise to an irreducible matrix; the one on the right does not, since, for example, material does not pass from compartment 1 to compartment 3.

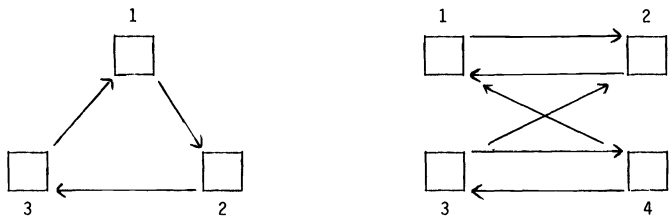


FIG. 1. *Irreducible* (a) *and reducible* (b) *systems.*

If a system is reducible, then there is a permutation of the labels of the compartments of the system that gives a matrix of the form

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ & A_{22} & \cdots & A_{2p} \\ & & \cdots & \vdots \\ 0 & & & A_{pp} \end{bmatrix}, \tag{5}$$

where the diagonal submatrices A_{ii} are square and either irreducible or zero matrices of order one (e.g., Berman and Plemmons [1, pp. 34–48]). Except for extremely large systems, there is good computer software [6] to effect this reduction. This report will assume that the matrices under consideration are irreducible C-matrices.

If all the submatrices above the diagonal blocks in (5) are zero, then the corresponding physical system is made up of p completely independent subsystems. The case of nonzero submatrices above the diagonal blocks is best described by an example. If a system is as in Fig. 2, then the steady state of the system will be solved by a system of equations of the form

$$\begin{bmatrix} * & * & 0 & 0 & * & 0 \\ * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & * & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{6}$$

where the *'s indicate nonzeros. Thus

$$A_{33} \begin{bmatrix} x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

can only have the solution $x_5 = x_6 = 0$, and only two systems of order 2 need be solved to provide a steady-state solution for the physical system in Fig. 2. This simple example shows the importance both physically and computationally of determining, if reasonably possible, the reducibility of the physical system.

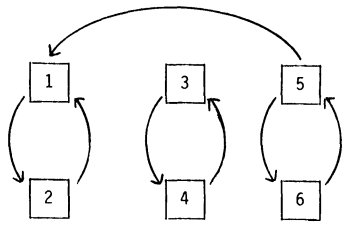


FIG. 2. *A reducible system in which compartments 5 and 6 will empty.*

***LU*-decompositions of compartmental matrices.** It is well known that the solution of general linear homogeneous systems is usually more difficult [5], [8] than that of nonsingular systems. However, the solution of irreducible compartmental systems will be shown to be relatively easy.

Singular irreducible M-matrices, A, have LU-decompositions such that L is lower triangular and U is upper triangular (Kuo [7] and cf. Berman and Plemmons [1, p. 157]). Furthermore, $l_{nn}u_{nn} = 0$, and L may be chosen as a nonsingular M-matrix. This shows that irreducible C-matrices have LU-decompositions. An alternate proof to Kuo's result is given here and it is shown that for a C-matrix A, L or U can be chosen as a C-matrix.

The existence of LU-decompositions for irreducible C-matrices is illustrated here by a system of order four. Let

$$A = \begin{bmatrix} -\sum a_i & b_1 & c_1 & d_1 \\ a_1 & -\sum b_i & c_2 & d_2 \\ a_2 & b_2 & -\sum c_i & d_3 \\ a_3 & b_3 & c_3 & -\sum d_i \end{bmatrix}.$$

Dropping the subscript from sums the first Gauss step [15] uses an elementary triangular matrix with first column

$$\begin{bmatrix} 1 \\ a_1/\sum a \\ a_2/\sum a \\ a_3/\sum a \end{bmatrix}.$$

There is no loss in generality and it simplifies the algebra to scale the elementary matrix by $\sum a$. Thus

$$\underset{M_1}{\begin{bmatrix} \sum a & 0 & 0 & 0 \\ a_1 & \sum a & 0 & 0 \\ a_2 & 0 & \sum a & 0 \\ a_3 & 0 & 0 & \sum a \end{bmatrix}} \underset{A}{\begin{bmatrix} -\sum a & b_1 & c_1 & d_1 \\ a_1 & -\sum b & c_2 & d_2 \\ a_2 & b_2 & -\sum c & d_3 \\ a_3 & b_3 & c_3 & -\sum d \end{bmatrix}}$$

$$= \left[\begin{array}{c|ccc} -(\sum a)^2 & b_1\sum a & c_1\sum a & d_1\sum a \\ \hline 0 & b_1a_1-\sum b\sum a & c_1a_1+c_2\sum a & d_1a_1+d_2\sum a \\ 0 & b_1a_2+b_2\sum a & c_1a_2-\sum c\sum a & d_1a_2+d_3\sum a \\ 0 & b_1a_3+b_3\sum a & c_1a_3+c_3\sum a & d_1a_3-\sum d\sum a \end{array}\right].$$

That is,

$$M_1A = \begin{bmatrix} * & * & * & * \\ 0 & & & \\ 0 & & A' & \\ 0 & & & \end{bmatrix},$$

where A' is an irreducible C-matrix. The irreducibility of A' follows because, otherwise, the expressions for the elements of A' would imply the matrix A to be reducible. Thus for general n-compartment irreducible systems, Gaussian elimination can be carried out on the successive A' submatrices without row interchanges. Finally, the A' matrix of order two will be reduced to a matrix with last row zero.

If a matrix has an LU-decomposition, and D is a nonsingular diagonal matrix, then $(LD)(D^{-1}U)$ is another such decomposition. This allows the choice of ones for the diagonal elements of U and thus irreducible C-matrices have LU-decompositions (illustrated for order four) of the form

$$LU = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ * & l_{22} & 0 & 0 \\ * & * & l_{33} & 0 \\ * & * & * & l_{44} \end{bmatrix} \begin{bmatrix} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \\ 0 & 0 & 0 & u_{nn} \end{bmatrix}, \tag{7}$$

where $l_{nn}u_{nn} = 0$. With this restriction, any convenient choice for l_{nn} and u_{nn} may be made. The same type of argument gives an alternate proof to Kuo's proof [7] that M-matrices have LU-factorizations of the form (7). Deletion of the last column of L and the last row of U gives what is called a full rank decomposition of A, i.e., the columns of L and those of U^T are each a set of linearly independent vectors. The choice $l_{nn} = 0$ and consideration of this full rank decomposition of A lead to a proof that L is a C-matrix. The next paragraph shows this.

Let LU be a full rank decomposition of an irreducible C-matrix as developed in the last paragraph, and let e be a column vector with all ones, Then since A is a C-matrix, $e^TA=0$ or $U^T(L^Te)=0$. Since U^T has linearly independent columns, $Le^T=0$. Thus the column sums of L are zero. The first Gauss step and the obvious induction show that the elements above the diagonal of U are nonpositive and those below the diagonal of L are nonnegative. Thus if L is again assumed square of order n and if l_{nn} is chosen as zero, L is a C-matrix.

The standard uniqueness theorem (e.g., Stewart [15, pp. 132–133]), for nonsingular LU-decompositions is easily modified for the rank $n-1$ C-matrices under consideration. Let L_1U_1 be an LU-factorization of a C-matrix A as in (7) with L_1 nonsingular. Then let L_2U_2 be any other LU-factorization of A. Then $L_1^{-1}L_2$ is lower triangular and

$$U_1=L_1^{-1}L_2U_2$$

is upper triangular. This implies that $L_1^{-1}L_2$ is a diagonal matrix D with nonzero diagonal elements, except for possibly the n, n element. Thus an LU-factorization of an irreducible C-matrix is unique up to a diagonal scale factor, i.e., $L_2=L_1D$ and $U_1=DU_2$.

Conversely, suppose a matrix A has an LU-decomposition with rank $(L)=$ rank $(U)=n-1$, $l_{nn}=u_{nn}=0$, L a C-matrix and U an M-matrix. One might expect that LU is a C-matrix. The following example shows this not to be the case:

$$A=\begin{bmatrix}-2 & 2 & 2c\\ 1 & -3 & 2-c\\ 1 & 1 & -(2+c)\end{bmatrix}=\begin{bmatrix}-2 & 0 & 0\\ 1 & -2 & 0\\ 1 & 2 & 0\end{bmatrix}\begin{bmatrix}1 & -1 & -c\\ 0 & 1 & -1\\ 0 & 0 & 0\end{bmatrix}.$$

In this example, c must be between zero and two for the LU-decomposition to be a C-matrix.

In summary, despite the singularity of irreducible C-matrices, they have LU-decompositions. The singularity is always characterized by $l_{nn}u_{nn}=0$. Other than this arbitrariness and the scaling of the diagonal elements of U, an LU-decomposition is unique. If the diagonal elements of U are chosen positive, L is a C-matrix and in any case its column sums other than the last are zero. If u_{nn} is chosen as zero, then the system $Ax=0$ can be solved by solving the equivalent triangular system $Ux=0$. This will be discussed in the next section.

A simple stable method for homogeneous compartmental systems. Given an irreducible C-matrix, it was shown in the last section that there is an LU-decomposition such that L is nonsingular and U has ones on the diagonal except for $u_{nn}=0$. The Crout algorithm (e.g., Stewart [15, p. 135]), without pivoting will give this LU-decomposition because no diagonal zeros occur until the last step. The algorithm is particularly simple to implement. The two steps in Algorithm 1 are carried out for $k=1,2,\cdots,n-1$.

ALGORITHM 1. Crout for Irreducible C-matrices.

$$L\colon a_{ik}\leftarrow a_{ik}-\sum_{m=1}^{k-1}a_{im}a_{mk},\qquad i=k,k+1,\cdots,n,$$

$$U\colon a_{kj}\leftarrow a_{kk}^{-1}\left(a_{kj}-\sum_{m=1}^{k-1}a_{km}a_{mj}\right),\qquad j=k+1,\cdots,n.$$

After the two steps in Algorithm 1 are carried out, the original C-matrix A will have been overwritten by the elements of L and the elements of U above its diagonal. The diagonal elements of U, except for u_{nn}, are implicitly assumed to be one, and l_{nn} being an arbitrary nonzero number is not calculated. The back substitution to calculate the solution vector x to $Ax = 0$ can start with $x_n = 1$ and then the vector must be scaled to conform to the conservation property, $\sum x = c$. Algorithm 2 assumes U has been calculated by the Crout algorithm (Algorithm 1).

ALGORITHM 2. Back Substitution for $Ux = 0$.

$$x_n = 1$$

$$x_i \leftarrow -\sum_{j=i+1}^{n} a_{ij}x_j, \qquad i = n-1, n-2, \cdots, 1$$

$$a = c \Big/ \sum_{i=2}^{n} x_i$$

$$x_i \leftarrow ax_i, \qquad i = 1, 2, \cdots, n$$

It was shown in the last section that the elements of U above the diagonal are nonpositive. Thus, from the second line of Algorithm 2, the components of the steady-state vector x are all nonnegative. Fiedler and Ptak [2] have shown that such a vector has strictly positive elements and the strict positivity follows from irreducibility.

To say a computer method is stable (e.g., Stewart [15, pp. 75–78]), is to say that the calculated solution is the actual solution to a problem close to the original problem. It is often necessary and often helps stability if row interchanges are performed to solve systems of equations (e.g., Stewart, [15, pp. 137–138]). For C-matrices, the ordering determined by row interchanges is precisely that given by no interchanges. The analysis to show that the Crout algorithm is stable for diagonally dominant matrices was done by Wilkinson [18] (also cf. Wendroff [17, pp. 122–123]). Wilkinson's analysis holds for irreducible C-matrices. Thus if $\bar{x}$ is the calculated solution on a t-digit decimal computer to a n-compartment system $Ax = 0$, then there is a matrix E such that

$$(A+E)\bar{x} = 0,$$

with the elements of E bounded:

$$|\varepsilon_{ij}| \leqq np\ (\max |a_{ij}|)10^{-t}, \tag{8}$$

where p is of order unity. It should be noted that $A+E$ is singular since the back substitution is started with $u_{nn} = 0$. The factor n in (8) may be eliminated ([18], cf. Stewart [15, p. 153]) if the inner products in Algorithm 1 are accumulated in double precision.

The solution of a modified system. Often the elements of a C-matrix are not well determined, or changing the coefficients is desired for other reasons. Thus it is important to have a cheap way to obtain the solution, x', to a modified system $A'x' = 0$.

More formally if σ is subtracted from an element a_{ij} and added to a_{jj} with $\sigma \leqq a_{ij}$ so that A' is a C-matrix, then

$$A' = A - \sigma(e_i - e_j)e_j^T,$$

where e_k is a kth unit vector, i.e., one as its kth component, zeros elsewhere.

The update method developed here is essentially the Sherman–Morrison method [15] for nonsingular systems. A necessary condition for this singular version is that the

difference of unit vectors be in the column space of any C-matrix. The following argument shows that containment to be true.

If a vector is in the column space of A, then it is of the form $\sum\alpha_i a_i$ where a_i are the columns of A such that $e^T a_i = 0$ where e is a column vector of all ones. Thus the set

$$\{w : e^T w = 0\} \tag{9}$$

contains the column space of A, and clearly is a subspace with dimension one less than the order of A. Therefore the set (9) must be the column space of A, i.e., the column space of A is the set of all vectors orthogonal to the vector e.

From an LU-decomposition of A, the system

$$Az = LUz = e_i - e_j$$

can then easily be solved in two steps: (a) solve $Lq = e_i - e_j$, (b) $Uz = q$. This requires much less work than solving $Ax = 0$ since L and U have already been calculated. A vector x' satisfying $A'x' = 0$ may be expressed in terms of the vectors x and z:

$$x' = x - [\sigma x_j/(\sigma z_j - 1)]z, \qquad \sigma z_j - 1 \neq 0. \tag{10}$$

Since

$$A'z = (1 - \sigma z_j)(e_i - e_j), \tag{11}$$

it suffices to choose $x' = z$, if $\sigma z_i - 1 = 0$.

If the same i, j, element of A' is to be further modified, then from (11)

$$(1 - \sigma z_j)^{-1} z$$

may take the role of the new z.

The solution vector x' given by (10) is not zero, for if it were, then

$$(\sigma z_j - 1)x = \sigma x_j z.$$

But in this case since $Ax = 0$,

$$Az = 0 = e_i - e_j,$$

a contradiction.

Other types of modifications for both C-matrices and more general matrices will be considered in future work with R. J. Plemmons and G. H. Golub.

Examples. If

$$A = \begin{bmatrix} -5 & 2 & 1 \\ 3 & -3 & 1 \\ 2 & 1 & -2 \end{bmatrix},$$

then an LU-decomposition for A is

$$\begin{bmatrix} -5 & 0 & 0 \\ 3 & -\frac{9}{5} & 0 \\ 2 & \frac{9}{5} & 1 \end{bmatrix} \begin{bmatrix} 1 & -\frac{2}{5} & -\frac{1}{5} \\ 0 & 1 & -\frac{8}{9} \\ 0 & 0 & 0 \end{bmatrix}.$$

Algorithm 1 overwrites the matrix A with

$$\begin{bmatrix} -5 & -\frac{2}{5} & -\frac{1}{5} \\ 3 & -\frac{9}{5} & -\frac{8}{9} \\ 2 & \frac{9}{5} & -2 \end{bmatrix}.$$

If the sum of the components of the steady-state solution x is assumed to be one, i.e., $c = 1$, then Algorithm 2 gives $x^T = (\frac{5}{22}, \frac{8}{22}, \frac{9}{22})$.

An example from the ecological modeling literature [4] is illustrated in Fig. 3. The relevant C-matrix is

$$\begin{bmatrix} -0.5000 & 0.08827 & 0 & 0 & 0 \\ 0.5000 & -0.47047 & 0.05682 & 0.08123 & 0 \\ 0 & 0.19110 & -0.05682 & 0 & 0 \\ 0 & 0.19110 & 0 & -0.11779 & 0.001198 \\ 0 & 0 & 0 & 0.03656 & -0.001198 \end{bmatrix}.$$

An LU-decomposition computed for A had

$$L = \begin{bmatrix} -0.5000 & 0 & 0 & 0 & 0 \\ 0.5000 & -0.3822 & 0 & 0 & 0 \\ 0 & 0.1911 & -0.02841 & 0 & 0 \\ 0 & 0.1911 & 0.02841 & -0.03656 & 0 \\ 0 & 0 & 0 & 0.03656 & 1 \end{bmatrix},$$

and

$$U = \begin{bmatrix} 1 & -0.17654 & 0 & 0 & 0 \\ 0 & 1 & -0.14867 & -0.2153 & 0 \\ 0 & 0 & 1 & -1.4296 & 0 \\ 0 & 0 & 0 & 1 & -0.032768 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

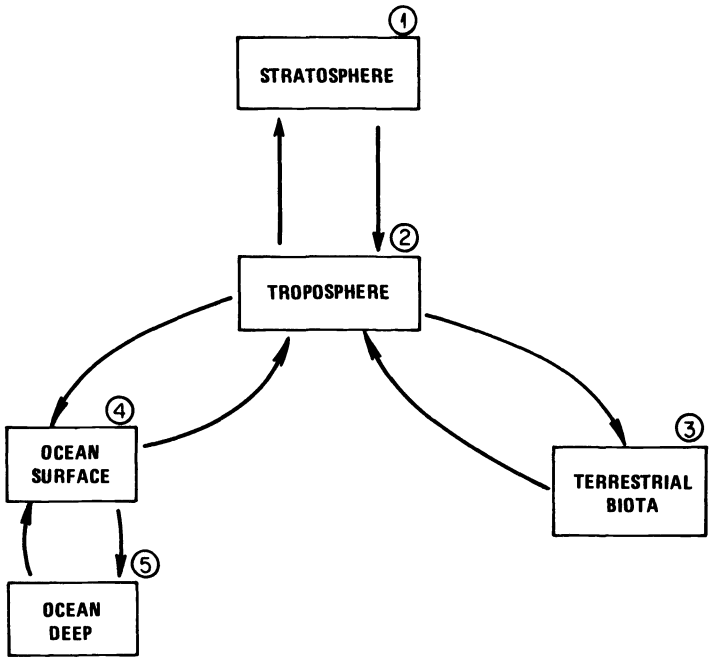


FIG. 3. *A carbon model.*

The matrix A was overwritten by L and the elements above the diagonal elements of U. The a_{nn} element, however, does remain unchanged. The sum of the coefficients of the solution vector, i.e., c, was assumed to be 41145.6 as in [4]. This resulted in a solution vector calculated as

$$x^T = (92.34, 523.04, 1759.12, 1230.49, 37551.61).$$

Acknowledgment. The authors are happy to acknowledge the helpful comments of M. T. Heath, R. J. Plemmons and R. C. Ward.

REFERENCES

[1] A. BERMAN AND R. J. PLEMMONS, *Nonnegative Matrices in the Mathematical Sciences*, Academic Press, New York, 1979.

[2] M. FIEDLER AND V. PTÁK, *On matrices with nonpositive off-diagonal elements and positive principal minors*, Czechoslovak Math. J., 12 (1962), pp. 382–400.

[3] R. E. FUNDERLIC AND M. T. HEATH, *Linear compartmental analysis of ecosystems*, ORNL/IBP-71/4, Oak Ridge National Laboratory, Tennessee, 1971.

[4] R. H. GARDNER, J. B. MANKIN AND W. R. EMANUEL, *A comparison of three carbon models*, Ecol. Modelling, 8 (1980), pp. 313–332.

[5] G. H. GOLUB AND F. T. LUK, *Singular value decomposition: applications and computations*, in Trans. 22nd Conference of Army Mathematicians, Army Res. Off. Rep. 77-1, 1977, pp. 577–605.

[6] M. J. HOPPER, ed., *Harwell Subroutine Library*, MC13D *Subroutine*, Harwell Rep. AERE-R9185, Harwell England, 1980, p. 29.

[7] I. KUO, *A note on factorizations of singular M-matrices*, Linear Algebra Appl., 16 (1977), pp. 217–220.

[8] C. MOLER, *Three research problems in numerical linear algebra*, in Proc. Symp. in Applied Mathematics, 22, G. H. Golub and J. Oliger, eds., American Mathematical Society, Providence, 1978.

[9] R. B. NEEL AND J. S. OLSON, *Use of analog computers for simulating the movement of isotopes in ecological systems*, ORNL/3171, Oak Ridge National Laboratory, Tennessee 1962.

[10] J. S. OLSON, *Health Physics Annual Report*, ORNL/4446, Oak Ridge National Laboratory, Tennessee, 1959.

[11] ———, *Health Physics Annual Report*, ORNL/4634, Oak Ridge National Laboratory, Tennessee, 1960.

[12] R. V. O'NEILL, *A review of compartmental analysis*, in Ecosystems Science in Compartmental Analysis of Ecosystem Models, J. H. Matis, V. C. Patton and G. C. White, eds., International Cooperative Publishing House, Fairland, MD 1979, pp. 3–37.

[13] C. W. SHEPPARD AND A. S. HOUSEHOLDER, *The mathematical basis of the interpretation of tracer experiments in closed steady-state systems*, J. Appl. Phys. 22 (1951), pp. 510–520.

[14] A. K. SOLOMON, *Equations for tracer experiments*, J. Clinical Investigation, 28 (1949), pp. 1297–1307.

[15] G. W. STEWART, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[16] T. TEORELL, *Kinetics of distribution of substances administered to the body*, Arch. Internal Pharmacodynamie, 57 (1937), pp. 205–240.

[17] B. WENDROFF, *Theoretical Numerical Analysis*, Academic Press, New York, 1966.

[18] J. H. WILKINSON, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 8 (1961), pp. 281–330.

[19] D. B. ZILVERSMIT, C. ENTENMAN AND M. C. FISHLER, *On the calculation of turnover time and turnover rate from experiments involving the use of labeling agents*, J. General Physiol., 26 (1943), pp. 325–331.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0002 $01.00/0

CONDITION NUMBER ESTIMATION FOR SPARSE MATRICES*

ROGER G. GRIMES† AND JOHN G. LEWIS†

Abstract. The LINPACK package of linear equation solving software provides a reliable and inexpensive algorithm for estimating the condition number of a dense matrix. The direct generalization to banded or sparse matrices is reliable, but not necessarily inexpensive. The simple modification described in this note can bring the cost of the algorithm back to a reasonable level.

Key words. matrix condition number, sparse matrices

1. Introduction. The most common estimate of the stability of the computation of the solution to the linear system, $\mathbf{Ax} = \mathbf{b}$, requires knowledge of the condition number of $\mathbf{A}$ with respect to inversion,

$$\kappa_1(\mathbf{A}) = \|\mathbf{A}\|_1 \|\mathbf{A}^{-1}\|_1.$$

This paper is concerned with reliable and efficient a posteriori estimates for this condition number.

Systems of linear algebraic equations are usually solved by computing an **LU** decomposition with some form of reordering. Cline, Moler, Stewart and Wilkinson [1] discuss an algorithm which uses the **LU** decomposition to compute an estimate of the condition number $\kappa_1(\mathbf{A})$. (See [6] for a simple modification to improve the accuracy of the condition number estimate.) Their algorithm may be summarized as follows:

1. Select a vector $\mathbf{b}$ of $+1$'s and -1's so as to locally maximize growth in the accumulated row sums for the solution of $\mathbf{w}$ where $\mathbf{U}^T\mathbf{w} = \mathbf{b}$.
2. Solve: $\mathbf{L}^T\mathbf{x} = \mathbf{w}$; $\mathbf{Ly} = \mathbf{x}$; $\mathbf{Uz} = \mathbf{y}$.
3. Then $\kappa_1(\mathbf{A}) \approx \|\mathbf{A}\|_1 \|\mathbf{z}\|_1 / \|\mathbf{x}\|_1$.

This algorithm is used in all the condition number estimation subroutines of LINPACK [2] with an implementation designed to work for large classes of problems. Particular care has been taken to avoid problems when the matrices and/or the condition number estimate include numbers near the overflow level of the computer. The vectors involved in the algorithm are chosen to show large growth. In order to prevent this growth from causing overflows, the LINPACK subroutines scale the current solution vector ($\mathbf{w}$ in Step 1, $\mathbf{x}$, $\mathbf{y}$, and then $\mathbf{z}$ in Step 2) whenever the magnitude of a component of the solution vector in the forward or back substitution grows larger than one. The scale factor is chosen to reduce that component to be of magnitude one. The LINPACK code also scales the solution vector to unity in the 1-norm after solving each of the four triangular systems. This latter scaling causes a significant reduction in the frequency of scaling during the following solution steps, but does not eliminate intermediate scaling altogether, nor does it affect the number of scaling operations performed in solving the initial special triangular system. In the discussion which follows, the work of scaling shall refer only to the intermediate scaling operations (to Chebyshev norm unity), and not to the scaling performed after a triangular system has been solved.

Scaling adds to the cost of the algorithm. In the worst case the scaling procedure might require $4n^2$ multiplications. This is inconsequential compared to the cost of factoring a dense matrix, but scaling can dominate the cost of computing the **LU**

* Received by the editors December 15, 1980, and in revised form April 27, 1981.

† Energy Technology Applications Division, Boeing Computer Services Company, Mail Stop 9C-01, 565 Andover Park West, Tukwila, Washington 98188.

decomposition of a banded or sparse matrix. In this note we propose a modification to the LINPACK implementation which reduces the frequency of scaling. The modification preserves the robustness while reducing the cost of scaling. A problem dependent bound on the cost of scaling which can be much less than $O(n^2)$ is given in the appendix. The modified algorithm has been implemented in subroutines to estimate the condition number of matrices factored in either a symmetric envelope format or a symmetric general sparse format, in addition to the dense and banded formats of the LINPACK codes.

2. Reduction of scaling operations. The LINPACK codes provide for scaling the "solution" vector in solving any of the three ordinary triangular systems, whenever the next component to be determined has magnitude greater than one. Scaling will occur in the special initial triangular system if the larger of the two possible outcomes has magnitude greater than one. When scaling is required, the current vector is scaled so that its largest component (the current component) has magnitude reduced to one. This can cause frequent scaling by numbers of magnitude near one. An extreme example of frequent scaling is illustrated by the diagonal matrix $\mathbf{D} = \text{diag}\,(d_1, d_2, \cdots, d_n)$, where $d_i = 1 - i \cdot \delta$ for $0 < \delta \ll 1$. For any n, the LINPACK codes will scale n times performing a total of n^2 multiplications. Example $\mathbf{A}_3$ in Table 1 below requires more than $2n^2$ additional multiplications. The maximum work for scaling is clearly $4n^2$.

Scaling is necessary for computational reliability since preventing the current component of the solution vector from growing larger than one reduces the likelihood of overflow. But scaling should provide some significant reduction in the size of the numbers encountered. The role of scaling is to prevent overflow without causing undue loss of accuracy in (small) components already computed. We propose that the scale factor should have magnitude bounded away from one and that the scale factor always be taken to be the smaller of the LINPACK scale factor and some number γ less than one. The LINPACK codes can be changed to incorporate this feature by altering four lines in each of the condition number estimation subroutines. Each appearance inside a DO loop of the code represented by

```
IF (···) GO TO ···
  S = "scale factor"
  CALL SSCAL (N, S, ·, 1)
```

should be changed to read

```
IF (···) GO TO ···
  S = AMIN1 (γ, "scale factor")
  CALL SSCAL (N, S, ·, 1).
```

The scalar γ should be chosen to avoid loss of accuracy. After some experimentation the authors settled on $\gamma = .001$ which effectively reduced the number of scaling operations while preserving accuracy in the estimate of the condition number on a computer with a wide dynamic range. The use of a reciprocal power of the machine radix may have minor advantages and a choice of $\gamma = .5$ will provide a bound on the cost of scaling without significantly reducing the robustness of the algorithm for limited dynamic range computers. For the matrix $\mathbf{D}$ described earlier a scale factor of .5 reduces the number of scaling operations from n to 1.

3. Timing comparisons. We modified LINPACK subroutine SPBCO (single precision positive definite symmetric banded linear equation factorizer and condition number estimator) to demonstrate the effect of the changes suggested above. Test

matrices were selected from two areas. The first was a set of parameterized problems chosen from Gregory and Karney [4]. These three problems have the form

$$\mathbf{A}_1 = \begin{bmatrix} 2 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & & \\ & & & \ddots & -1 & & \\ & & & -1 & 2 & -1 & \\ & & & & -1 & 2 & \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 3 & -1 & & & & & \\ -1 & 3 & -1 & & & & \\ & -1 & 3 & -1 & & & \\ & & & \ddots & -1 & & \\ & & & -1 & 3 & -1 & \\ & & & & -1 & 3 & \end{bmatrix},$$

$$\mathbf{A}_3 = \begin{bmatrix} 5 & -4 & 1 & & & \\ -4 & 6 & -4 & 1 & & \\ 1 & -4 & 6 & -4 & 1 & \\ & & \ddots & & & \\ & & 1 & -4 & 6 & -4 \\ & & & 1 & -4 & 5 \end{bmatrix}$$

These parameterized problems were tested with $n = 500$, 1,000, and 2,000. The remaining two test matrices were chosen from a set of stiffness matrices from dynamic structural analyses. These last two matrices had order 203 and 396, with half bandwidth 21 and 112 respectively.

Table 1 displays the results of the testing with $\gamma = 1, .5$, and .001. The timings given are in seconds. The factorization time (performed by SPBFA) is given separately from the time spent in estimating the condition number. The number of scaling operations is given in parentheses following the execution time. These tests were performed on a CDC CYBER 175 computer using the FTN 4.8 compiler (OPT = 2) under the NOS 1.4 operating system.

TABLE 1
Timing results

Test Problem	N	Condition Number	SPBFA* Time	SPBCO Timings* $\gamma = 1$	$\gamma = .5$	$\gamma = .001$
$\mathbf{A}_1$	500	1.00×10^5	.011	.102 (499)	.044 (8)	.043 (1)
	1,000	4.01×10^5	.021	.309 (999)	.090 (9)	.088 (1)
	2,000	1.60×10^6	.041	1.051 (1999)	.180 (10)	.177 (2)
$\mathbf{A}_2$	500	5.00	.011	.045 (1)	.045 (1)	.045 (1)
	1,000	5.00	.021	.087 (1)	.087 (1)	.087 (1)
	2,000	5.00	.041	.178 (1)	.178 (1)	.178 (1)
$\mathbf{A}_3$	500	1.03×10^{10}	.020	.221 (1409)	.053 (40)	.049 (5)
	1,000	1.65×10^{11}	.040	.734 (2853)	.104 (45)	.093 (7)
	2,000	2.62×10^{12}	.075	2.699 (5769)	.202 (50)	.191 (7)
$\mathbf{A}_4$	203	7.07×10^{12}	.141	.052 (17)	.051 (12)	.051 (7)
$\mathbf{A}_5$	396	2.15×10^6	2.376	.272 (16)	.270 (12)	.267 (5)

* Timings are given in CPU seconds on a CDC CYBER 175 computer using the FTN 4.8 compiler (OPT = 2) under the NOS 1.4 operating system. The number of scaling steps used in SPBCO is given in parentheses after the actual execution time.

Problems $\mathbf{A}_1$ and $\mathbf{A}_3$ demonstrate the significant savings which can accrue from reducing the frequency of scaling operations. The remaining problems show only small savings in execution time. As indicated by these results the proposed modification to the LINPACK implementation of the Cline, Moler, Stewart and Wilkinson algorithm offers substantial savings in execution time for some problems without a loss of portability or accuracy and with no increase in storage requirements.

4. Appendix. We present here two theoretical results which show that our modification to the LINPACK algorithm can never result in more work and which give a bound on the work needed which can be much less than $4n^2$.

Both results view the LINPACK algorithm as the successive solution of four triangular linear systems:

$$\begin{aligned} \mathbf{U}^T\mathbf{w} &= \mathbf{b}, & \hat{\mathbf{w}} &= \mathbf{w}/\|\mathbf{w}\|_1, \\ \mathbf{L}^T\mathbf{x} &= \hat{\mathbf{w}}, & \hat{\mathbf{x}} &= \mathbf{x}/\|\mathbf{x}\|_1, \\ \mathbf{L}\mathbf{y} &= \hat{\mathbf{x}}, & \hat{\mathbf{y}} &= \mathbf{y}/\|\mathbf{y}\|_1, \\ \mathbf{U}\mathbf{z} &= \hat{\mathbf{y}}. \end{aligned}$$

Each of these steps will be viewed as the solution of

$$\mathbf{T}\mathbf{u} = \mathbf{v},$$

where $\mathbf{T}$ is a triangular matrix.

THEOREM 1. *The modified algorithm performs a scaling on the solution vector at the* kth *step of solving* $\mathbf{T}\mathbf{u} = \mathbf{v}$ *only if the original algorithm does also.*

Proof. Let P_j be the accumulated product of scale factors for the first j steps of the original algorithm, and similarily define $\hat{P}_j$ for the modified algorithm. Let $u^{(1)}$, $u^{(2)}, \cdots, u^{(n)}$ be the components of the solution without scaling, in the order determined. The original algorithm rescales the solution whenever $P_{j-1} < |u^{(j)}|$ and sets $P_j = |u^{(j)}|$. Clearly

$$P_k = \max\left(\max_{j=1,\cdots,k} (|u^{(j)}|), 1\right).$$

The modified algorithm rescales the solution whenever $\hat{P}_{j-1} < |u^{(j)}|$ and sets $\hat{P}_j \geqq |u^{(j)}|$. Thus

$$\hat{P}_k \geqq \max\left(\max_{j=1,\cdots,k} (|u^{(j)}|), 1\right),$$

and so $\hat{P}_k \geqq P_k$. Since the modified algorithm only rescales whenever $\hat{P}_{j-1} < |u^{(j)}|$, it follows from the fact that $\hat{P}_k \geqq P_k$ that the original algorithm must also rescale at this point.

THEOREM 2. *The number of multiplications performed in rescaling by the modified algorithm is bounded above by the smaller of* $4n^2$ *and*

$$\begin{aligned} S_\gamma(\mathbf{A}) = n \cdot (&\max(\log_{(1/\gamma)} \|\mathbf{L}^{-1}\|_1^2 + 2, 0) \\ &+ \max(\log_{(1/\gamma)} \|\mathbf{U}^{-1}\|_1^2 + 2, 0)), \end{aligned}$$

where we assume that $\mathbf{A} = \mathbf{L}\mathbf{U}$ *is nonsingular.*

Proof. The solution at each of the four steps has the form $\mathbf{u} = \mathbf{T}^{-1}\mathbf{v}$, where $\mathbf{T}$ is triangular and $\|\mathbf{v}\|_\infty \leqq 1$. Clearly $\|\mathbf{u}\|_\infty \geqq (1/\gamma)^{s-1}$, where s is the number of rescaling steps taken. It follows that

$$s \leqq \max(\log_{(1/\gamma)} \|\mathbf{u}\|_\infty + 1, 0).$$

The bound $S_\gamma(\mathbf{A})$ follows by applying standard norm inequalities to each step of the algorithm. First,

$$\|\mathbf{w}\|_\infty \leqq \|\mathbf{U}^{-T}\|_\infty \|\mathbf{b}\|_\infty = \|\mathbf{U}^{-1}\|_1,$$

so

$$s_1 \leqq \max\,(\log_{(1/\gamma)} \|\mathbf{U}^{-1}\|_1 + 1, 0).$$

Second,

$$\|\mathbf{x}\|_\infty \leqq \|\mathbf{L}^{-T}\|_\infty \|\hat{\mathbf{w}}\|_\infty \leqq \|\mathbf{L}^{-1}\|_1,$$

so

$$s_2 \leqq \max\,(\log_{(1/\gamma)} \|\mathbf{L}^{-1}\|_1 + 1, 0).$$

Next,

$$\|\mathbf{y}\|_\infty \leqq \|\mathbf{y}\|_1 \leqq \|\mathbf{L}^{-1}\|_1 \|\hat{\mathbf{x}}\|_1 = \|\mathbf{L}^{-1}\|_1,$$

and hence

$$s_3 \leqq \max\,(\log_{(1/\gamma)} \|\mathbf{L}^{-1}\|_1 + 1, 0).$$

Similarily,

$$\|\mathbf{z}\|_\infty \leqq \|\mathbf{z}\|_1 \leqq \|\mathbf{U}^{-1}\|_1 \|\hat{\mathbf{y}}\|_1 = \|\mathbf{U}^{-1}\|_1,$$

and hence

$$s_4 \leqq \max\,(\log_{(1/\gamma)} \|\mathbf{U}^{-1}\|_1 + 1, 0).$$

The bound $S_\gamma(\mathbf{A})$ is obtained by taking $S_\gamma(\mathbf{A}) = s_1 + s_2 + s_3 + s_4$.

We note that both $S_\gamma(\mathbf{A})$ and the actual number of rescaling operations vary when $\mathbf{A}$ is multiplied by a scalar.

5. Acknowledgments. This work was part of the GT-STRUDL LANCZOS project within the Energy Technology Applications Division of Boeing Computer Services. This refinement in the condition number estimation algorithm has been implemented in conjunction with the symmetric envelope factorizer and the symmetric general sparse factorizer in SPARSPAK [3]. Condition number estimation is used in conjunction with error analysis in the LANCZOS eigenanalysis algorithm for the symmetric generalized eigenvalue problem [5].

REFERENCES

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

[2] J. DONGARRA, J. BUNCH, C. MOLER AND G. STEWART, LINPACK *Users' Guide*, Society for Industrial and Applied Mathematics, Philadelphia, 1979.

[3] A. GEORGE, J. LIU AND E. NG, *User Guide to* SPARSPAK: *Waterloo Sparse Linear Equations Package*, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1979.

[4] R. GREGORY AND D. KARNEY, *A Collection of Matrices for Testing Computational Algorithms*, Krieger, Huntington, New York, 1978.

[5] J. LEWIS, R. GRIMES, *Practical Lanczos algorithms for solving structural engineering eigenvalue problems*, in *Sparse Matrices and Their Uses*, I. Duff, ed., Academic Press, 1981, New York,

[6] D. P. O'LEARY, *Estimating matrix condition numbers*, this Journal, 1 (1980), pp. 205–209.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0003 $01.00/0

NUMERICAL TECHNIQUE TO TRACE THE LOCI OF THE COMPLEX ROOTS OF CHARACTERISTIC EQUATIONS*

E. BAHAR† AND M. FITZWATER†

Abstract. A numerical technique is presented to compute the complex roots of characteristic equations for a wide class of engineering problems. Approximate values for the roots need not be known a priori and closed form analytical expressions for the derivative of the modal equations are not required. A method has also been developed to trace the loci of the complex roots as one or several parameters of the characteristic equation vary.

The key to the method described in this paper is the determination of the covering space on which all the complex roots lie. As a result the complex roots can be found using standard numerical techniques developed for the real zero problem.

As an illustrative example, the solution of the modal equation for the vertically polarized waves in an irregular spheroidal model of the earth-ionosphere waveguide is presented. Since it is not necessary to use several pole-free forms of the modal equation to overcome overflow and underflow problems in the numerical computations, one can avoid the inadvertent search for "phantom" roots and the possibility of losing a root due to the necessity to switch functions over the complex

Key words. complex roots, characteristic equation, root loci, conformal mapping, covering space, waveguide modes

1. Introduction. The problem of determining the complex roots of characteristic (modal) equations is crucial to the resolution of a large variety of problems in mathematical physics. However, as pointed out by Hamming [13], "It is curious that the real-zero problem has been extensively investigated while the complex-zero problem has generally been ignored. Evidently, it is a field ripe for further research." Thus more attention has been given to this problem in recent years [14].

The motivation for this work is the problem of coupling of the characteristic waves (or modes) in a laterally nonuniform model of the earth-ionosphere waveguide. The medium of propagation and the boundaries are in general dissipative, thus the roots ν^n $(n = 1, 2, 3, \cdots)$ of the characteristic (modal) equation

$$G(\nu) = |G(\nu)| \exp [i\psi(\nu)] = 0, \tag{1.1}$$

are the complex values of the orders of spherical Bessel functions with complex arguments. Since the physical dimensions and the electromagnetic parameters of the waveguide are assumed to vary along the propagation path, it is also necessary to determine the locus of each complex root ν^n of the characteristic (modal) equation over a range of parameters describing the earth-ionosphere waveguide.

The characteristic equation for the earth-ionosphere waveguide is dealt with in detail in this work since it poses several particularly difficult problems not usually encountered in characteristic equations (such as Nth order polynomials). A principal difficulty with the modal equation for the earth ionosphere waveguide stems from the overflow and underflow problems one encounters in evaluating the spherical Bessel functions of complex order and argument that appear in the equation. As a result it is not possible to express the modal equation in a single pole-free form which is suitable for numerical computations in the entire region of the complex ν plane where all the relevant characteristic roots lie. Furthermore, since the roots ν^n are orders of the spherical Bessel functions, no closed form analytic expressions for the derivatives of the modal

* Received by the editors January 28, 1980, and in revised form March 20, 1981.
† Electrical Engineering Department, University of Nebraska—Lincoln, Lincoln, Nebraska 68588.

equation with respect to ν are known. Thus it is very difficult to apply some of the standard root finding techniques to problems of radio wave propagation over the earth's surface.

In a recent technical report [12], Goodhart and Pappert describe in detail a novel root-finding technique, developed by Shellman and documented by Morfitt [18], for the evaluation of the complex roots ν^n $(n = 1, 2, 3, \cdots)$ of the characteristic equation (1.1). In their work the authors exploit the properties of functions of complex variables which are analytic everywhere except at isolated poles. Thus special consideration is given to the properties of the phase of $G(\nu)$ around nth order poles or zeros where the constant phase contours of $G(\nu)$ intersect. Furthermore, since the phase of $G(\nu)$ increases by $2\pi m$ along a contour (in a counterclockwise sense) enclosing only an mth order zero while the phase decreases by $2\pi n$ along a similar contour enclosing only an nth order pole, the accumulated phase change of the function $G(\nu)$ around any closed (counterclockwise) contour C in the complex ν plane is equal to 2π times the number of zeros minus the number of poles. This assumes that no zeros or poles lie on C and that mth order zeros or poles are accounted for m times. Goodhart and Pappert [12] describe in detail the method they use to track the constant phase contours using the information on the phase of $G(\nu)$ at the four corners of small squares in the complex ν-plane.

In order to overcome underflow and overflow problems arising from the large variations in the magnitudes of the functions appearing in the modal equation and in order to use pole-free versions of the modal equation, Goodhart and Pappert employ eight different forms of the modal equation. One set of four equations is continuous while the other set of four equations is discontinuous. As pointed out by them, use of the continuous forms, however, introduces many spurious phase contours while use of the discontinuous forms could lead to the loss of a root of the modal equation. The authors also alert the reader to other complications such as situations that could lead to an inadvertent search for "phantom" roots.

Once a mesh square is known to contain a zero, a more precise location of the zero is obtained by Goodhart and Pappert by an interpolation scheme which employs both the magnitude and phase of the function $G(\nu)$, [18]. Finally, the Newton–Raphson method is used to converge on the location of the zero.

In the interest of brevity, the reader is referred to the published reports by Shellman and Morfitt [18] and Goodhart and Pappert [12] for a comprehensive and interesting description of the technique developed by them.

For the technique presented in this paper only one form of the modal equation is used. To overcome the problem of overflow and underflow a scaling factor is introduced such that the modal equation is invariant to the changes in the scaling factor [6]. Since it is not necessary to use several pole-free forms of the equation, no extraneous phase contours are introduced nor does the possibility of losing a root arise due to the necessity to switch functions over the complex plane [12]. Approximate values for the roots of the characteristic equation need not be known a priori. The Newton–Raphson method is not used in any phase of the root finding technique described in this work; thus analytical or numerical derivatives of the modal equation are not needed.

The key to the root finding method described in this paper is the determination of the covering space (lines) on which all the complex roots lie. The covering space is readily determined for the nondissipative case (straight lines in the complex ν plane) and the complex roots can be found using standard numerical techniques developed for the real zero problem. This relatively simple problem is considered first in § 3(a). In § 3(b) the more general dissipative case is considered and the technique for tracking the covering space is described in detail. In § 4 a method is described to trace the loci of the

complex roots of the modal equations as one or more parameters of the modal equation varies and in § 5 an illustrative example is presented. It is interesting to note that in certain critical regions of the complex ν plane (corresponding, for instance, to the pseudo-Brewster angle), the attenuation $(-\mathrm{Im}\,(\nu^n))$ does not increase monotonically as n increases; thus a significant root may be outside the initial rectangular search mesh used in the technique developed by Goodhart and Pappert [12].

2. Formulation of the problem. In this work, the characteristic equation is cast in the form

$$F(\nu) = 1. \tag{2.1a}$$

The complex roots of the equation are denoted by the symbol ν^n and it is assumed that the characteristic equation may have either a finite or an infinite number of roots. The details of the procedure used to evaluate the roots of the characteristic equation (2.1a) are given in § 3 and the general procedure used to determine the locus of each root when a parameter of the modal equation is changed is presented in § 4.

Solely for the purpose of presenting an illustrative example in § 5, the problem of propagation of guided radio waves is considered here in detail. While this problem is of considerable practical interest to physicists and engineers it also helps illustrate, in a concrete manner, the difficult problems encountered in the evaluation of complex roots. The physical phenomena associated with the different complex roots in critical transition regions are also described in this paper; however, the reader need not be familiar with them to follow the general procedures presented in detail in §§ 3 and 4.

The modal equation for the guided waves in stratified media can generally be written as follows:

$$F(\nu) = R^U(\nu) R^D(\nu) = 1 \tag{2.1b}$$

in which the reflection coefficient $R^U(\nu)$ is the ratio of the fields of the downward and upward propagating waves at an arbitrary reference level when the incident wave is propagating upward. Similarly, the reflection coefficient $R^D(\nu)$ is the ratio of the fields of the upward and downward propagating waves at the arbitrary reference level when the incident wave is propagating downward. Thus, for horizontally stratified media $R^U(\nu)$ and $R^D(\nu)$ can be expressed in terms of the familiar Fresnel reflection coefficients and the exponential functions [2], [3], and for cylindrically and spherically stratified media the reflection coefficients are expressed in terms of the cylindrical and the spherical Bessel functions of complex argument and order ν, [4], [5]. Since the reference level at which the reflection coefficients are computed is arbitrary, the modal equation (2.1b) may be expressed in several different forms. In order to solve the modal equation for the dominant modes that contribute most significantly to the electromagnetic fields in the ith layer of the stratified structure, it is convenient to let the reference level be in the ith layer of the structure and (2.1b) is written as follows

$$F_i(\nu) = R_i^{Ur}(\nu) R_i^{Dr}(\nu) = 1. \tag{2.2}$$

Thus for example, if the transmitter and receiver are located in the uppermost layer $(i = 0)$, $R_0^U(\nu) = 0$ and (2.2) reduces to

$$1/R_0^{Dr}(\nu) = 0. \tag{2.3a}$$

Thus in this case the poles of the reflection coefficient $R_0^{Dr}(\nu)$ are to be found. For

horizontally and vertically polarized waves $R_0^D(\nu)$ can be expressed as

(2.3b)
$$R_{0H}^D(\nu)=\frac{z_0^H-1}{z_0^H+1}, \qquad R_{0V}^D(\nu)=\frac{y_0^V-1}{y_0^V+1}$$

in which the normalized impedance z_0^H is the ratio of the tangential components of the electric and magnetic fields for horizontally polarized waves at the reference level. Similarly y_0^V, the normalized wave admittance, is the ratio of the tangential components of the magnetic and electric fields for vertically polarized waves at the reference level. Thus, to find the poles of the reflection coefficient R_0^{Dr} the modal equation may also be written in the general form (2.1a)

(2.4)
$$-z_0^H(\nu)=1 \quad \text{or} \quad -y_0^V(\nu)=1.$$

In a multilayered model of the spherical earth-ionosphere waveguide which is considered here in detail as an illustrative example, the expressions for $R_i^{Ur}(\nu)$ and $R_i^{Dr}(\nu)$ are given by [5]

(2.5a)
$$R_i^{Dr}=R_i^D h_\nu^{(1)}(k_i r_{i,i+1})/h_\nu^{(2)}(k_i r_{i,i+1}), \qquad i=0, 1, \cdots, m-1$$

and for the innermost layer $i=m$

(2.5b)
$$R_m^{Dr}=1.$$

Similarly,

(2.6a)
$$R_i^{Ur}=R_i^U h_\nu^{(2)}(k_i r_{i-1,i})/h_\nu^{(1)}(k_i r_{i-1,i}), \qquad i=1, 2, \cdots, m$$

and for the outermost layer $i=0$

(2.6b)
$$R_0^{Ur}=0.$$

In (2.5) and (2.6) $h_\nu^{(P)}$ are the spherical Hankel functions of kind P $(P=1, 2)$, $k_i=\omega(\mu_i\varepsilon_i)^{1/2}$ is the wave number for the medium in layer i and $r=r_{i,i+1}$ is the interface between medium i and $i+1$. Furthermore,

(2.7a)
$$R_i^D=[R_{i+1,i}+R_{i+1}^{DH}(1+R_{i+1,i}+R_{i,i+1})]/[1-R_{i+1}^{DH}R_{i,i+1}], \qquad i=0, 1, 2, \cdots, m-1$$

and

(2.7b)
$$R_i^U=[R_{i-1,i}+R_{i-1}^{UH}(1+R_{i-1,i}+R_{i,i-1})]/[1-R_{i-1}^{UH}R_{i,i-1}], \qquad i=1, 2, 3, \cdots, m.$$

The two media reflection coefficients $R_{i,i+1}$ and $R_{i+1,i}$ for $i=0, 1 \cdots m-1$ are

(2.8a)
$$R_{i,i+1}=-\left.\frac{\eta_{i+1}\ln'[k_{i+1}rh_\nu^{(2)}(k_{i+1}r)]-\eta_i\ln'[k_i rh_\nu^{(2)}(k_i r)]}{\eta_{i+1}\ln'[k_{i+1}rh_\nu^{(1)}(k_{i+1}r)]-\eta_i\ln'[k_i rh_\nu^{(2)}(k_i r)]}\right|_{r=r_{i,i+1}},$$

(2.8b)
$$R_{i+1,i}=-\left.\frac{\eta_{i+1}\ln'[k_{i+1}rh_\nu^{(1)}(k_{i+1}r)]-\eta_i\ln'[k_i rh_\nu^{(1)}(k_i r)]}{\eta_{i+1}\ln'[k_{i+1}rh_\nu^{(1)}(k_{i+1}r)]-\eta_i\ln'[k_i rh_\nu^{(2)}(k_i r)]}\right|_{r=r_{i,i+1}}$$

in which $\eta_i=(\mu_i/\varepsilon_i)^{1/2}$ is the intrinsic impedance for the medium in the ith layer and $\ln' f(kr)\equiv[df/d(kr)]/f$. Thus as $k_i r_{i,i+1}\to\infty$, $R_{i,i+1}=-R_{i+1,i}$ reduce to the Fresnel reflection coefficients at the planar interface between two semi-infinite media. The reflection coefficients R_i^{DH} and R_i^{UH} are

(2.9a)
$$R_i^{DH}=R_i^{Dr}h_\nu^{(2)}(k_i r_{i-1,i})/h_\nu^{(1)}(k_i r_{i-1,i}), \qquad i=1, 2, \cdots, m,$$

and

(2.9b)
$$R_i^{UH}=R_i^{Ur}h_\nu^{(1)}(k_i r_{i,i+1})/h_\nu^{(2)}(k_i r_{i,i+1}), \qquad i=0, 1, \cdots, m-1.$$

The general form of the modal equation (2.1) is not restricted to layers with homogeneous media. The modal equation for stratified media with continuously varying permittivity profiles, such as dielectric waveguides, can also be cast in the form (2.1) [7].

As in the case of the modal equation for the earth-ionosphere waveguide, it is assumed here that: (a) Good initial estimates for the roots of the modal equation are not always known; (b) Closed form analytic expressions for the derivative $dF(\nu)/d\nu$ are not available and numerical differentiation of $F(\nu)$ is very sensitive to error because both the phase and the absolute value of $F(\nu)$ vary rapidly in critical regions of the complex ν plane.

In any realistic model of the earth ionosphere waveguide, for instance, it is necessary to account for lateral variation in the boundaries and in the electromagnetic parameters of the media. These variations result in mode coupling [5]. Thus in this work a method is presented to: (i) Determine ν^n, the roots of modal equation for all the dominant modes, assuming that no initial estimates of ν^n are given and that the total number of such roots is not known a priori; (ii) Determine the locus of each root ν^n as a function of frequency or as the electromagnetic parameters of the media and the boundaries vary laterally along the propagation path.

3. Loci of $|F(\nu)| = 1$ and the roots of the modal equation at a fixed cross section and frequency

(a) Nondissipative media with perfectly reflecting boundaries. Consider first a spherical waveguide consisting of a nondissipative medium (μ_0, ε_0) bounded by perfectly reflecting concentric spherical boundaries at $r = r_{0,1}$ and $r = r_{1,2}$ respectively. In this case it is relatively easy to determine the roots ν^n of the modal equation (2.1). For the propagating modes ν^n is real and lies between $\nu = k_1 r_{0,1}$ and $\nu = -\frac{1}{2}$ and for the evanescent modes ν^n lies on a line parallel to the imaginary axis, $\nu = -\frac{1}{2} + i\alpha$ with $0 > \alpha > -\infty$. In the corresponding cylindrical problem ν^n lies on the negative imaginary axis for the evanescent modes [8].

The propagating modes are also classified as the whispering gallery modes or the earth-detached modes for $k_1 r_{1,2} < \nu_n < k_1 r_{0,1}$ [19], [20], [21] and the regular earth-ionosphere waveguide modes (that are reflected off both the earth and ionosphere boundaries) for $-\frac{1}{2} < \nu^n < k_1 r_{1,2}$. Along the real axis between $\nu = k_1 r_{0,1}$ and $\nu = -\frac{1}{2}$ and along the line $\nu = -\frac{1}{2} + i\alpha$, $0 > \alpha > -\infty$ the magnitude

$$|F(\nu)| = M(\nu) = 1. \tag{3.1a}$$

Several standard numerical methods developed for the real zero problem can be used to determine the roots of the modal equation at points on these lines where the phase angle

$$\arg [F(\nu)] = \phi(\nu) = 0 \tag{3.1b}$$

or $\phi = 2\pi m$ $(m = 0, \pm 1, \pm 2, \cdots)$, [23]. Since ν is the order of the spherical Hankel functions, the expression for $F'(\nu) = dF/d\nu$ cannot be written in a closed analytical form [1] and numerical differentiation is subject to significant errors in critical regions. In this work the interval halving technique (bisection method [10]) is used to locate the roots ν^n on the locus $M(\nu) = 1$, (3.1a).

(b) Dissipative media with nonperfectly reflecting boundaries. For the general case in which the medium and the boundaries of the waveguide are dissipative all the waveguide modes attenuate along the propagation path. The roots ν^n are complex for the dissipative case and the loci of the points satisfying $M(\nu) = 1$, (3.1a), must be determined numerically in order to use this method to solve the modal equation.

Since the function $F(\nu)$ is analytical along the loci $M(\nu)=1$, it follows from the Cauchy–Riemann conditions that

$$\frac{1}{M}\frac{\partial M}{\partial \nu_R}=\frac{\partial \phi}{\partial \nu_I} \quad \text{and} \quad \frac{1}{M}\frac{\partial M}{\partial \nu_I}=-\frac{\partial \phi}{\partial \nu_R}, \tag{3.2a}$$

where

$$\nu=\nu_R+i\nu_I. \tag{3.2b}$$

Thus

$$\frac{1}{M}\operatorname{grad} M\cdot\operatorname{grad}\phi=\frac{1}{M}\left(\frac{\partial M}{\partial \nu_R}\frac{\partial \phi}{\partial \nu_R}+\frac{\partial M}{\partial \nu_I}\frac{\partial \phi}{\partial \nu_I}\right)=0 \tag{3.3}$$

and the constant ϕ contours (including $\phi=0$) cut the locus $M=1$ at right angles in the complex ν plane. The conformal mapping properties of analytical functions of complex variables are exploited throughout this work. In the terminology of algebraic topology the mapping, $M(\nu)=1$, defines in the ν plane the "covering space" of the unit circle, $|F(\nu)|=1$, [17].

To trace the locus $M(\nu)=1$ for the general dissipative case the following method is used (see Fig. 1). Assuming that the modes contributing significantly to propagation in the region between the earth and the ionosphere are of interest, only those roots for which $\operatorname{Re}(\nu_n)<k_1 r_{0,1}$ are sought. Thus beginning with the point $\nu=k_1 r_{0,1}$, and using the one-dimensional interval halving method, the imaginary part of ν is varied until

$$|M(\nu_p)-1|<\delta, \qquad p=1,2,3\cdots, \tag{3.4}$$

and the first point ν_1 (not necessarily a root) is found on the locus $M=1$ (δ is typically 10^{-2} to 10^{-3}). This process is repeated once more beginning with the point $\nu_1-\Delta$ (where

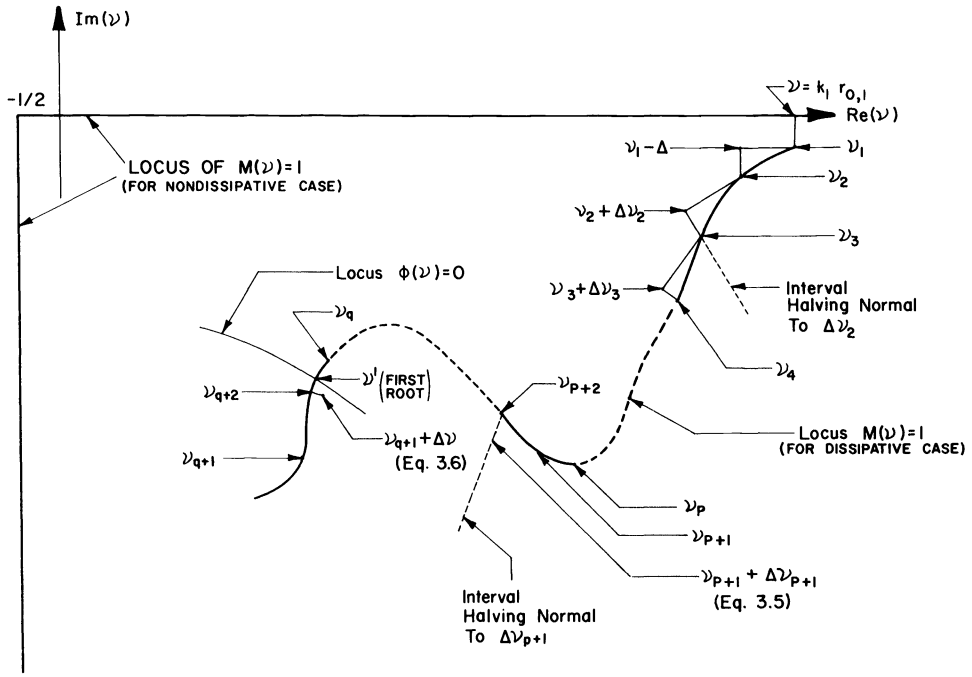


FIG. 1. *Locus of* $M(\nu)=1$ *and the roots of* $F(\nu)=1$.

Δ is a positive increment) and the interval halving procedure is followed parallel to the imaginary axis. The size of Δ is adjusted such that the difference in the phase $\phi(\nu)$ for the two adjacent points on $M=1$, ν_1 and ν_2, is about 0.2 radians.

The remaining points on the locus $M=1$ are determined as follows (see Fig. 1). Let ν_p and ν_{p+1} be the last two points found on the locus $M=1$ (satisfying 3.4). The initial value assumed for the next point, ν_{p+2}, is

$$\nu=\nu_{p+1}+\Delta\nu_{p+1}=\nu_{p+1}+\nu_{p+1}-\nu_p=2\nu_{p+1}-\nu_p. \tag{3.5}$$

The point ν_{p+2} on the locus $M=1$ is obtained through the interval halving process. However, this time instead of phasor increments parallel to the imaginary axis (as in the cases $p=1$ and 2), the phasor increments are taken parallel to $i\Delta\nu_{p+1}$ since these increments are approximately perpendicular to the locus $M=1$.

This process is repeated until the phasor $F(\nu_{q+1})$ crosses the positive real axis. Let ν_q and ν_{q+1} correspond to the points on $M(\nu)=1$ such that $\phi(\nu_q)>0$ and $\phi(\nu_{q+1})<0$. Obviously the root of the modal equation ν^1 lies on the locus $M(\nu)=1$ between the points ν_q and ν_{q+1}. Since $F'(\nu)$ cannot be written in closed analytical form, the secant method is used to locate the root [10] (see Fig. 1). Thus the initial value assumed for the next point, ν_{q+2}, on the locus $M=1$ is obtained through linear interpolation:

$$\nu=\nu_{q+1}+\frac{0-\phi_{q+1}}{\phi_q-\phi_{q+1}}(\nu_q-\nu_{q+1})\equiv\nu_{q+1}-\frac{\phi_{q+1}}{\phi_q-\phi_{q+1}}\Delta\nu_{q+1}. \tag{3.6}$$

The interval halving procedure is carried out parallel to the phasor $i\Delta\nu_{q+1}$ that is approximately perpendicular to the locus $M=1$ until the new point ν_{q+2} is located on this locus. This procedure of linear interpolation in ϕ and interval halving is repeated until the first root is located on the locus $M=1$ such that

$$|F(\nu^1)-1|<\delta_0, \tag{3.7}$$

where δ_0 is typically 10^{-3}. Condition (3.7) for the root ν^1 should not be confused with condition (3.4) for points ν_p along the locus $M=1$ which do not necessarily satisfy (3.7) since $-\pi<\phi_p\leqq\pi$.

Beginning with ν^1 the procedure described above is repeated again to find new points on the locus $M(\nu)=1$, until $F(\nu)$ again crosses the positive real axis. The next root ν^2 is located through the process of linear interpolation (3.6) parallel to the locus $M(\nu)=1$ and interval having perpendicular to $M(\nu)=1$. It should be pointed out, however, that any one of the numerical techniques developed for the real zero problem could also be used instead of linear interpolation and interval halving [11].

All the roots of the modal equation (2.1) $\nu^1,\cdots,\nu^N$ are found in this manner for the modes that lie on the principal branch of (3.1a) and contribute significantly to the particular propagation problems under consideration. The number N depends on the excitation, type of irregular waveguide (that results in mode coupling) and on the distances along the propagation path. Usually the magnitude of the imaginary part of ν^N (associated with wave attenuation) is the principal factor that determines the number of modes N. It should be pointed out, however, that the magnitude of ν_I^n does not necessarily increase monotonically as the number n associated with the nth root found on the locus $M(\nu)=1$, increases.

In the nondissipative case, for instance, the locus $M(\nu)=1$, (3.1a) has only one branch in the lower half complex ν plane (see § 3a). However, for the dissipative case it may be necessary to consider, besides the principal branch of the locus $M(\nu)=1$, additional closed branches around the zeros or poles of $F(\nu)$, (2.1). Thus for instance if

$F(\nu)$ has a simple zero at ν_0, a single root of the modal equation (2.1) can be found where the $\phi = 0$ contour intersects the closed branch of $M(\nu) = 1$ around ν_0 [9], [15]. For the propagation problem considered here, as an illustrative example, such an isolated branch of $M(\nu) = 1$ is associated with the pseudo-Brewster angle. If $|\text{Im}(\nu_0)| \gg k_1 r_{1,2}$ the contribution from the mode on this branch may be ignored.

4. Loci of the roots ν^n of the modal (characteristic) equation. For irregular waveguide structures in which the boundaries and the electromagnetic parameters (μ, ε) vary along the propagation path, it is necessary to determine the locus of each of the roots ν^n along the propagation path in order to solve the differential equations for the coupled mode amplitudes [5]. In the case of waveguide bends with varying curvature, for instance, it is necessary to compute the value of ν^n as a function of curvature [8]. Similarly, to determine the transmission of transient signals through a uniform waveguide, it is necessary to trace the loci of ν^n as a function of frequency ω.

For the case of nondissipative media with perfectly reflecting boundaries, the locus of each of the roots ν^n is along the positive real axis for the propagating modes and parallel to the negative imaginary axis for the evanescent modes. In these cases one of the standard methods can be readily used to track the locus of each root ν^n as a given parameter or parameters vary along the propagation path [11].

In this work, dissipative media with non-perfectly reflecting boundaries are considered. In this case the direction of the locus of ν^n is not parallel to the real or imaginary axes. Thus it is more difficult to trace the loci of ν^n in the complex ν plane. Because of the reasons stated in § 3, the following procedure was found to be suitable to track the locus of the root ν^n (see Fig. 2).

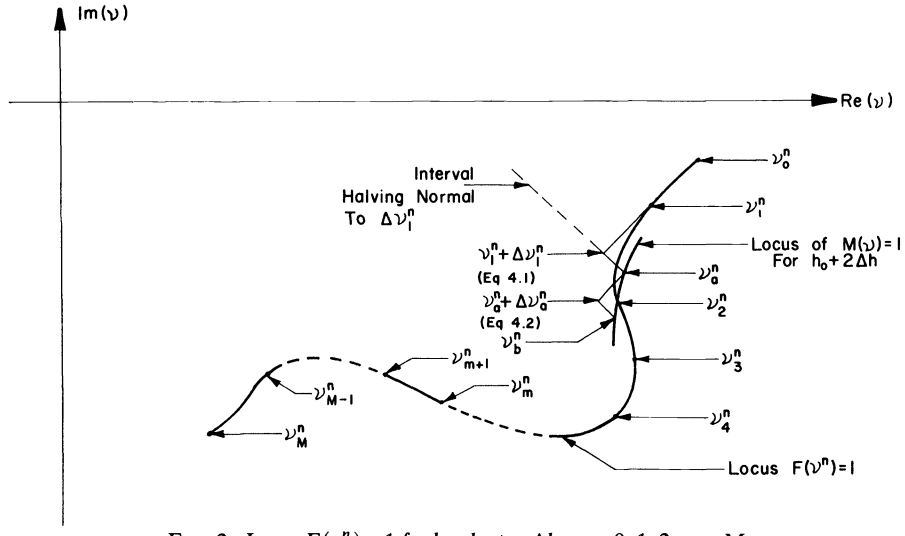


FIG. 2. *Locus* $F(\nu^n) = 1$ *for* $h = h_0 + m\Delta h$, $m = 0, 1, 2, \cdots, M$.

Assuming that the variable parameter along the propagation path is the height of the waveguide h, a set of roots of the modal equation is computed for each of the heights $h = h_0$ and $h = h_0 + \Delta h$ using the technique described in § 3. These sets of roots are denoted by the symbols ν_0^n and ν_1^n, respectively $(n = 1, 2, \cdots, N)$. The procedure in § 3 can, of course, also be repeated to determine the set of roots ν_2^n for $h = h_0 + 2\Delta h$. Instead, however, the following procedure is found to be more efficient.

The initial value assumed for ν_2^n is

$$\nu = \nu_1^n + \frac{\Delta\nu_1^n}{\Delta h}\Delta h \equiv \nu_1^n + \frac{\nu_1^n - \nu_0^n}{\Delta h}\Delta h \tag{4.1}$$

and the interval halving procedure is carried out parallel to the phasor $i\Delta\nu_1$ until a new point, ν_a^n, on the locus $M(\nu) = 1$ is located for $h = h_0 + 2\Delta h$. This search normal to the phasor $\Delta\nu_1$ is needed to account for curvature of the locus ν^n. Another point, ν_b^n, on the locus $M(\nu) = 1$ is found in a similar way using

$$\nu = \nu_a^n + \Delta\nu_a^n = \nu_a^n + \frac{\Delta\nu_1^n}{\Delta h}\frac{\Delta h}{4} \tag{4.2}$$

as the assumed initial value. With these two points ν_a^n and ν_b^n on the locus $M = 1$ the procedure of linear interpolation in phase (3.6) and interval halving described in § 3 can be used to locate the complex root ν_2^n corresponding to $h = h_0 + 2\Delta h$.

This locus tracing technique for the roots ν_m^n of the modal equation (2.1) is performed for

$$h = h_0 + m\Delta h = h_m, \tag{4.3}$$

where

$$m = 2, 3, 4, \cdots, M \quad \text{and} \quad \Delta h = (h_M - h_0)/M.$$

This scheme can also be used if any other parameter of the waveguide is perturbed along the propagation path or if it is necessary to determine the locus of the root ν^n for a range of frequencies.

5. Illustrative example. To illustrate the root finding procedures described in this paper, the modal equation for vertically polarized waves in an irregular model of the earth-ionosphere waveguide is considered here [5], [6]. For simplicity a three layer model is assumed with

$$\varepsilon = \begin{cases} \varepsilon_0 = \varepsilon_f(1-i), & \text{ionosphere} \\ \varepsilon_1 = \varepsilon_f, & \text{air} \\ \varepsilon_2 = \varepsilon_f(10 - il0), & \text{earth} \end{cases}, \qquad \mu = \mu_i = \mu_f \quad (i = 0, 1, 2) \tag{5.1}$$

in which ε_f and μ_f are the permittivity and permeability for free space. The effective height of the ionosphere h is assumed to vary between $h_M = 60$ km and $h_0 = 90$ km, thus the effective ionosphere boundary is given by

$$r = r_{0,1} = r_{1,2} + h, \qquad h_M \leqq h \leqq h_0 \tag{5.2}$$

in which $r_{1,2} = 6.4 \times 10^3$ km is the radius of the earth. At the frequency,

$$f = 15 \text{ kHz}, \tag{5.3}$$

assumed in this illustrative example, both the earth-detached modes as well as the earth-ionosphere waveguide modes that are reflected off the ionosphere and earth, contribute significantly to radio wave propagation.

In Fig. 3 the search for points on the locus $M(\nu) = 1$ in the complex ν plane is illustrated for both $h_0 = 90$ km and for $h_1 = h_0 + \Delta h$, where $\Delta h = (h_M - h_0)/20 = 1.5$ km. Beginning with the point $\nu = k_1 r_{0,1}$ the interval halving procedure is used to locate the first point ν_1 on the locus $M(\nu) = 1$. This procedure is repeated beginning with $\nu_1 - \Delta$ (see § 3) and the second point ν_2 on the locus $M(\nu) = 1$ is located. From here on the initial value assumed for ν_{p+1} is $\nu = \nu_{p+1} + \Delta\nu_{p+1}$ (3.5), where $\Delta\nu_{p+1} = \nu_{p+1} - \nu_p$ is approximately in the direction tangent to the locus $M(\nu) = 1$ at ν_{p+1}. Note that for the

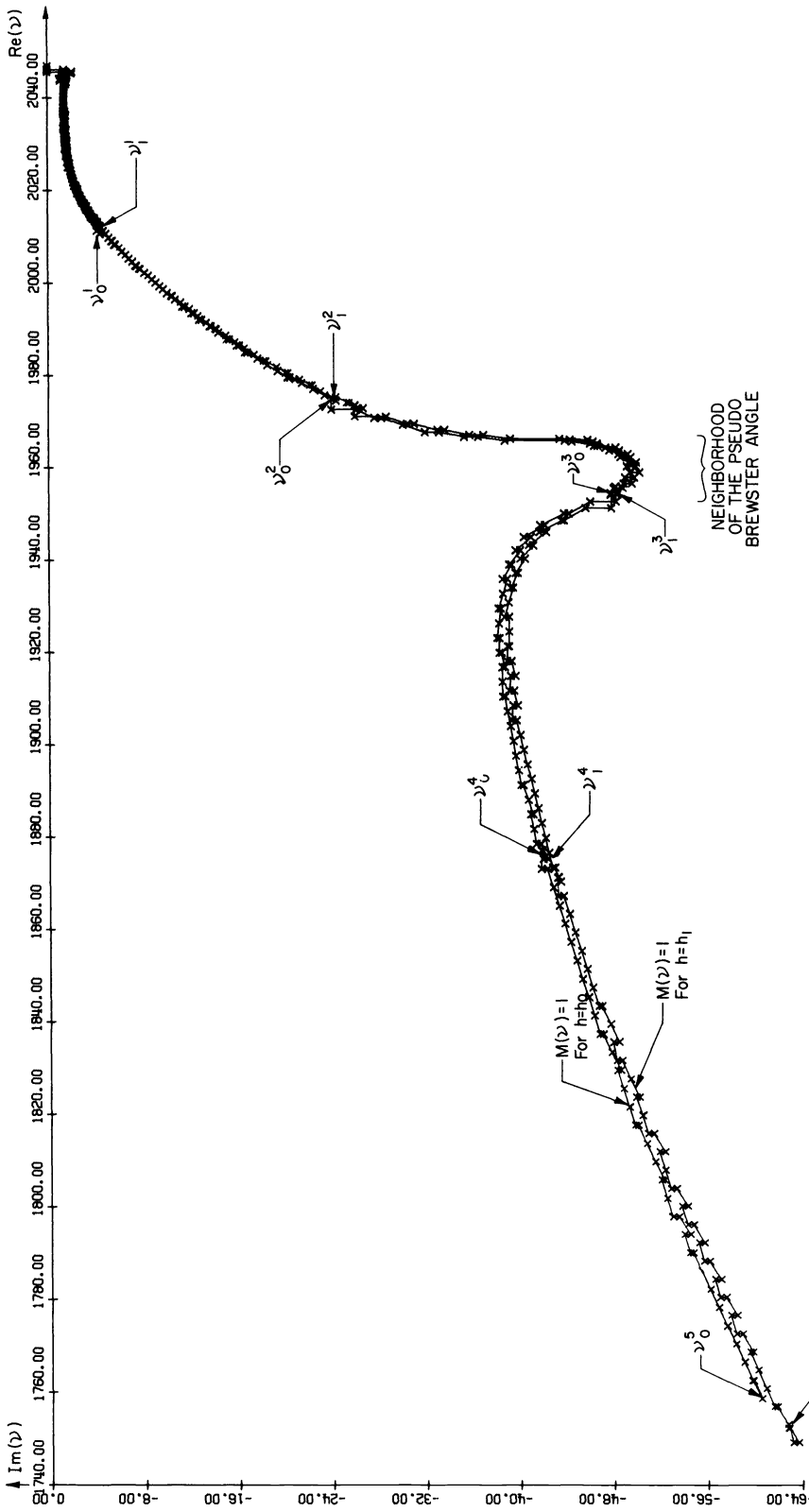


FIG. 3. *The first five roots of the modal equation* $F(\nu)=1$ *for* $h=h_B$ *and for* $h=h_B+\Delta h$, $\Delta h=(h_B-h_A)/20$.

purpose of the illustration, both the initial value assumed for ν near the locus, (3.5), as well as the points on the locus satisfying (3.4) are shown on Fig. 3. The interval halving procedure used to advance from the initial value ν (3.5) to the point ν_{p+2} on the locus uses increments in the direction of the phasors $i\Delta\nu_{p+1}$. Thus the phasors joining the initial values assumed for ν (3.5) and the corresponding points on the locus ν_{p+2} are approximately normal to the locus $M(\nu)=1$. The smaller the curvature of the locus $M(\nu)=1$ the smaller the distance between the assumed initial value for ν and the corresponding point on the locus ν_{p+2}.

Once the phasor $F(\nu)$ crosses the positive real axis ($\phi(\nu)=0$), the linear phase interpolation procedure (3.6) and the interval halving procedure are used to locate the root ν^1 of the modal equation satisfying (3.7) (see Fig. 3). To locate the rest of the roots (ν^2, ν^3, ν^4 and ν^5), these procedures are repeated. The distances between the roots ν_0^n for h_0 and the roots ν_1^n for $h_1=h_0+\Delta h$ becomes progressively larger as the mode number increases.

Since the first value assumed for ν is on the real axis ($\nu=k_0, r_{0,1}$), it is preferable to begin the root search for $h_0=h_{\max}$ where $|\mathrm{Im}\,(\nu^1)|$ is smallest.

Mode number 1 is an earth-detached mode, while modes 2 and 3 on both sides of the trough in the locus $M(\nu)=1$ are modes of near grazing incidence at the earth's surface. It is interesting to note that the attenuation associated with mode number 3($-\mathrm{Im}\,(\nu^3)$) is larger than the attenuation for mode number 4. The trough in the locus of $M(\nu)=1$ (near grazing incidence at the earth's surface) is associated with the minimum in the magnitude of the reflection coefficient for vertically polarized waves near grazing incidence. This minimum in the magnitude of the reflection coefficient for vertically polarized waves (near grazing incidence at the earth's surface) is associated with the pseudo-Brewster angle [16]. Thus the relatively large attenuation associated with mode number 3 is related to the pseudo-Brewster angle phenomenon which is not present for horizontally polarized waves. In the neighborhood of the pseudo-Brewster angle the function $F(\nu)$ varies very rapidly and in this transition region it is very difficult to compute $F'(\nu)$.

In Fig. 4, the phasor $F(\nu)$ is plotted for points near the locus $M(v)=1$ from $\nu=k_1 r_{0,1}$ to the point ν^1 (the first root of the modal equation). As indicated in Figs. 1

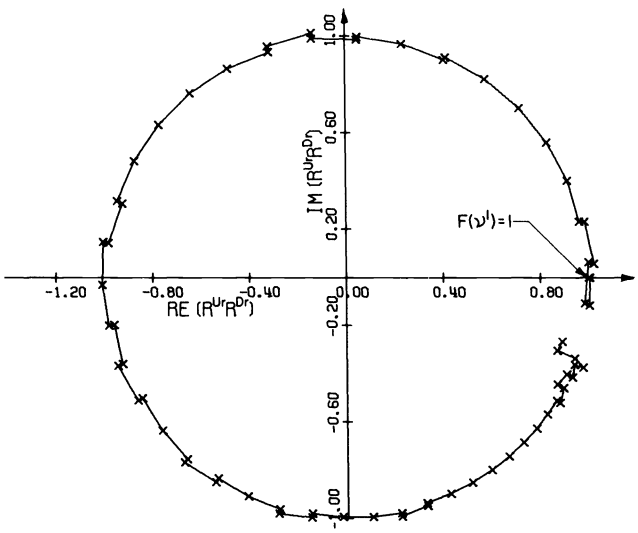


FIG. 4. *Search for the first root shown in the* $F(\nu)$-plane.

and 3, not all these points are on the locus $M(\nu) = |F(\nu)| = 1$. It should be observed that the linear phase interpolation procedure (3.6) is triggered when $F(\nu)$ crosses the positive real axis. For the illustrative example, more intermediate points on the locus $M(\nu) = 1$ are shown than are required for the numerical evaluation of the roots. In Fig. 5, the phasor $F(\nu)$ is plotted as the search for the root ν^2 proceeds from the previously found root ν^1. The direction of search in Figs. 4 and 5 is counterclockwise.

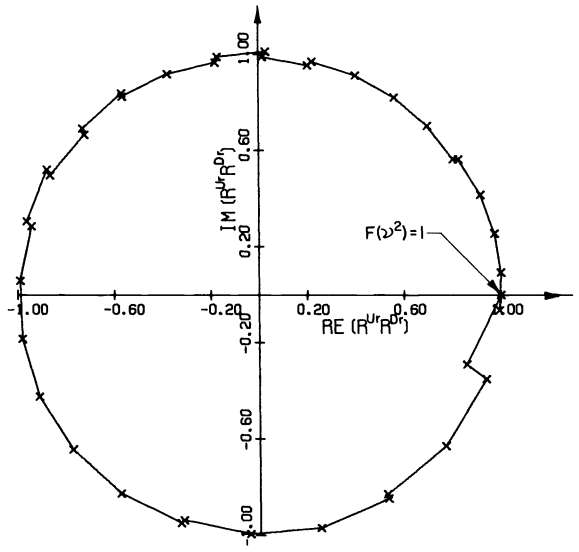


FIG. 5. *Search for the second root shown in the $F(\nu)$ plane.*

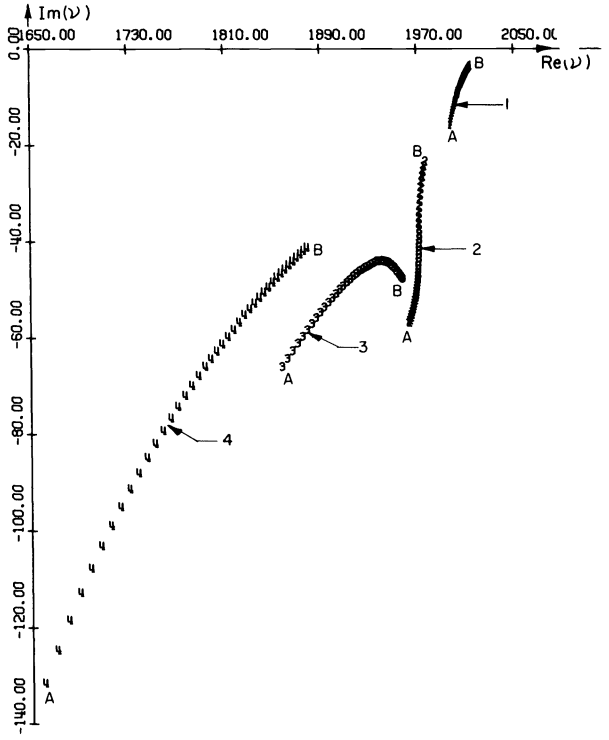


FIG. 6. *Loci of the roots ν^n for $n = 1, 2, 3, 4$ as h varies from 60 km to 90 km.*

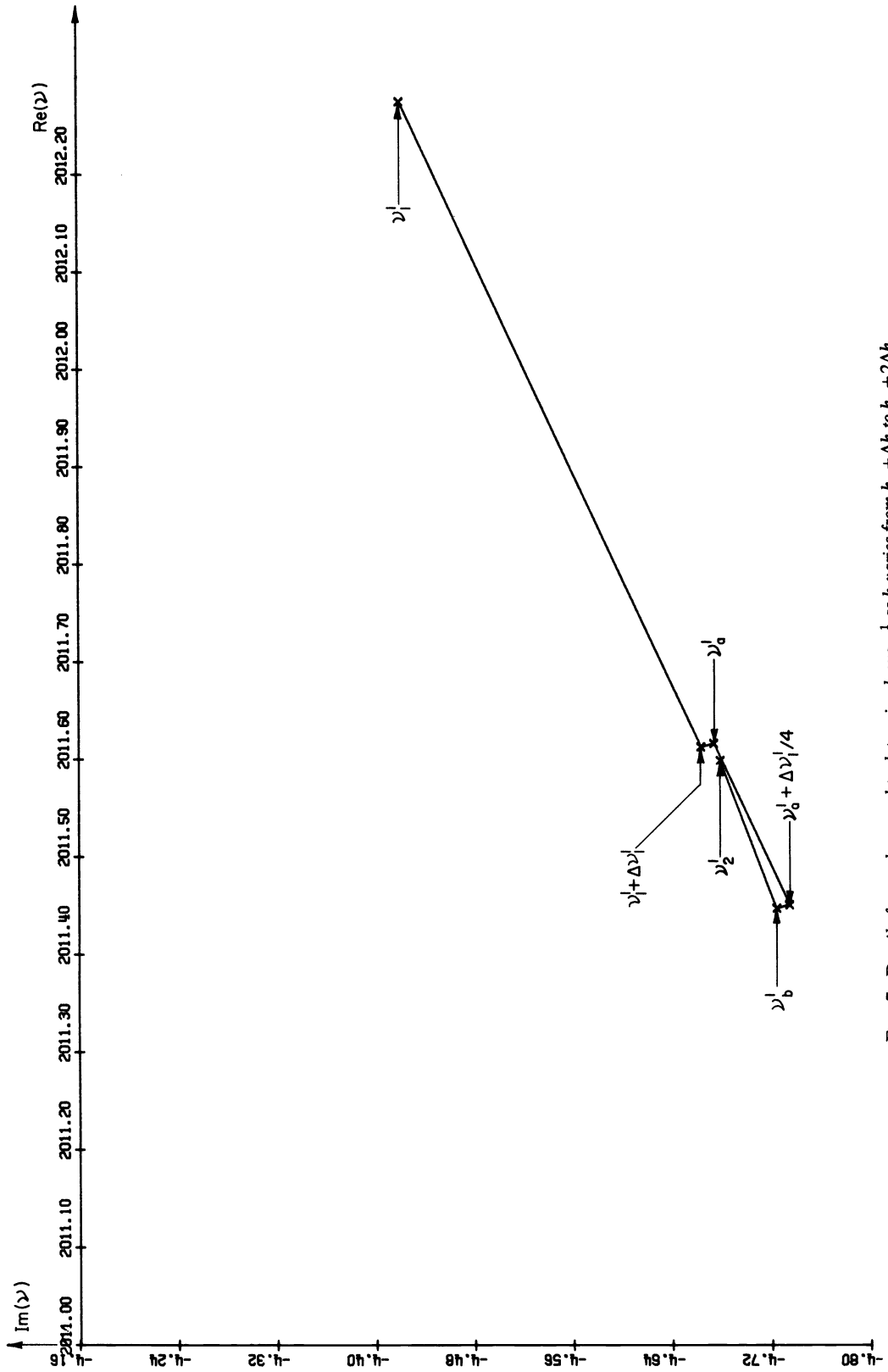


FIG. 7. *Detail of procedure used to determine locus* ν^1 *as* h *varies from* $h_0+\Delta h$ *to* $h_0+2\Delta h$.

In Fig. 6 the loci of the roots ν^n are shown for $n = 1, 2, 3, 4$ with h varying from $h_0 = h_B$ to $h_M = h_A$. The procedure for tracing the locus ν^n is described in § 4. Note the peculiar behavior of the locus for mode number 3 which is associated with waves near the pseudo-Brewster angle.

The steps in the procedure for finding the point ν_2^1 on the locus ν^1 are illustrated in Fig. 7. As described in § 4, the intermediate points ν_a^1 and ν_b^1 are located on the locus $M(\nu) = 1$ for $h = h_0 - 2\Delta h$. Thus they are not on the locus for $\nu^1 (F(\nu^1) = 1)$.

6. Concluding remarks. A technique is presented for the solution of the complex roots of characteristic equations appearing in a large variety of problems in mathematical physics. In addition, the method can be used to trace the loci of these roots as one or more parameters of the modal equation are varied. It is not assumed, in this work, that approximate values for the roots are known, nor is the number of roots to be determined known a priori.

The presence of poles, in the analytical expression for the modal equation, does not have a significant bearing on the method presented and only one form of the modal equation is used even though the functions appearing in the equation fluctuate over several hundred orders of magnitude.

For the illustrative example presented, an irregular model of the earth-ionosphere waveguide is considered. It is shown that in the critical transition region (corresponding to waves at the pseudo-Brewster angle), there is a sharp increase in the attenuation factor $(-\text{Im}(\nu^n))$ [22]. For good conducting ground this occurs for vertically polarized waves near grazing incidence. At 15 kHz (assumed in the example), both the earth-detached modes as well as the regular earth-ionosphere waveguide modes (that are reflected off the earth and ionosphere boundaries) are present. The method presented in this paper tracks all these different kinds of modes in a systematic manner. It also provides a straightforward scheme for the numbering and the classification of the different modes of the system.

REFERENCES

[1] M. ABRAMOWITZ AND I. A. STEGUN, *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, Appl Math. Ser. 55, National Bureau of Standards, Washington, DC, 1964.

[2] E. BAHAR, *Electromagnetic wave propagation in inhomogeneous multilayered structures of arbitrarily varying thickness—generalized field transforms*, J. Math. Phys., 14 (1973), pp. 1024–1029.

[3] ———, *Depolarization of electromagnetic waves excited by distributions of electric and magnetic sources in inhomogeneous multilayered structures of arbitrarily varying thickness*, J. Math. Phys., 14 (1973), pp. 1502–1509.

[4] ———, *Field transforms for multilayered cylindrical and spherical structures on finite conductivity*, Canad. J. Phys., 53 (1975), pp. 1078–1087.

[5] ———, *Electromagnetic waves in irregular multilayered spherical structures of finite conductivity—full wave solutions*, Radio Sci., 11 (1976), pp. 137–147.

[6] ———, *Computations of the transmission and reflection coefficients in an irregular spheroidal model of the earth ionosphere waveguide*, Radio Sci., 15 (1980), pp. 987–1000.

[7] E. BAHAR AND B. S. AGRAWAL, *Application of generalized characteristic vectors to problems of propagation in clad inhomogeneous dielectric waveguides*, IEEE Trans. Antennas and Propagation, 27 (1979), pp. 345–352.

[8] E. BAHAR AND G. GOVINDARAJAN, *Rectangular and annular modal analysis of multimode waveguide bends*, IEEE Trans. Microwave Theory Tech., MTT-21 (1973), pp. 819–824.

[9] K. G. BUDDEN, *The Wave-Guide Mode Theory of Wave Propagation*, Prentice-Hall, Englewood Cliffs, NJ, 1961.

[10] S. D. CONTE, *Elementary Numerical Analysis*, McGraw-Hill, New York, 1965.

[11] G. E. FORSYTHE, M. A. MALCOLM AND C. B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[12] C. L. GOODHART AND R. A. PAPPERT, *Application of a root finding method for tropospheric ducting produced by trilinear refractivity profiles*, Tech. Rep., Naval Ocean Systems Center, San Diego, CA, 1977.

[13] R. W. HAMMING, *Numerical Methods for Scientists and Engineers*, 2nd ed., McGraw-Hill, New York, 1973.

[14] P. HENRICI, *Applied and Computational Complex Analysis*, John Wiley, New York, 1974.

[15] H. H. HOWE AND J. R. WAIT, *Mode calculations for VLF ionospheric propagation*, National Bureau of Standards, (USA) VLF Symposium Paper 36, Boulder, Colorado, Jan. 1957.

[16] E. C. JORDAN AND BALMAIN, *Electromagnetic Waves and Radiating Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1968.

[17] W. S. MASSEY, *Algebraic Topology, An Introduction*, Springer-Verlag, New York, 1977.

[18] D. G. MORFITT AND C. H. SHELLMAN, *MODESRCH: an improved computer program for obtaining ELF/VLF/LF mode constants in an earth ionosphere waveguide*, Naval Electronics Laboratory Center Interim Rep. 77T, for Defense Nuclear Agency, October, 1976.

[19] LORD RAYLEIGH, *The problem of the whispering gallery*, Phil. Mag., 20 (1910), pp. 1001–1004.

[20] ———, *Further applications of the Bessel functions of high order to the whispering gallery and allied problems*, Phil. Mag., 27 (1914), pp. 100–109.

[21] J. R. WAIT, *Electromagnetic Waves in Stratified Media*, Pergamon Press, New York, 1962.

[22] S. WESTERLUND AND F. H. REDER, *VLF radio signals propagating over the Greenland ice-sheet*, J. Atmospheric and Terrestrial Phys., 35 (1973), pp. 1475–1491.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0004 $01.00/0

MULTIPLE SOLUTIONS AND BIFURCATION OF FINITE DIFFERENCE APPROXIMATIONS TO SOME STEADY PROBLEMS OF FLUID DYNAMICS*

A. B. STEPHENS† AND G. R. SHUBIN†

Abstract. We review and extend an earlier study of the behavior of multiple finite difference solutions for a centered difference approximation of the steady Burgers' equation. Using the fact that all of the inviscid (viscosity = 0) solutions can be found, we numerically continue these solutions with respect to viscosity and thereby uncover turning points and bifurcation points. In addition, we demonstrate analogous behavior for a model of one-dimensional duct flow and for a particular discretization of the supersonic blunt body problem.

Key words. Burgers' equation, multiple solutions, bifurcation, continuation, fluid dynamics, finite differences

1. Introduction. In computational fluid dynamics, finite difference equations furnish a discrete approximation to the differential equations which are a continuous description of the flow field. It is usually assumed that a numerical solution of these difference equations will, to some reasonable degree of accuracy, actually approximate the flow field. However, since the governing equations for the continuous problem are nonlinear, it is ordinarily quite difficult to establish the existence and uniqueness of finite difference solutions. In fact, for some model problems, certain very reasonable difference schemes can yield multiple (real) solutions even when the continuous problem being approximated is known to have a unique solution.

For a practical problem, when a single solution is obtained we usually cannot tell whether other solutions exist. In general, there is no algorithm for determining all of the solutions of a nonlinear system. However, under certain circumstances, if one solution is known others may be determined [3]. If several solutions are found, it may be very difficult to select the best approximation. This is especially true if the solutions are "close together." If the solutions are far apart, we may try to compare the solutions with experimental data to identify the physically relevant one(s). We may also look for properties that the discrete solutions should inherit from the continuous problem. Another approach might be to refine the computational mesh, assuming the convergence of one of the difference solutions to the correct solution of the differential equations in the limit. However, we generally have no guarantee that the "spurious" solutions will go away in this limit, nor would we be able in practice to obtain a sufficiently small mesh size.

These practical questions can be interpreted in a more theoretical setting. Suppose that a given steady state fluid dynamics problem has been formulated as a system of ordinary or partial differential equations subject to certain boundary conditions. We assume (though this is usually not clear) that this system is well posed and provides an adequate description of the physics. This system is then discretized consistently to yield a system of difference equations. We restrict ourselves here to real-valued solutions of these equations since they are the ones that may be obtained with the usual computational schemes and are the only ones with physical meaning. The following questions arise:

(1) Does this system of difference equations possess no solution, a unique solution,

* Received by the editors November 13, 1980. This work was supported by the Naval Surface Weapons Center Independent Research Fund.

† Applied Mathematics Branch, Naval Surface Weapons Center, Silver Spring, Maryland 20910.

or multiple solutions? If multiple solutions exist, which one provides the best approximation and how can it be found and identified?

(2) How do the solutions depend on physical parameters (e.g., viscosity) and discretization parameters (e.g., finite difference mesh size)? Are there bifurcation or turning points? How do the solutions behave in the limit of vanishing mesh size?

(3) Which solutions to these steady difference equations are obtainable via some time-dependent or iterative technique (i.e., which are "stable" for some technique)?

Alternately, assuming a difference scheme has not been selected, we may further ask

(4) Is there some consistent difference approximation which possesses a unique solution?

For anything but the simplest model problems, these questions are very difficult to answer. With the steady Burgers equation as a model, Kellogg, Shubin, and Stephens [9] investigated multiplicity of solutions for three finite difference approximations. Schreiber and Keller [18] have found multiple solutions for the driven cavity problem. For certain ODE boundary value problems, Beyn and Doedel [4], Allgower [1], and Peitgen, Saupe, and Schmitt [15] have investigated multiple continuous and discrete solutions. Osher [14] has designed a scheme which possesses a unique solution for an ODE singular perturbation problem. Some work has also been done for inviscid fluid dynamics problems which possess multiple (weak) solutions to the continuous problem and require an entropy condition to select the correct physical approximation. Harten, Hyman, and Lax [7] have shown that, for a single conservation law, monotone schemes will converge to the entropy-satisfying solution while solutions obtained with the Lax–Wendroff scheme may not. These kinds of multiple solutions have also been found for some approximations of the small disturbance transonic equation by Stephens and Werschulz [21], while Engquist and Osher [6] have designed a scheme which converges only to entropy-satisfying solutions. Shubin, Stephens, and Glaz [19] found multiple solutions for one-dimensional inviscid duct flow.

In the present paper we review the results of [9] and extend them to study the behavior of multiple finite difference solutions for a centered difference approximation of Burgers' equation when the viscosity (or cell Reynolds number) is varied. This extension is partly analytic and partly numerical. We use the fact that all of the inviscid (viscosity $\nu = 0$) finite difference solutions can be explicitly found, and use a numerical continuation procedure to advance these solutions to positive values of ν and thereby uncover turning points and bifurcation points. As shown in [9], for sufficiently large ν (i.e., when a cell Reynolds number condition is satisfied) there is a unique solution in a certain domain Ω and all other solutions either become complex-valued or lie outside Ω. Some properties of these difference solutions are investigated. In addition to Burgers' equation, we look at a model of viscous one-dimensional duct flow and use the M-function approach of [9] and numerical continuation to study the discrete solutions. We also show some analogous computational results for a particular discretization of the inviscid axisymmetric blunt body problem [19] when artificial viscosity is added.

2. Burgers' equation

2A. The continuous and discrete problems. We are interested in this section in the steady Burgers equation

$$uu_x - \nu u_{xx} = 0, \qquad \nu > 0 \tag{2.1}$$

with boundary conditions $u(0) = 1$, $u(1) = -1$, whose solution is given by $u(x) = -k \tanh(k(x-0.5)/(2\nu))$ where the constant k satisfies $1 = k \tanh(k/4\nu)$. The time-dependent equation $u_t + uu_x = \nu u_{xx}$ was introduced by J. M. Burgers as a simple model

for the more complicated differential equations of fluid flow. The terms uu_x and νu_{xx} model convection and diffusion respectively.

Let $x_i = ih$, $0 \leqq i \leqq n+1$, $h = 1/(n+1)$ be uniformly spaced mesh points and let u_i represent an approximation to $u(x_i)$. We consider the following centered difference approximation to (2.1)

$$u_{i+1}^2 - u_{i-1}^2 - \theta(u_{i+1} - 2u_i + u_{i-1}) = 0, \qquad \theta = \frac{4\nu}{h} \tag{2.2}$$

where $1 \leqq i \leqq n$ and we set $u_0 = u(0) = 1$ and $u_{n+1} = u(1) = -1$. For fixed h we write the nonlinear system symbolically as $F(\mathbf{u}(\nu), \nu) = 0$.

2B. Review of previous results. We now briefly discuss the results of [9] concerning multiple solutions to various difference approximations (and more general boundary conditions) of which (2.2) is a particular case. When a scheme analogous to (2.2)

$$\frac{c(u_{i+1} - u_{i-1})}{2h} - \frac{\nu(u_{i+1} - 2u_i + u_{i-1})}{h^2} = 0$$

is applied to the linearized equation $cu_x - \nu u_{xx} = 0$, the condition $ch/\nu < 2$ is required for a nonoscillatory solution. This restriction is called a *cell Reynolds number condition*, where ch/ν is called the *cell Reynolds number*. For Burgers' equation (2.1) the cell Reynolds number is defined to be h/ν. Cell Reynolds number conditions occur for more general viscous flow problems and they impose a severe restriction on the class of problems which can be treated accurately [17].

For fixed ν and h it is possible to find multiple solutions to (2.2) both numerically and, in some special cases, analytically. Furthermore, there are instances where these solutions are to some degree "close together". We now indicate how the theory of M-functions relates the solutions of (2.2) to the cell Reynolds number.

We recall that a mapping $F : \Omega \subset R^n \to R^n$ is an M-function if it is (1) inverse isotone and (2) off diagonally antitone [16]. In particular it is known [16] that if Ω is an n-dimensional rectangle, F is an M-function on Ω if the Jacobian matrix F_u is an M-matrix on Ω. A sufficient condition that F_u be an M-matrix is (i) its diagonal elements are positive, (ii) its off-diagonal elements are negative, and (iii) it is irreducibly diagonally dominant. In reference to the equation $F\mathbf{y} = \mathbf{b}$, $\mathbf{y}^1, \mathbf{y}^2 \in \Omega$ are called sub and super solutions on Ω if $F\mathbf{y}^1 \leqq \mathbf{b}$ and $F\mathbf{y}^2 \geqq \mathbf{b}$ respectively, where the inequalities are with respect to the natural ordering on R^n. It then follows from a modification of a theorem of Rheinboldt and Ortega [13, p. 465] that if F is an M-function defined on the rectangle Ω and if $\mathbf{y}^1, \mathbf{y}^2$ are sub and super solutions for $F\mathbf{y} = \mathbf{b}$, then $F\mathbf{y} = \mathbf{b}$ has a *unique solution* $\mathbf{y}^*$, $\mathbf{y}^1 \leqq \mathbf{y}^* \leqq \mathbf{y}^2$, *with respect to* Ω. That is, there is exactly one solution to $F\mathbf{y} = \mathbf{b}$ in Ω although there may be other solutions outside of Ω. This theoretical framework may then be applied to (2.2) to conclude (for fixed h and ν) that if $h/\nu < 2$ then (2.2) has a solution $\mathbf{u}^*$ which is unique with respect to an n-dimensional rectangle Ω_h and satisfies

$$\mathbf{u}^* \in \Lambda \subset \Omega_h,$$

where

$$\Lambda = \{\mathbf{y} \mid \mathbf{y} \in R^n, |y_i| \leqq 1\},$$

$$\Omega_h = \{\mathbf{y} \mid \mathbf{y} \in R^n, |y_i| \leqq 2\nu/h\}.$$

In other words, there is a unique solution with respect to Ω_h when the cell Reynolds number condition is satisfied. Note that as $h \to 0$ or $\nu \to \infty$ the set Ω_h fills out all of R^n. In

[9] it is also shown that $\mathbf{u}^*$ is antisymmetric with respect to $x = \frac{1}{2}$ ($u_i = -u_{n-i+1}$, $i = 1, \cdots, n$). For $h/\nu < 2$ a solution to (2.2) lying in Ω_h can also be obtained by the Brouwer fixed point theorem. If we rewrite (2.2) in the form $G(\mathbf{u}) = \mathbf{u}$ it can then be shown that $G(\Omega_h) \subset \Omega_h$.

With reference to question (4) in the Introduction, the second order accurate scheme

$$\frac{u_{i+1}^2 - u_{i-1}^2}{4h} - \frac{\nu}{h^2}(\gamma_{i+1}u_{i+1} - 2\gamma_i u_i + \gamma_{i-1}u_{i-1}) = 0,$$

where

$$\gamma_i = \left(\frac{u_i h}{2\nu}\right) \coth\left(\frac{u_i h}{2\nu}\right)$$

was shown in [9] to have a unique solution. Also, when the boundary conditions are such that (2.1) has a positive (negative) solution, the proper one-sided differencing for uu_x with centered differencing for u_{xx} gives a scheme which has a unique positive (negative) solution.

2C. Numerical continuation of solutions. We now look at the solutions to (2.2) in more detail. The general idea which we pursue is to compute solutions to (2.2) by numerical continuation with respect to ν starting from finite difference solutions to the inviscid equations ($\nu = 0$). That is, we consider the nonlinear system

$$u_{i+1}^2 - u_{i-1}^2 = 0, \quad u_0 = 1, \quad u_{n+1} = -1, \tag{2.3}$$

which may be regarded as a difference scheme for

$$uu_x = 0, \quad u(0) = 1, \quad u(1) = -1. \tag{2.4}$$

Equation (2.4) does not admit a solution in the classical sense but admits generalized solutions with jump discontinuities. In fact, (2.4) does possess a solution

$$u(x) = \begin{cases} 1, & x < \frac{1}{2}, \\ -1, & x > \frac{1}{2}, \end{cases}$$

which is the unique inviscid limit of solutions to (2.1). For the remainder of the discussion in this section we restrict ourselves to the case where n is even. In that case there are 2^n solutions to (2.3) given by $u_i = \pm 1$. These solutions are enumerated in Table 1 for the case $n = 4$. In the case of n odd some of the u_i are not coupled to the boundary conditions and there are an infinite number of solutions. Starting with the solutions to (2.3) we numerically integrate the curves $(d/d\nu)\, F(\mathbf{u}(\nu), \nu) = F_u u_\nu + F_\nu = 0$. In order to show the existence of such curves, at least for ν sufficiently small, we use the implicit function theorem and evaluate F_u at the solutions of (2.3). The resulting matrix F_u is

$$\begin{bmatrix} 0 & u_2 & 0 & & \\ -u_1 & 0 & u_3 & & \\ 0 & -u_2 & 0 & \ddots & \\ & & \ddots & & u_n \\ & & & -u_{n-1} & 0 \end{bmatrix}$$

where each $u_i = \pm 1$. It is easy to show by minor expansion that the determinant of this matrix is 1 or -1.

TABLE 1
Discrete inviscid solutions to Burgers' equation ($a=-1$) *and duct flow equation* ($a=E/D$) *for* $n=4$. *See Figs.* 1b *and* 3b *respectively.*

Solution	u_1	u_2	u_3	u_4
1	1	1	1	1
2	1	1	1	a
3	1	1	a	1
4	1	1	a	a
5	1	a	1	1
6	1	a	1	a
7	1	a	a	1
8	1	a	a	a
9	a	1	1	1
10	a	1	1	a
11	a	1	a	1
12	a	1	a	a
13	a	a	1	1
14	a	a	1	a
15	a	a	a	1
16	a	a	a	a

The actual computations of solutions to (2.2) for increasing ν were performed using Davidenko's method [13]. For a particular value of ν Newton's method was used to solve (2.2), ν was then increased and the process repeated. In a region where computation became difficult due to turning or bifurcation points the following method was used. The system (2.2) with variable ν was augmented with $\det(F_u(\mathbf{u}(\nu), \nu)) = 0$ and solved. We note that more sophisticated methods have been developed for continuing solutions through turning or bifurcation points [2], [8], [12]. The qualitative behavior of the solutions as a function of ν is shown for the cases $n=2$ and $n=4$ in Fig. 1 where the vertical axis represents n-dimensional Euclidean space. In Fig. 1 we also depict the set Ω_h. For the case $n=4$ there are three turning points and three bifurcation points. There are four antisymmetric solutions (4, 6, 11, and 13). We call solution 4 the *principal solution* since it is the solution which best approximates (2.1). All other solutions are called *spurious*. The principal solution bifurcates twice and when $h/\nu < 2$ ($\nu > \frac{1}{10}$) the principal solution lies in Ω_h with all other solutions (9, 11, 15) outside of Ω_h as predicted by the results discussed earlier. Figure 1 is complete except for the possible existence of loops which cannot be found by the present method.

For the simple case of two interior mesh points a geometrical model indicates how bifurcation occurs. In this case (2.2) is a system with two equations and two unknowns u_1, u_2. If these equations are added to and subtracted from each other the following equivalent equations are obtained

$$(u_2+u_1)(u_2-u_1+\theta)=0,$$
$$(u_1+\tfrac{3}{2}\theta)^2+(u_2-\tfrac{3}{2}\theta)^2=2+2\theta+\tfrac{9}{2}\theta^2.$$

Solutions are given by the intersections of the lines $u_2=-u_1$ and $u_2=u_1-\theta$ with the circle of radius $(2+2\theta+\frac{9}{2}\theta^2)^{1/2}$ and center $(-3\theta/2, 3\theta/2)$. This situation is pictured in

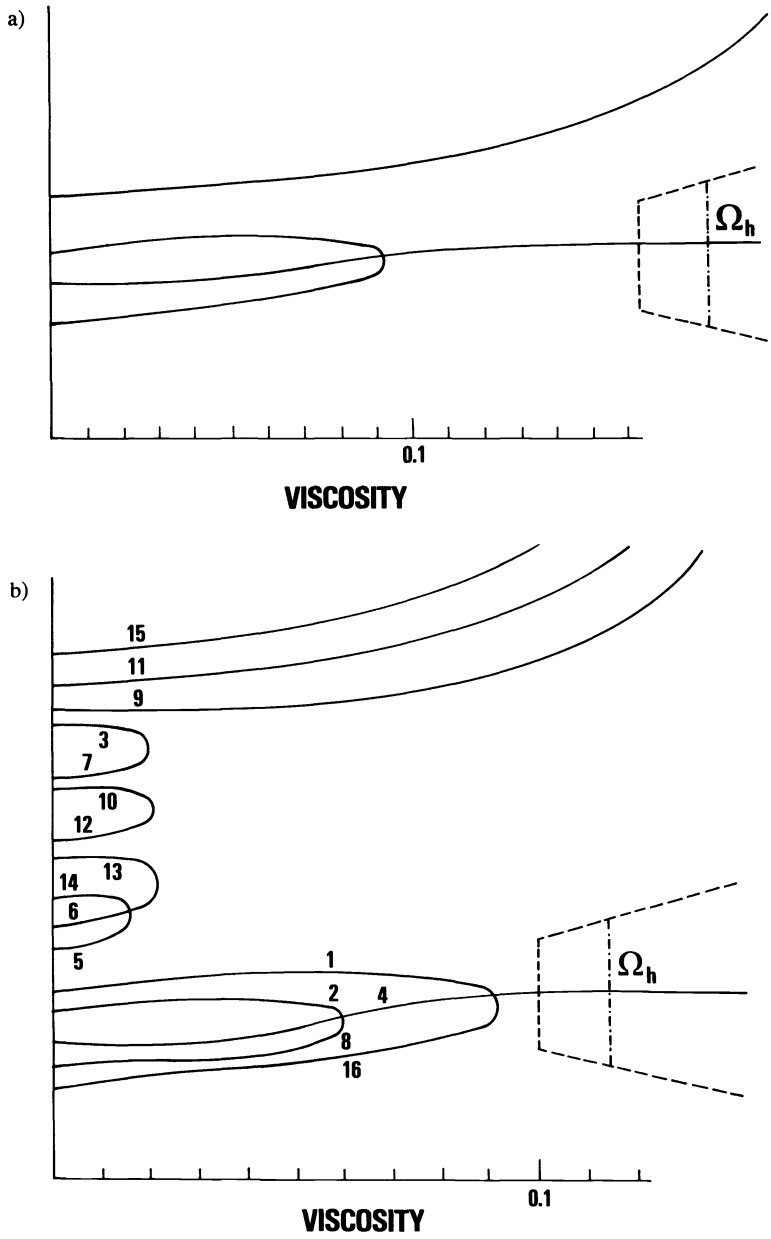


FIG. 1. *Behavior of discrete Burgers' solutions as a function of viscosity* ν *for* a) $n=2$ *and* b) $n=4$. *Numbered solutions correspond to Table* 1.

Fig. 2. As ν increases from 0, the line $u_2=u_1-\theta$ moves through the circle and at a critical value of ν tangents the circle at a point and bifurcation of that solution occurs. A similar but more complicated diagram models the antisymmetric solutions (one of which is the principal solution) in the case $n=4$. It is interesting to note in the general case of n even that the solutions of (2.2) must lie on an $(n-1)$-sphere with center $(-(n+1)\theta/2, (n+1)\theta/2, -(n+1)\theta/2, \cdots)$ and radius $(n+n\theta+n(n+1)^2\theta^2/4)^{1/2}$. To see this, multiply equations $2k$ by k $(k=1, \cdots, n/2)$ and equations $n-2k+1$ by $-k$ $(k=1, \cdots, n/2)$ and then add.

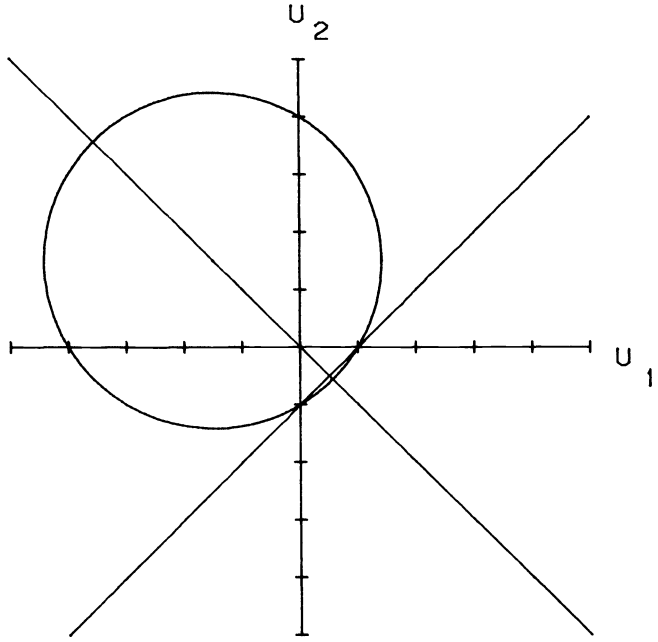


FIG. 2. *Geometric interpretation of discrete Burgers' solutions for* $n = 2$.

Bifurcations and turning points occur only when F_u is singular. We now examine the singularity of F_u for antisymmetric solutions when $n = 4$. In this case ($u_1 = -u_4$, $u_2 = -u_3$) F_u is

$$\begin{bmatrix} 2\theta & -\theta + 2u_2 & 0 & 0 \\ -\theta - 2u_1 & 2\theta & -\theta - 2u_2 & 0 \\ 0 & -\theta - 2u_2 & 2\theta & -\theta + 2u_1 \\ 0 & 0 & -\theta + 2u_2 & 2\theta \end{bmatrix}.$$

We remark that F_u is singular whenever $u_i = 2\nu/h = \theta/2$, $i = 1, 2$. In the case that $u_1 = 2\nu/h$, the sum of the rows of F_u is zero. When $u_2 = 2\nu/h$, F_u is reducible and $\det F_u = (2\theta)^2((2\theta)^2 - (-\theta - 2u_2)^2) = 0$. Although the argument is more involved it can be shown for the general case of n even that F_u is singular whenever $u_i = 2\nu/h$. For the case $n = 4$ and considering the principal solution we see that $u_1(\nu) = 1$ at $\nu = 0$ and approaches $2/3$ as $\nu \to \infty$. Since $u_1(\nu)$ is a continuous function of ν there must be a value of ν for which $u_1(\nu) = 2\nu/h$ and $u_2(\nu)$ must also take the value $2\nu/h$ by similar reasoning. In the general case we conjecture that there is a principal solution which bifurcates $n/2$ times.

As a final point of interest we discuss the obtainability of solutions of (2.2) by the use of time-dependent methods. Such methods are a common means of obtaining solutions to steady problems. Basically one considers a finite difference approximation to a time-dependent formulation of the problem, guesses an initial value and then marches through time until a steady state is reached to within some tolerance. We have differenced the time-dependent Burgers equation $u_t + uu_x = \nu u_{xx}$ with the predictor-corrector scheme of Brailovskaya [17] and with the standard forward-time, centered-space (FTCS) shceme. In the steady state both schemes reduce to (2.2). At $\nu = .005$ the Brailovskaya scheme would converge to solutions 1, 4, and 16, whereas FTCS would

only converge to the principal solution 4. However, other experiments indicate that both schemes will converge to spurious solutions near bifurcation points.

3. One-dimensional duct flow. We now consider a model for one-dimensional viscous, steady, compressible flow in a duct of variable cross sectional area $A(x)$. This model possesses solutions that are quite similar to those for Burgers' equation. The governing equations (see, e.g., [5], [11]) may be reduced to the single equation

$$Du_x + E\left(\frac{1}{u}\right)_x - Fu_{xx} - G = 0, \tag{3.1}$$

where $D = (\gamma+1)c/(2\gamma) - \mu A_x$, $E = (\gamma-1)cH/\gamma$, $F = \mu A/\gamma$, and $G = (\gamma-1)c(H/u - u/2)A_x/(\gamma A)$. Here u is the velocity, γ the ratio of specific heats, μ the viscosity coefficient, and c and H are constants of integration. If we linearize u about u_0 and assume constant area flow ($A_x = 0$), (3.1) reduces to

$$\left(D - \frac{E}{u_0^2}\right)u_x - Fu_{xx} = 0, \tag{3.2}$$

which is a linear Burgers equation. Consequently if centered differencing is used to approximate (3.2), the cell Reynolds number criterion for a nonoscillatory solution is $hD|1 - E/(Du_0)^2|/F < 2$, where h is the mesh size. It can be shown that this is equivalent to the condition $\rho_0 h|u_0(1 - 1/M^2)|/\mu < 2$ given by Meintjes [11], where ρ_0 is the density and M is the Mach number.

In order to apply our M-function approach we find it necessary to take $A_x = 0$, so that in (3.1) D, E and F are constants and $G = 0$. The simplified (3.1) is discretized (analogously to Burgers' equation) as

$$\frac{D(u_{i+1} - u_{i-1})}{2h} + \frac{E\left(\frac{1}{u_{i+1}} - \frac{1}{u_{i-1}}\right)}{2h} - \frac{F(u_{i+1} - 2u_i + u_{i-1})}{h^2} = 0, \tag{3.3}$$

and the boundary conditions are taken to be

$$u_0 = \alpha, \qquad u_{n+1} = \beta, \tag{3.4}$$

where $\alpha, \beta > 0$. Let $\phi = \min(\alpha, \beta)$ and $\psi = \max(\alpha, \beta)$. Then in a manner entirely parallel to that for Burgers' equation, it can be shown that (3.3), (3.4) possess a unique solution with respect to the rectangle

$$\Omega_h^D = \left\{ \mathbf{y} \,\middle|\, \mathbf{y} \in R^n, \left(\frac{E/D}{1 + \frac{2F}{hD}}\right)^{1/2} \leqq y_i \leqq \begin{cases} \infty & \text{if } h \leqq \dfrac{2F}{D}, \\ \left(\dfrac{E/D}{1 - \frac{2F}{hD}}\right)^{1/2} & \text{if } h > \dfrac{2F}{D} \end{cases} \right\},$$

when $hD(E/(D\phi^2) - 1)/F < 2$, and either $\phi^2 < E/(2D)$ or $hD(1 - E/(D\psi^2))/F < 2$. Hence again, for fixed h, it is possible to choose μ (and thus F) large enough to assure a unique solution in a certain n-dimensional rectangle. Similarly, for fixed μ, h can be made small enough. Furthermore, the linear criterion for nonoscillatory solutions and the nonlinear criterion for uniqueness are again strongly related.

As for Burgers' equation, when n is even there are 2^n solutions of (3.3), (3.4) when $\mu = F = 0$. Choosing $\alpha = 1$ and $\beta = E/D$, the inviscid solutions are given by taking $u_i = 1$ or $u_i = E/D$ at each mesh point i (see Table 1 for $n = 4$). We again employ our numerical continuation procedure with $E/D = 0.63$ to follow all of the $\mu = 0$ solutions to values of viscosity $\mu > 0$. The resulting diagrams showing the behavior of the solutions for $n = 2$ and $n = 4$ are given in Fig. 3. We observe that no solutions bifurcate

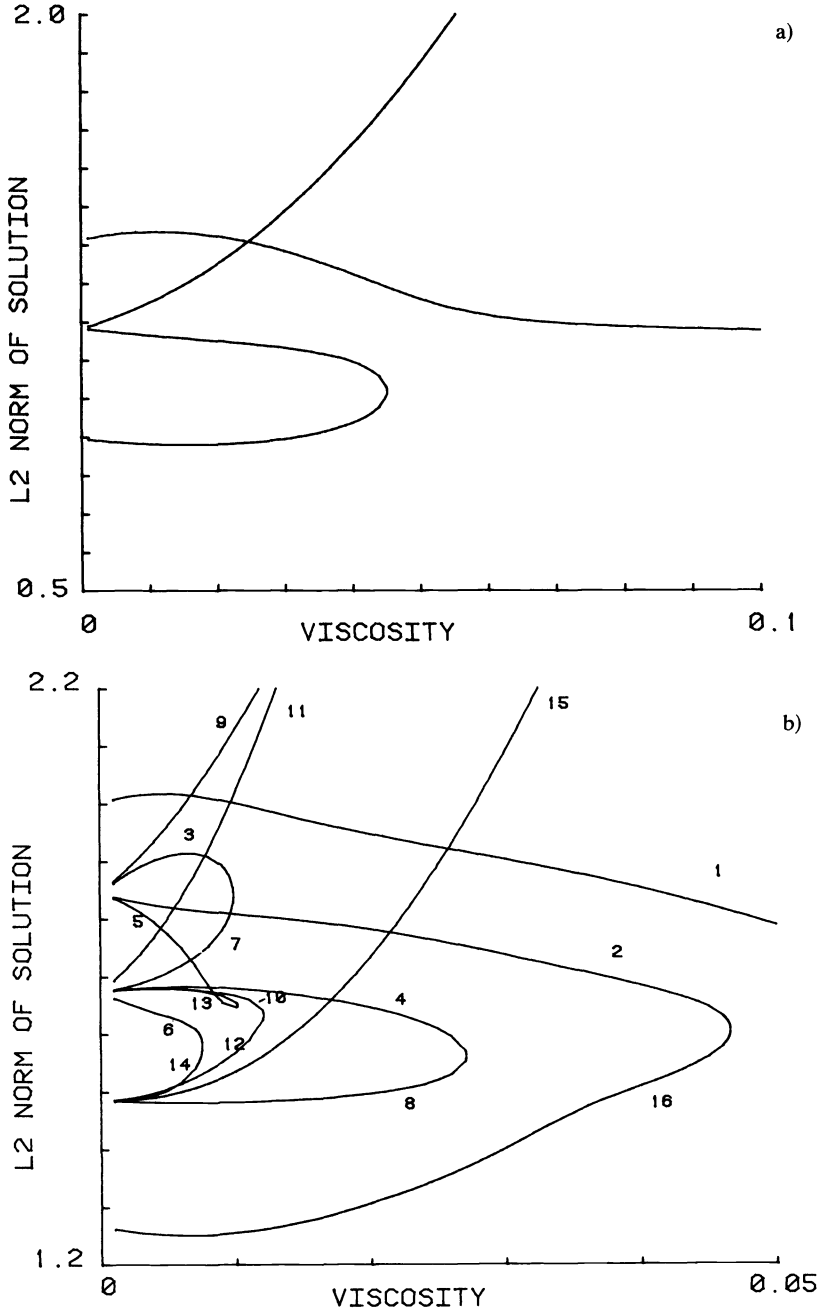


FIG. 3. *Behavior of discrete duct flow solutions as a function of viscosity* μ *for* a) $n = 2$ *and* b) $n = 4$. *Numbered solutions correspond to Table* 1.

off the principal solution in these cases. It is also interesting that, of the solutions that remain bounded, the principal solution is the one with maximum l_2-norm.

Some numerical experiments with variable area $A(x)$ were also carried out for $n = 2$ and $n = 4$. In these computations, turning points were found when the constant area duct solutions for $\mu = 0$ were continued with respect to duct shape $A(x)$.

4. Supersonic blunt body problem. In [20] a particular discretization of the steady inviscid axisymmetric blunt body problem was found to have two solutions for Mach 3 flow over a sphere. In this problem an incoming supersonic stream impinges on the sphere and a bow shock forms separating the undisturbed flow from the disturbed "shock layer" between the sphere and the bow shock. The problem is to determine the flow field in the shock layer and the position of the explicitly tracked bow shock. In this approach a variation of Newton's method was used to proceed from some initial guess to a steady state. Lines of constant density (isopycnics) and the computed bow shock positions for the two solutions A and B are shown on the left in Fig. 4. We note that the computed bow shocks are oscillatory in "opposite senses" in the two solutions. These solutions are certainly close together and it is difficult to select the best approximation. Both solutions agree reasonably well with experiment. In addition to the Newton procedure, we have also used the method of Brailovskaya to integrate time-dependent equations which yield the same steady state difference system. When this method is started near solution B it converges to B, but when started near A it fails to converge.

Here we investigate the behavior of solutions A and B as artificial viscosity is added to the governing inviscid equations. Without going into detail, a smoothing

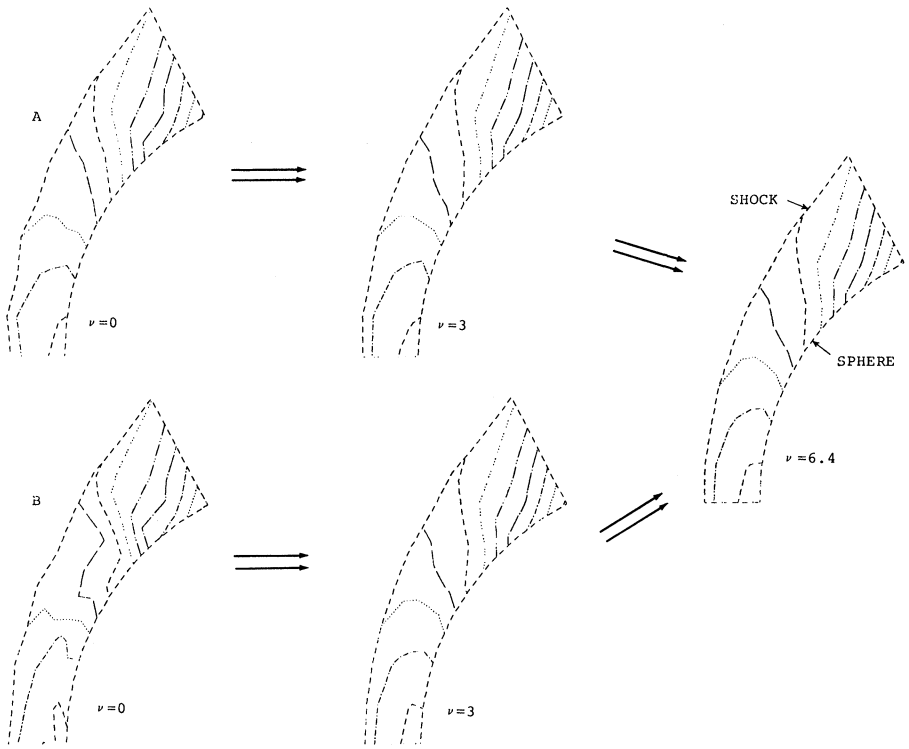


FIG. 4. *Behavior of computed bow shocks and isopycnics for two solutions of the blunt body problem as artificial viscosity ν is increased.*

operator is added to each component of the governing PDE system. This is frequently done to damp computational oscillations or to "stabilize" a scheme. In this procedure the inviscid boundary conditions are maintained. The present computational experiment is therefore not strictly analogous to the model problems previously discussed since in those problems the correct viscous and inviscid boundary conditions are the same.

We use our numerical continuation procedure to proceed from one value of artificial viscosity to the next. As shown in Fig. 4, as the artificial viscosity ν is increased the solutions tend toward each other. At some critical viscosity $\nu = 6.4$ the solutions coincide, and for greater values it seems that no nearby solution exists. Furthermore, the determinants of the Jacobian matrices for the two solutions are of opposite sign and tend toward each other as the critical viscosity is approached. It therefore appears that a turning point has been found. Were it not for the above-mentioned boundary condition question, we might be tempted by analogy with the model results to think of both A and B as spurious solutions (i.e., not the principal solution).

5. Concluding remarks. For the problems considered here, we have been able to partially answer the questions posed in the Introduction. By examining the Burgers equation in detail we see that the behavior of multiple finite difference solutions can be quite complicated even for a very simple model. For this model we can understand some of the behavior because we have a theory which predicts the crucial role of the cell Reynolds number and we have a computational method to follow the solutions. These results suggest a relationship between computational oscillations and multiple solutions which we do not yet fully understand.

In more general circumstances the questions in the Introduction are more difficult to answer. Nevertheless, our computational results for the blunt body problem and other work previously cited suggest that the phenomena uncovered for the discrete Burgers equation may also be found in more complicated practical problems. In the absence of methods guaranteeing a unique solution, we can only recommend that investigators be careful in implementing their methods and in checking their numerical results. When implementing methods this means starting iterative or time-accurate methods from different initial guesses, insisting on tight convergence criteria, repeating computations with different mesh sizes, and even using different methods whose final solutions satisfy the same system or difference equations. We remark that although time accurate methods may converge to spurious solutions, they are apparently less likely to do so than are general iterative methods. However, time accurate methods are usually slower to converge to a steady state [10]. In judging the computed results we should be suspicious of unanticipated oscillatory solutions and should check for qualitative properties that the discrete solutions should inherit from the continuous problem. We must be careful in evaluating results obtained with different mesh sizes. If such results are greatly different we must obviously be cautious; however, even when the results are nearly the same, we still cannot be completely sure that we are on the principal solution branch.

REFERENCES

[1] E. ALLGOWER, *On a discretization of* $y'' + \lambda y^k = 0$, Proc. Conf. Roy. Irish Acad., Academic Press, New York, 1975, pp. 1–15.

[2] 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.

[3] E. ALLGOWER AND K. GEORGI, *Homotopy methods for approximating several solutions to nonlinear systems of equations*, in Numerical Solution of Highly Nonlinear Problems, W. Forster, ed., North-Holland, Amsterdam, 1980, pp. 253–270.

[4] W.-J. BEYN AND E. DOEDEL, *Stability and multiplicity of solutions to discretizations of nonlinear ordinary differential equations*, this Journal, 2 (1981), pp. 107–120.

[5] L. CROCCO, *A suggestion for the numerical solution of the steady Navier-Stokes equations*, AIAA J., 3 (1965), pp. 1824–1832.

[6] B. ENGQUIST AND S. OSHER, *Stable and entropy satisfying approximations for transonic flow calculations*, Math. Comp., 34 (1980), pp. 45–75.

[7] A. HARTEN, J. M. HYMAN AND P. LAX, *On finite difference approximations and entropy conditions for shocks*, Comm. Pure Appl. Math., 29 (1976), pp. 297–322.

[8] 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.

[9] 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.

[10] H. LOMAX AND J. STEGER, *Relaxation methods in fluid mechanics*, Ann. Rev. Fluid Mechanics, 7 (1975), pp. 63–87.

[11] K. MEINTJES, *Predictor-corrector methods for time dependent compressible flows*, PhD Thesis, Princeton Univ., Princeton, NJ, 1980.

[12] G. MOORE AND A. SPENCE, *The calculation of turning points of nonlinear equations*, SIAM J. Numer. Anal., 17 (1980), pp. 567–576.

[13] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[14] S. OSHER, *Nonlinear singular perturbation problems and one-sided difference schemes*, SIAM J. Numer. Anal., 18 (1981), pp. 129-144.

[15] H.-O. PEITGEN, D. SAUPE AND K. SCHMITT, *Nonlinear elliptic boundary value problems versus their finite difference approximations: Numerically irrelevant solutions*, Rep. 19, Universität Bremen, 1980, J. Reine Angew. Math., to appear.

[16] W. C. RHEINBOLDT, *Methods for Solving Systems of Nonlinear Equations*, CBMS Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 1974.

[17] P. ROACHE, *Computational Fluid Dynamics*, Hermosa, Albuquerque, NM, 1976.

[18] R. SCHREIBER AND H. KELLER, private communication.

[19] G. R. SHUBIN, A. B. STEPHENS AND H. M. GLAZ, *Steady shock tracking and Newton's method applied to one dimensional duct flow*, J. Comput. Phys., 39 (1981), pp. 364–374.

[20] G. R. SHUBIN, A. B. STEPHENS, H. M. GLAZ, A. B. WARDLAW AND L. B. HACKERMAN, *Steady shock tracking, Newton's method, and the supersonic blunt body problem*, this Journal, to appear.

[21] A. B. STEPHENS AND A. G. WERSCHULZ, *A comparison of Newton-like methods for the transonic small disturbance equation*, NSWC TR 80-271, Naval Surface Weapons Center, Silver Spring, MD, 1980.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0005 $01.00/0

SOLUTION OF LARGE-SCALE SPARSE LEAST SQUARES PROBLEMS USING AUXILIARY STORAGE*

J. A. GEORGE,† M. T. HEATH‡ AND R. J. PLEMMONS§

Abstract. Very large sparse linear least squares problems arise in a variety of applications, such as geodetic network adjustments, photogrammetry, earthquake studies, and certain types of finite element analysis. Many of these problems are so large that it is impossible to solve them without using auxiliary storage devices. Some problems are so massive that the storage needed for their solution exceeds the *virtual* address space of the largest machines. In this paper we describe a method for solving such problems on a typical (large) computer and provide the results of some experiments illustrating the effectiveness of our approach. The method includes an automatic partitioning scheme which is essential to the efficient management of the data on auxiliary files.

Key words. large-scale sparse least squares, auxiliary storage, orthogonal decomposition, incomplete nested dissection

1. Introduction and overview

1.1. Introduction. In this paper a method is presented for solving the linear least squares problem

$$\min_x \|Ax - b\|_2 \tag{1.1}$$

when the $m \times n$ matrix A is very large and sparse and has full column rank n. The problems we wish to solve are so large that the use of auxiliary storage is essential regardless of the numerical method employed. In some cases storage requirements may even exceed the virtual address space of the largest machines, and therefore auxiliary space cannot be managed implicitly by a paging algorithm. Our approach is to break the large problem up into smaller subproblems which are processed sequentially, eventually producing the solution to the original problem. Such an approach requires a method for partitioning the large problem, a computational module for processing the subproblems, and an algorithm for managing external files containing intermediate results. In the remainder of this section, after giving some specific examples of large-scale least squares problems, we survey possible numerical methods and then describe the particular technique we have chosen for the computational module. In § 2, problem partitioning and data management are discussed. Section 3 presents numerical test results and observations.

1.2. Examples of large-scale least squares problems. In recent years least squares problems of ever increasing size have arisen with ever increasing frequency. One reason for this is that modern data acquisition technology allows the collection of massive amounts of data. Another factor is the tendency of scientists to formulate more and more complex and comprehensive models in order to obtain finer resolution and more realistic detail in describing physical systems. Particular areas in which such large-scale squares problems occur include geodetic surveying (Avila and Tomlin

* Received by the editors August 11, 1980. 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 Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1.

‡ Computer Sciences Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830.

§ Departments of Computer Science and Mathematics, North Carolina State University, Raleigh, North Carolina 27650. The research of this author was supported in part by the U.S. Army Research Office.

[1979], Golub and Plemmons [1980]), photogrammetry (Golub, Luk and Pagano [1980]), earthquake studies (Vanicek, Elliott and Castile [1979]), and in the natural factor formulation of the finite element method (Argyris and Brönlund [1975], Argyris et al. [1978]). An example of truly spectacular size is the least squares adjustment of coordinates (latitudes and longitudes) of stations comprising the North American Datum, to be completed in 1983 by the U.S. National Geodetic Survey (Kolata [1978]). This enormous task requires solving, perhaps several times, a least squares problem having *six million equations in four hundred thousand unknowns.*

1.3. Numerical methods for sparse least squares. Several methods have been proposed for solving sparse linear least squares problems (Duff and Reid [1976], Björck [1976], Gill and Murray [1976]). For our purposes the most important qualities in a numerical method will be storage requirements, numerical stability, and convenience in utilizing auxiliary storage.

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 determining 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.

A well-known stable alternative to the normal equations is provided by orthogonal factorization (Golub [1965]). An orthogonal matrix Q is computed which reduces A to upper trapezoidal form

$$QA = \begin{bmatrix} R \\ 0 \end{bmatrix}, \qquad Qb = \begin{bmatrix} y \\ z \end{bmatrix}, \tag{1.3}$$

where R is $n \times n$ and upper triangular. Since Q does not change the two-norm, we have

$$\|Ax - b\|_2 = \left\| \begin{bmatrix} R \\ 0 \end{bmatrix} x - \begin{bmatrix} y \\ z \end{bmatrix} \right\|_2, \tag{1.4}$$

and therefore the solution to (1.1) is obtained by solving the triangular system $Rx = y$. 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-in. The use of Givens rotations is much more attractive in that the matrix is processed by rows, gradually building up R, and intermediate fill-in

is confined to the working row. This approach, implemented with a good column ordering and an efficient data structure, is the basis for the computational module described in § 1.4.

Other direct nonnormal-equations methods for sparse least squares, including those of Peters and Wilkinson [1970] (as implemented by Björck and Duff [1980]) and Hachtel [1976], were considered, but these elimination methods require row and column pivoting to preserve sparsity as well as some form of pivoting for stability, necessitating access to the entire matrix. This feature greatly inhibits the partitioning of large problems and the flexible use of auxiliary storage.

1.4. The computational module. The numerical method we use is developed in detail in George and Heath [1980]. Our motivation is to combine the flexibility, convenience and low storage requirements of the normal equations with the stability of orthogonal factorization. The basic steps of the algorithm are as follows:

ALGORITHM 1. Orthogonal decomposition of A.

1. Determine the structure (not the numerical values) of $B = A^T A$.
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 by processing the rows of A one by one using Givens rotations.

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 well developed sparse matrix software (George and Liu [1979]). 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-in during the numerical computation is unnecessary. 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. A suboptimal row ordering to reduce the amount of computation associated with intermediate fill-in during the orthogonal decomposition phase was shown to be effective in George and Heath [1980] and is used here. Alternatively, for problems having widely varying weights or row norms it may be necessary to choose the row ordering so as to maintain stability. Thus Algorithm 1 requires the same storage and exploits sparsity at least to the same degree as the normal equations, allows convenient use of auxiliary storage, and in addition is numerically stable.

2. Decomposition of A using auxiliary storage

2.1. Introduction. This section consists of two parts. The first describes an algorithm for finding a column ordering and partitioning of A which lends itself to the efficient use of auxiliary storage. The method uses slightly modified ideas and techniques which have already been described in detail by George and Liu [1978], so our presentation is brief and limited to showing our modifications and the relevance of the scheme to the least squares problem. It is important to note that in some contexts such partitionings/orderings arise as a natural by-product of the modelling procedure (finite element analysis) or data acquisition (geodesy). In these cases, this "preprocessing" algorithm would not be required.

The second part of this section deals with the utilization of the software described in § 1.4 and the partitioning provided by Algorithm 2 of § 2.2, along with files on auxiliary storage, to solve very large least squares problems.

2.2 Finding an appropriate ordering and partitioning for the columns of A. In the sequel it is convenient to work with the symmetric graph associated with the normal equations matrix $B = A^TA$. The graph $G = (X, E)$ associated with B has n nodes x_i, $i = 1, 2, \cdots, n$, which form X, and an edge set E consisting of unordered pairs of nodes with $\{x_i, x_j\} \in E$ if and only if $B_{ij} = B_{ji} \neq 0$, $i \neq j$. Thus the nodes x_i correspond to the variables of the least squares problem, i.e., to the columns of A. Implicit here is the assumption that the nodes of G have been labelled as the columns of A. Thus, a relabelling of the nodes of G corresponds to a symmetric permutation of the matrix B, or equivalently, a column permutation of A. Given G *without* any labels, finding an appropriate permutation of the columns of A can be viewed as finding an appropriate labelling for G.

A graph $G' = (X', E')$ is a *subgraph* of $G = (X, E)$ if $X' \subset X$ and $E' \subset E$. For $Y \subset X$, $G(Y)$ refers to the subgraph $(Y, E(Y))$ of G, where $E(Y) = \{\{u, v\} \in E | u, v \in Y\}$.

Nodes x and y are said to be *adjacent* if $\{x, y\}$ is an edge in E. For $Y \subset X$, the *adjacent set* of Y is defined as

$$\text{Adj}\,(Y) = \{x \in X - Y | \{x, y\} \in E \text{ for some } y \in Y\}.$$

A *path* of length l is a sequence of l edges $\{x_0, x_1\}, \{x_1, x_2\}, \cdots, \{x_{l-1}, x_l\}$ where all the nodes are distinct except for possibly x_0 and x_l. A graph G is *connected* if there is a path joining each pair of distinct nodes.

A partitioning $\mathcal{P}$ of G is an ordered collection of node sets

$$\mathcal{P} = \{Y_1, Y_2, \cdots, Y_p\},$$

where $Y_i \cap Y_j = \varnothing$, $i \neq j$, and $\bigcup_{i=1}^{p} Y_i = X$.

Given a partitioning $\mathcal{P}$ of G, the only numberings (labellings) we will be concerned with in this paper will be *compatible* with $\mathcal{P}$. That is, each Y_i is numbered consecutively, and nodes in Y_i are numbered before those in Y_{i+1}.

A subset $Y \subset X$ of the node set of G is a *separator* of G if $G(X - Y)$ consists of two or more connected components. A separator is minimal if no subset of it is a separator.

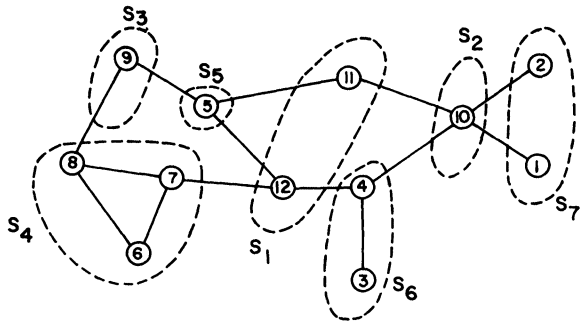
As we shall see below, a desirable column ordering and partitioning for A is provided by a *nested dissection partitioning* of the graph of B (George, Poole and Voigt [1978]). The algorithm we use here can be described as follows, where $G = (X, E)$ is the graph of B and μ is a user-supplied parameter.

ALGORITHM 2. Incomplete nested dissection partitioning.

1. Set $V = X$, $p = 0$, and $n = |X|$.
2. If $V = \varnothing$, go to Step 4. Otherwise, let $G(T)$ be a connected component of $G(V)$ and set $p = p + 1$. If $|T| \leqq \mu$, set $S_p = T$; otherwise, find S_p, a minimal separator of $G(T)$ which disconnects it into two or more components of approximately equal size.
3. Set $V = V - S_p$, $n = n - |S_p|$, and go to Step 2.
4. Set $\mathcal{P} = \{Y_1, Y_2, \cdots, Y_p\}$, where $Y_i = S_{p+1-i}$, $i = 1, 2, \cdots, p$.

Apart from the inclusion of the threshold μ, and the omission of any specific labelling strategy for each S_p, our implementation corresponds exactly to that described by George and Liu [1978, pp. 1054–1060], so the reader is referred there for details. For our purposes, any ordering compatible with $\mathcal{P}$ is acceptable, and the choice of μ is discussed in § 3. It should be obvious that μ governs the relative "completeness" of the dissection procedure. An example of a nested dissection partitioning along with a compatible ordering is given in Fig. 2.1.

In order to avoid unnecessarily complicated notation in the following section, we assume from now on that A has been reordered by columns and that the Y_i simply consist of the appropriate consecutive subsequences of the first n integers. In other words, the nodes of Y_i have been replaced by their labels.



$\mu = 3$
$p = 7$
$\mathcal{P} = \{Y_1, Y_2, \cdots, Y_7\}$ where $Y_i = S_{8-i}$.

FIG. 2.1. *An example of a nested dissection partitioning of a graph and an induced numbering of its nodes.*

We now define a partitioning $\mathcal{R} = \{Z_1, Z_2, \cdots, Z_p\}$ of the row numbers of A, induced by the column partitioning $\mathcal{P}$, as follows. Let

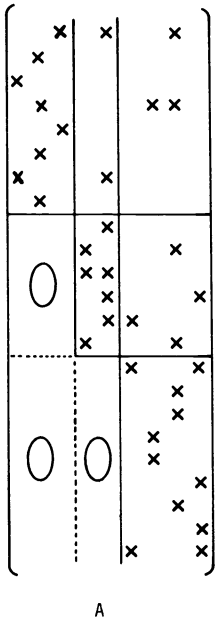
$$Z_i = \{k \mid \exists j \in Y_i \ni a_{kj} \neq 0\} - \left\{ \bigcup_{l=1}^{i-1} Z_l \right\}, \qquad i = 1, 2, \cdots, p,$$

with $Z_0 = \varnothing$. The manner in which the Z_i are defined above implies that in general, for a sparse matrix A, if the rows of A are permuted so that those specified by Z_i appear above those specified by Z_{i+1}, then A will have *block upper trapezoidal form*, as depicted in Fig. 2.2, for three diagonal blocks. Again, to keep the presentation simple, we assume that the rows of A have been relabelled so that the Z_i consist of consecutive integers, with those in Z_i preceding those in Z_{i+1}. We denote the submatrices of A by A_{ij}, $1 \leqq i, j \leqq p$, and our objective is to find a form for A such that $A_{ij} = 0$ for $i > j$.

As mentioned earlier, this process of dissection of the problem variables into independent subsets may arise more or less automatically or may be carried out by some means other than the one proposed above.

For example, in geodesy, observations between or among control points are collected and tabulated by geographic region, so the variables (coordinates) and observations are naturally arranged in a hierarchical structure. Moreover, automatic subdivision can be implemented on the basis of specifying latitude and longitude boundaries as separators for the coordinate sets. For a more detailed description of this process, called Helmert blocking by geodesists, see Golub and Plemmons [1980] and the references cited therein.

In the analysis of structures, the variables in the model are often subdivided according to subassemblies, with each component being processed by an individual design group. This may occur at several levels, leading to "*multi-level substructuring.*" This partitioning procedure together with the natural factor formulation of the finite element method (Argyris et al. [1978]) leads to a sparse least squares problem having block upper trapezoidal structure, as illustrated in Fig. 2.2.



$Y_1 = \{1, 2, 3\}$ $Y_1' = \{5, 7, 8\}$

$Y_2 = \{4, 5\}$ $Y_2' = \{6, 8, 9\}$

$Y_3 = \{6, 7, 8, 9\}$ $Y_3' = \varnothing$

$Z_1 = \{1, 2, \cdots, 8\}$

$Z_2 = \{9, 10, \cdots, 14\}$

$Z_3 = \{15, 16, \cdots, 23\}$

$\mathcal{P} = \{Y_1, Y_2, Y_3\}$

$\mathcal{R} = \{Z_1, Z_2, Z_3\}$

FIG. 2.2. *A* 23×9 *sparse matrix* A *partitioned by columns according to* $\mathcal{P}$ *and the induced row partitioning* $\mathcal{R}$.

A special case of the block upper trapezoidal form is the so-called *block angular form*, where $A_{ij} = 0$ unless $j = i$ or $j = p$. This form arises naturally in many mathematical programming contexts, but more important from our point of view is that Weil and Kettler [1971] have provided a heuristic algorithm for permuting a *general* sparse matrix into block angular form. We have not used their algorithm, but in some contexts it might serve as an alternate "preprocessor" (partition generator) to the one proposed in this section. Golub and Plemmons [1980] have exploited this block angular form structure in connection with computing orthogonal decompositions of problems arising in geodesy.

2.3. Reduction of A using auxiliary storage. Before describing in detail how the partitioning $\mathcal{P}$ is used in exploiting auxiliary storage, we first review the basic computational procedure, without any reference to the actual implementation. For definiteness, we assume $|\mathcal{P}| = 3$ and that A has the form shown in Fig. 2.3. The computation consists of p major steps; the ith step results in the generation of $|Y_i|$ rows of the upper triangular factor R of A. During the ith step, the columns of Y_i and a subset of columns from $\bigcup_{j=i+1}^{p} Y_j$ are involved in the computation. We denote the column numbers of this subset by Y_i'.

The computation involved in the first step is depicted in Fig. 2.3, and all the other major steps are similar, generating the sequence of successively smaller matrices $A_1, A_2, A_3, \cdots, A_p$. At the first step, which transforms $A = A_1$ to A_2, the matrix $[A_{11}|\tilde{A}_1]$ is reduced to upper triangular form using Algorithm 1 described in § 1.4. The matrix $\tilde{A}_1$ consists of those columns of A_{12} and A_{13} which are non-null. (Thus, $\tilde{A}_1$ is simply a "compressed" version of A_{12} and A_{13}.) The first $|Y_1|$ columns of the resulting upper triangular matrix are the first $|Y_1|$ columns of R; the remaining columns are "compressed," bearing the same relationship to R as $\tilde{A}_1$ does to $[A_{12}|A_{13}]$. The rows corresponding to these latter columns are "expanded" and put on the top of the

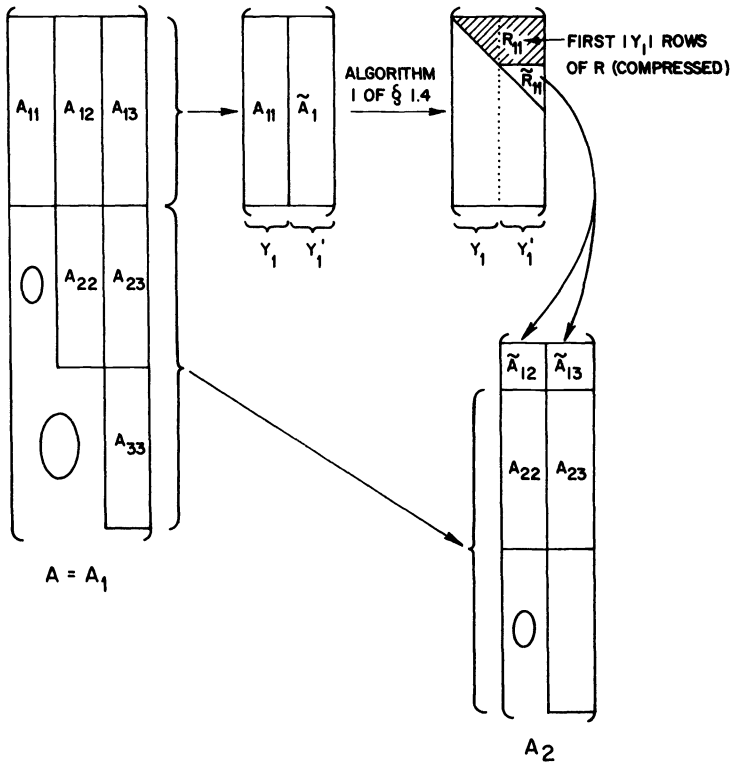


FIG. 2.3. *Pictorial description of the first step of the $|\mathcal{P}|$ step reduction of A to upper trapezoidal form.*

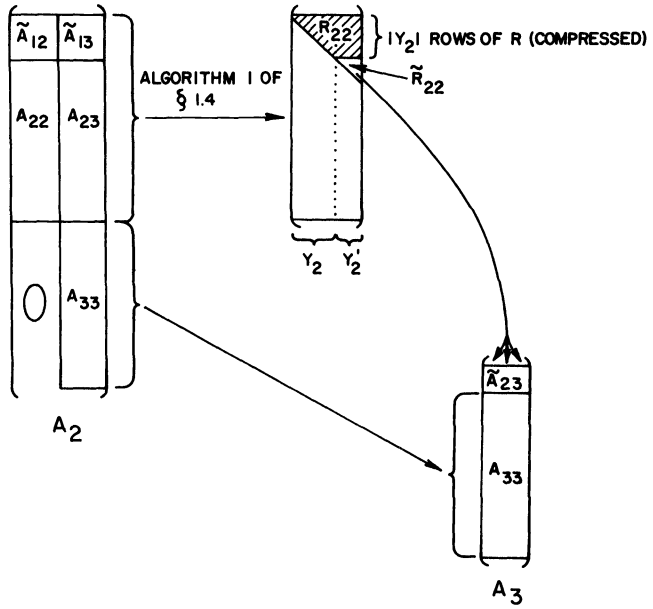


FIG. 2.4. *Depiction of the second step of the block-reduction of A to upper trapezoidal form.*

second row-block of A, yielding A_2. The next step of the computation is depicted in Fig. 2.4, and the final step has no special features.

Thus in the general case, after $i-1$ steps of the reduction process, the matrix A_i will have the form shown in Fig. 2.5.

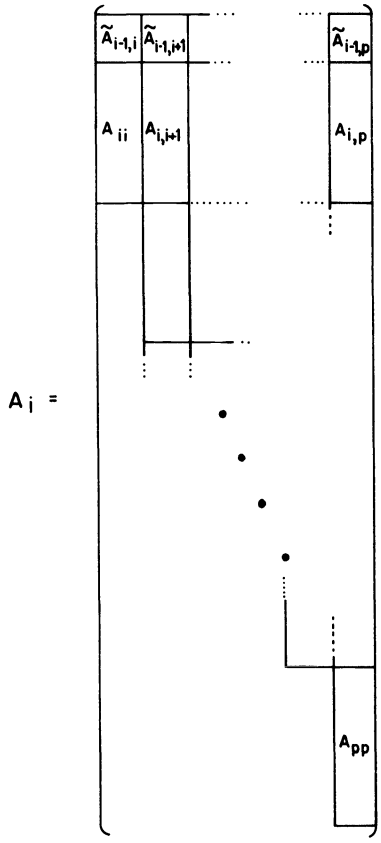


FIG. 2.5. *Block structure of* A_i.

Note that since we assume that the rows of A_i are stored on auxiliary storage at all times, the only storage needed at the ith stage of the computation is that required for the presumably sparse upper triangular matrix R_i, where

$$R_i = \begin{matrix} R_{ii} \\ \tilde{R}_{ii} \end{matrix}.$$

Another important practical observation is that in Algorithm 1 of § 1.4, and in the one that is described in the sequel, there is *no restriction* on the order that the rows of any of the A_i are stored so that any beneficial row ordering may be used.

In what follows it is helpful to have names for certain subsets of the rows of A_i. We define the set of rows of A_i having nonzeros in columns which interesect Y_i by $\bar{Z}_i$, and we use Z_i' to denote the remaining rows of A_i. Thus, it is precisely those rows in $\bar{Z}_i$ which are involved in the ith major step of the computation.

For simplicity, we have not included the right-hand side in our Figs. 2.3–2.5, but we intend that it be processed simultaneously with the rows of A, and carried along in parallel. In particular, throughout this paper, when we refer to processing a "row of A" or an "equation," we implicitly include the corresponding element of b, the weight (if any), and so on. Our program formally allows for weights, and in order to handle them uniformly (i.e., to avoid distinguishing between virgin rows in A_i and those that have been transformed), a weight of 1 is associated with transformed equations.

We are now ready to describe the algorithm for reducing A, which involves the use of four files. These files may be stored on tapes, disks, drums; only *serial* access to the files is necessary. The files are:

1. R-file: accepts the rows of R as they are computed.
2. C-file: (current file) at the beginning of the ith major step of the computation, contains the rows of A_i in arbitrary order.
3. $\bar{Z}$-file: during the ith step of the computation, contains the rows in the set $\bar{Z}_i$; that is, those rows of A that are involved in the computation at the ith major step.
4. Z'-file: at the beginning of the ith step, this file is empty. The C-file is read, and split into the $\bar{Z}$-file and this file, which receives the rows of A_i in the set Z_i'. After the computation of R_i is performed, the rows of $\tilde{R}_{ii}$ are written on this file, and the Z'-file *becomes the C-file for the next step of the computation.* (It now contains the rows of A_{i+1}.)

Thus the $\bar{Z}$-file is a scratch file, and the roles of the C-file and Z'-file alternate at each succeeding major step of the computation.

ALGORITHM 3. Block reduction of A to upper triangular form.

For $i = 1, 2, \cdots, p$ do the following:

1. Rewind the C-file, $\bar{Z}$-file, and Z'-file. Read the equations from the C-file one by one, and for each do the following:
   1.1. If the equation number is in $\bar{Z}_i$, write the equation on the $\bar{Z}$-file and record the structure it contributes to the normal equations matrix $H_i^T H_i$, corresponding to the matrix $H_i \equiv [A_{ii} \mid \tilde{A}_i]$.
   1.2. If the equation is in Z_i', write the equation on the Z'-file.
2. Order the equations so that $H_i^T H_i$ suffers low fill-in, restricting the ordering so that the variables in Y_i' appear *last.*
3. Create the data structure for R_i.
4. Rewind the $\bar{Z}$-file. Read the equations from it and compute R_i, also applying the transformations to b.
5. Write the rows of R_{ii} on the R-file along with the transformed elements of b.
6. Write the rows of $\tilde{R}_{ii}$ on the Z'-file, along with the transformed elements of b.
7. Reverse the names of the C-file and Z'-files.

It should be clear that the combination of the structure recording part of Step 1.1 along with Steps 2, 3, and 4 is essentially Algorithm 1, described in § 1.4. The only modification is the adjustment of the ordering provided by Algorithm 1 so that variables in Y_i' appear last; all others remain in their same relative position. The ordering of the variables of Y_i that is provided in Step 2 is recorded so that the rows of R written on the R file can be processed in the correct order in the back substitution. (Note that Algorithm 2 in § 2.1 only provided the partitioning and did not provide an ordering for each Y_i.)

At the conclusion of the reduction of A to upper trapezoidal form, along with the simultaneous reduction of b, the rows of R and the corresponding right-hand side

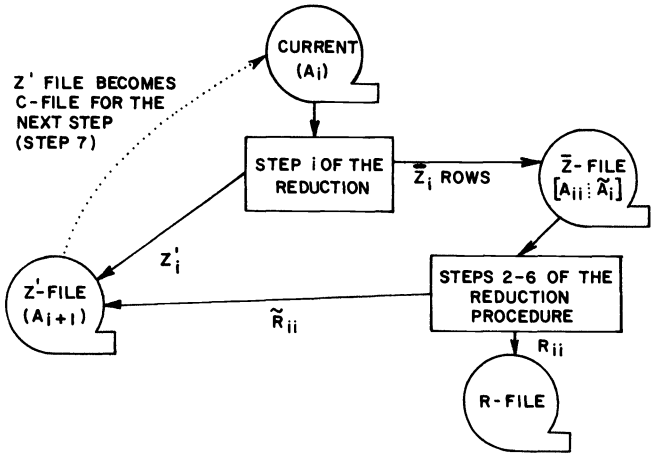


FIG. 2.6. *Diagram of the data flow of the algorithm for reducing A to upper trapezoidal form.*

elements (y in (1.3) of § 1) will be on the R-file, with the "write head" positioned at the end of the last record. The following simple back substitution is then used to compute x.

For $i = n, n-1, \cdots, 1$ do the following:

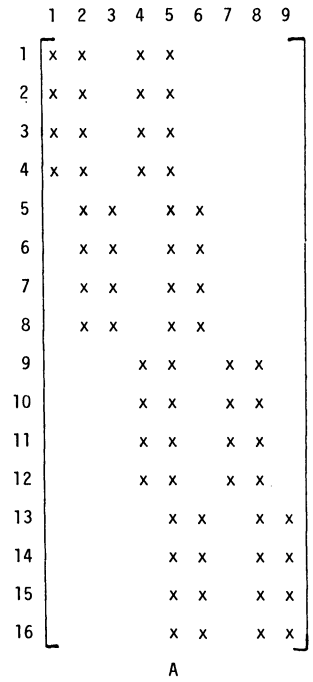
1. backspace the R-file;
2. read the ith row of R and corresponding right-hand side element y_i and compute x_i;
3. backspace the R-file.

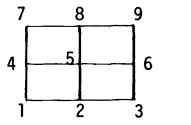
3. Numerical experiments and observations

3.1. Introduction. This section contains results of some experiments performed using an implementation of the algorithms described in § 1.4 and § 2. Our test problems are of various sizes (ranging from 1,444 equations and 400 unknowns up to 17,946 equations and 4,554 unknowns) and of two different types. One class of problems is typical of those that would arise in the natural factor formulation of the finite element method (Argyris and Brönlund [1975]), and the second class typifies those arising in geodetic adjustment problems (Golub and Plemmons [1980]). The descriptions of the problems we provide include only the details necessary to characterize their size and structure; for important information on the physical origins and their mathematical models, the reader should consult the references.

3.2. Test problems. Our finite element test problems are associated with a $q \times q$ grid consisting of $(q-1)^2$ small squares, as shown in Fig. 3.1 with $q = 3$. Associated with each of the $n = q^2$ grid points is a variable, and associated with each small square are four equations (observations) involving the four corner grid points (variables) of the square. Thus, the associated coefficient matrix is $(n = q^2) \times (m = 4(q-1)^2)$, as illustrated in Fig. 3.1.

Our second set of test problems were also derived from a $q \times q$ mesh, but of a rather different type. The purpose here is to construct problems typical of those arising in geodetic adjustments. The mesh can be viewed as being composed of q^2 "junction boxes," connected to their neighbors by chains of length l, as shown in Fig. 3.2 where $q = 3$ and $l = 4$. There are two variables associated with each of the $5q^2 + 2q(q-1)$ $(3(l-1)+1)$ vertices in the mesh, η observations associated with each pair of nodes joined by an edge (involving four variables), and η observations associated with each





Finite element grid

FIG 3.1. *A* 3×3 *finite element grid and its associated* 16×9 *least squares coefficient matrix arising in the natural factor formulation of the finite element method.*

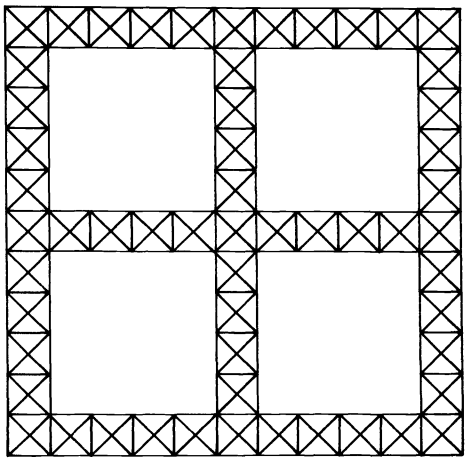


FIG. 3.2. *Idealization of a* 3×3 *geodetic network having connecting survey chains of length* 4.

triangle in the mesh (involving six variables). Thus, the associated least squares problems have $n = 10q^2 + 4q(q-1)(3(l-1)+1)$ variables, and $m = \eta(12q^2 + 2q(q-1)(11l-1))$ observations. In typical real problems, l is around 5 or 6, so we set $l = 5$ in our experiments, yielding $n = 62q^2 - 52q$ and $m = \eta(120q^2 - 108q)$. In our experiments we set $\eta = 2$, yielding $m/n \approx 4$ for large q.

3.3. Numerical experiments. Since one of our main objectives is to provide a means of solving very large problems on computers having limited main storage,

our first experiments involve solving a sequence of problems of increasing size, using a fixed amount of array storage. Of course, as the problems increase in size, the dissection process which provides the partitioning (Algorithm 2 in § 2) must be allowed to proceed further, creating more blocks. The results of these experiments are summarized in Table 3.1. All tests were made on an IBM 3033 at the Oak Ridge National Laboratory. The times reported are in seconds and the storage in words.

TABLE 3.1.

Summary of test results showing storage and excecution times for a sequence of problems, where the maximum total amount of fast storage is fixed at about 30,000 *words.*

1. *Geodetic network problems*

Unknowns n	Equations m	Junction boxes q	Block size limit μ	Number blocks p	Maximum storage used	Factor and solve time in seconds	Total elapsed time in seconds
402	1,512	3	500	1	7,430	1.69	4.69
1,290	4,920	5	1,500	1	25,367	7.21	22.77
2,674	10,248	7	1,500	3	32,070	22.7	57.69
4,554	17,469	9	1,500	7	30,141	46.18	107.40

2. *Finite element grid problems*

Unknowns n	Equations m	Nodes q	Block size limit μ	Number blocks p	Maximum storage used	Factor and solve time in seconds	Total elapsed time in seconds
400	1,444	20	500	1	10,029	3.10	5.97
1,225	4,624	35	1,000	3	30,547	20.78	32.23
2,500	9,604	50	1,000	7	28,462	72.98	96.10
4,225	16,384	65	500	23	28,987	142.81	208.20

The *factor and solve time* includes the orthogonal decomposition time in applying Algorithm 1, together with the final back substitution time in computing the least squares solution. The *total elapsed time* also includes I/O processing and represents the total amount of IBM 3033 time in solving the problem.

The *maximum storage used* includes all array storage, storage for permutations, bookkeeping, etc., in words on the IBM 3033.

Our second set of experiments was designed to demonstrate the influence of the block size limit μ on execution times and total storage requirements. We solved one moderately large-scale problem from each of the two classes a number of times, with different values of μ, leading to different values for the resulting number of blocks p. The results of these experiments are summarized in Table 3.2.

3.4. Observations. 1. In Table 3.1, the factor and solve times, as well as the total elapsed times, increase in a roughly linear fashion as the problem sizes go up under the constraint of fixed maximum amount of in-core storage.

2. For the two problems represented in Table 3.2, the factor and solve times are relatively insensitive to the block size limit μ. These times gradually increase for the geodetic network and oscillate for the finite element grid. Also, the maximum storage used drops considerably at first and then levels out as the block size limit decreases in each problem.

TABLE 3.2

Summary of results for the solution of a single problem from each class, using different levels of blocking.

1. *Geodetic network*

$n = 2{,}674$ unknowns, $m = 10{,}248$ equations

Block size limit μ	Number blocks p	Maximum storage used	Factor and solve time in seconds	Total elapsed time in seconds
3,000	1	53,858	17.73	71.40
1,500	3	32,070	22.27	57.69
800	7	17,966	23.66	52.71
500	18	13,699	27.56	61.80

2. *Finite element grid*

$n = 2{,}500$ unknowns, $m = 9{,}604$ equations

Block size limit μ	Number blocks p	Maximum storage used	Factor and solve time in seconds	Total elapsed time in seconds
2,000	3	73,002	72.58	106.80
1,500	5	46,178	68.73	96.00
1,000	7	28,462	72.98	96.10
500	15	23,332	69.11	100.80

3. The fact that finite element problems generally result in a less sparse observation matrix than geodetic network problems has the obvious result. In each of our tables the storage and execution times are larger for the finite element problems.

4. One possible disadvantage of the use of our automatic blocking program (Algorithm 2) is that the entire structure of A^TA must be initially stored in-core in order to apply the nested dissection scheme. However, in practice this would not often be a serious limitation since some preliminary blocking is usually provided for the larger problems as as a by-product of the modelling procedure (finite element analysis) or data acquisition (geodesy).

5. In each of our problems the time required for the automatic blocking provided by Algorithm 2 turns out to be small in comparison to the total elapsed time. These automatic blocking times are not reported separately in Tables 3.1 and 3.2; however, Algorithm 2 requires only .78 seconds out of a total elapsed time of 208.20 seconds for our largest problem.

6. Our scheme is storage effective. In each test problem the maximum storage used is a modest multiple of the number of variables n. In summary, our numerical experiments demonstrate that the approach taken in this paper can be used to solve large-scale least squares problems using a relatively small amount of in-core storage.

REFERENCES

J. H. ARGYRIS AND O. E. BRÖNLUND [1975], *The natural factor formulation of the stiffness matrix displacement method*, Comput. Meth. Appl. Mech. Engrg., 5, pp. 97–119.

J. H. ARGYRIS, T. L. JOHNSEN AND H.-P. MLEJNEK [1978], *On the natural factor in nonlinear analysis*, ibid., 15, pp. 389–406.

J. K. AVILA AND J. A. TOMLIN [1979], *Solution of very large least squares problems by nested dissection on a parallel processor*, in Proc. Computer Science and Statistics: Twelfth Annual Symposium on the Interface, Waterloo, Ontario, Canada.

A. BJÖRCK [1976], *Methods for sparse linear least squares problems*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, pp. 177–194.

A. BJÖRCK AND I. S. DUFF [1980], *A direct method for the solution of sparse linear least squares problems*, Linear Algebra Appl., 34, pp. 43–67.

I. S. DUFF AND J. K. REID [1976], *A comparison of some methods for the solution of sparse overdetermined systems of linear equations*, J. Inst. Math. Appl., 7, pp. 267–280.

A. GEORGE AND M. T. HEATH [1980], *Solution of sparse least squares problems using Givens rotations*, Linear Algebra Appl., 34, pp. 69–83.

A. GEORGE AND J. LIU [1978], *An automatic nested dissection algorithm for irregular finite element problems*, SIAM J. Numer. Anal. 15, pp. 1053–1069.

——— [1979], *The design of a user interface for a sparse matrix package*, ACM Trans. Math. Software, 5, pp. 134–162.

——— [1980], *Computer Solutions of Large Sparse Symmetric Positive Definite Systems of Linear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey.

A. GEORGE, W. POOLE AND R. VOIGHT [1978], *Incomplete nested dissection for solving n by n grid problems*, SIAM J. Numer. Anal., 15, pp. 662–673.

P. GILL AND W. MURRAY [1976], *The orthogonal factorization of a large sparse matrix*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, pp. 201–212.

G. H. GOLUB [1965], *Numerical methods for solving linear least squares problems*, Numer. Math., 7, pp. 206–216.

G. H. GOLUB AND R. J. PLEMMONS [1980], *Large scale geodetic least squares adjustments by dissection and orthogonal decomposition*, Linear Algebra and Appl., 34, pp. 3–28.

G. H. GOLUB, F. T. LUK AND M. PAGANO [1979], *A sparse least squares problem in photogrammetry*, in Proc. Computer Science and Statistics: Twelfth Annual Symposium on the Interface, Waterloo, Canada.

G. HACHTEL [1976], *The sparse tableau approach to finite element assembly*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, pp. 344–364.

G. B. KOLATA [1978], *Geodesy: Dealing with an enormous computer task*, Science, 200, pp. 421-422, 466.

G. PETERS AND J. H. WILKINSON [1970], *The least squares problem and pseudo-inverses*, Comput. J., 12, pp. 309–316.

P. VANICEK, M. R. ELLIOTT AND R. O. CASTLE [1979], *Four-dimensional modeling of recent vertical movements in the area of the Southern California uplift*, Tectonophysics, 52, pp. 287–300.

R. L. WEIL AND P. C. KETTLER [1971], *Rearranging matrices to block-angular form for decomposition (and other) algorithms*, Management Sci., 18, pp. 98–108.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0006 $01.00/0

THE MULTI-GRID METHOD FOR THE DIFFUSION EQUATION WITH STRONGLY DISCONTINUOUS COEFFICIENTS*

R. E. ALCOUFFE,† ACHI BRANDT,‡ J. E. DENDY, JR† AND J. W. PAINTER†

Abstract. The subject of this paper is the application of the multi-grid method to the solution of

$$-\nabla \cdot (D(x,y)\nabla U(x,y)) + \sigma(x,y)U(x,y) = f(x,y)$$

in a bounded region Ω of R^2 where D is positive and D, σ, and f are allowed to be discontinuous across internal boundaries Γ of Ω. The emphasis is on discontinuities of orders of magnitude in D, when special techniques must be applied to restore the multi-grid method to good efficiency. These techniques are based on the continuity of $D\nabla U$ across Γ. Two basic methods are derived, one in which the approximating finite difference operators on coarser grids are five point operators (assuming the finite difference operator on the finest grid is a five point one) and one in which they are nine point operators.

Key words. multi-grid method, diffusion equation, discontinuous coefficients

1. Introduction. The subject of this paper is the application of the multi-grid method to the solution of

$$\begin{aligned} -\nabla \cdot (D(x,y)\nabla U(x,y)) + \sigma(x,y)U(x,y) &= f(x,y), \qquad (x,y) \in \Omega, \\ \nu(x,y) \cdot D(x,y)\nabla U(x,y) + \gamma(x,y)U(x,y) &= 0, \qquad (x,y) \in \partial\Omega, \end{aligned} \tag{1.1a}$$

where Ω is a bounded region in R^2 with boundary $\partial\Omega$, ν is the outward normal to $\partial\Omega$, D is positive, σ and γ are nonnegative, and D, σ, and f are allowed to be discontinuous across internal boundaries Γ of Ω. We make the natural assumption that

(1.1b) U and $\mu \cdot (D\nabla U)$ are continuous at (x,y) for almost every $(x,y) \in \Gamma$,

where for each $(x,y) \in \Gamma$, $\mu(x,y)$ is a fixed normal vector to Γ. (The "almost every" is necessary to exclude juncture points of Γ, points where two pieces of Γ intersect and where the continuity of $\mu \cdot (D\nabla U)$ does not make sense.)

The emphasis in this paper is on strong discontinuities in D. Indeed, when D has jumps of a factor of up to an order of magnitude across Γ, the multi-grid method as described in [2] works quite well, but when D has jumps of orders of magnitude across Γ, the method of [2] exhibits poor convergence. The primary reason for this failure is as follows. If D jumps by orders of magnitude across Γ, then so also must ∇U in order that (1.1b) be satisfied. Because $D\nabla U$ is the entity that is continuous, it is the error in $D\nabla U$, not the error in ∇U, that is smoothed by relaxation sweeps. In contrast, the method of [2] was really based on the continuity of ∇U and on the smoothing by relaxation sweeps of the error in ∇U. Thus, [2] recommended bilinear interpolation of U, which approximates the continuity of ∇U across Γ. If D jumps by orders of magnitude, then a more appropriate interpolation is one which approximates the continuity of $D\nabla U$ across Γ. (Actually the situation is more complicated than described above. We have found that if Γ consists of many line segments, then even when D jumps by only an order of magnitude across Γ, the method of [2] can perform badly. In this situation the number of discontinuites in D apparently plays a role.)

* Received by the editors May 30, 1980, and in revised form April 27, 1981. This work was supported by the U.S. Department of Energy under contract W-7405-ENG.36.

† Theoretical Division, Los Alamos Scientific Laboratory, Los Alamos, New Mexico, 87545.

‡ Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel.

Grid to grid interpolation is not the only problem. One must also guarantee that for a given grid, the difference equations on the next coarser grid approximate those on the given grid; the design of such difference equations is again based on the continuity of $D\nabla U$, not on that of ∇U, as in [2]. This paper discusses two such designs: one which gives rise to five point difference operators on all grids and one which gives rise to nine point operators on all but the finest grid (assuming, of course, that the finest grid difference operator is a five point one).

The above issues are discussed in more detail in the remainder of this paper. But the main finding of this paper is that the multi-grid method for (1.1) can be made to work with its usual good efficiency.

The application in which we were most interested was the neutron diffusion equation, in which D, σ, and F are piecewise constant in (1.1); indeed all of the numerical examples in § 9 are for this case. However, neither the algorithm nor its efficiency depends on the assumption of piecewise constancy; hence, the paper is posed in terms of (1.1).

The neutron diffusion equation is, of course, of great interest by itself. The application that initially motivated us, however, was the multi-group neutron transport equation (the linear Boltzmann equation in which the energy variation is approximated by coupled transport equations). There is a technique called diffusion synthetic acceleration [1], for accelerating the solution of the multi-group neutron transport equation by solving auxiliary multi-group neutron diffusion equations. This technique requires storage of data for many groups for the transport equation and for its approximating auxiliary diffusion equation. The current code, TTDAMG, which implements this technique has the constraint that the storage for each group not exceed the storage available in the small core memory of a CDC 7600. This constraint puts a premium on storage and dictates some of the decisions we had to make; we indicate, however, when this is the case.

For both TTDAMG and the neutron diffusion equation per se, a large percentage of problems encountered in practice are eigenvalue problems. We, however, defer discussion of eigenvalue problems to a future communication.

An outline of the remainder of this paper is as follows. In § 2 we discuss the difference scheme. In § 3 we review the multi-grid method. In § 4 we discuss the choice of a relaxation scheme. In § 5 we discuss how the difference operators on coarse grids should be defined. In § 6 we give some details of implementation. In § 7 we study a special problem of discretization near a juncture point. In § 8 we discuss some pathological difference equations for which the techniques in this paper are likely to fail. Finally, in § 9 we present some numerical examples.

2. The difference scheme. For simplicity we assume from now on that Ω is a rectangle $(0, X)\times(0, Y)$; neither the algorithm nor its efficiency depends on this assumption. As usual we set up a finite difference grid

$$\{0 = x_0 < x_1 < \cdots < x_n = X\} \times \{0 = y_0 < y_1 < \cdots < y_m = Y\}. \tag{2.1}$$

Define $h_i = x_{i+1} - x_i$, $k_j = y_{j+1} - y_j$. Given such a grid it is natural to assume that

(2.2) the internal interfaces Γ are composed of horizontal and vertical line segments.

We also assume that

(2.3) Γ lies on grid lines.

The difference scheme can now be derived as in [11, pp. 184–190] or [12, pp. 31–38].

We do not want to repeat that derivation here, but we want to comment on how (1.1b) is incorporated into the difference scheme. In the derivation, for each interior (x_i, y_j), one integrates (1.1a) over the four rectangles with common vertex (x_i, y_j) whose union is the rectangle $R_{i,j}$ with vertices

$$\left(x_i - \frac{h_{i-1}}{2}, y_j - \frac{k_{j-1}}{2}\right), \qquad \left(x_i + \frac{h_i}{2}, y_j - \frac{k_{j-1}}{2}\right),$$

$$\left(x_i - \frac{h_{i-1}}{2}, y_j + \frac{k_j}{2}\right), \qquad \left(x_i + \frac{h_i}{2}, y_j + \frac{k_j}{2}\right).$$

One uses Green's theorem to convert the appropriate volume integrals to line integrals. When one sums these four quantities, the interior line integrals cancel ecause of (1.1b) and one is left with line integrals over the boundary of $R_{i,j}$. For $(x_i, y_j) \in \partial\Omega$ one integrates only over rectangles in the interior of Ω. The following difference scheme results when the line integrals are approximated:

$$\begin{aligned} &-A_{i,j+1/2}(U_{i,j+1} - U_{i,j}) - A_{i,j-1/2}(U_{i,j-1} - U_{i,j}) - B_{i+1/2,j}(U_{i+1,j} - U_{i,j}) \\ &\qquad - B_{i-1/2,j}(U_{i-1,j} - U_{i,j}) + C_{i,j}U_{i,j} = F_{i,j}, \qquad 0 \leqq i \leqq n, \quad 0 \leqq j \leqq m, \end{aligned} \tag{2.4}$$

where

$$A_{i,j+1/2} = \frac{1}{2}(D_{i+1/2,j+1/2} + D_{i-1/2,j+1/2})\left(\frac{h_{i-1} + h_i}{2k_j}\right),$$

$$sB_{i+1/2,j} = \frac{1}{2}(D_{i+1/2,j+1/2} + D_{i+1/2,j-1/2})\left(\frac{k_{j-1} + k_j}{2h_i}\right);$$

$$F_{i,j} = \frac{1}{4}(h_{i-1}k_{j-1}f_{i-1/2,j-1/2} + h_{i-1}k_j f_{i-1/2,j+1/2} + h_i k_j f_{i+1/2,j+1/2} + h_i k_{j-1} f_{i+1/2,j-1/2}),$$

(2.5)

$$C_{i,j} = \frac{1}{4}(h_{i-1}k_{j-1}\sigma_{i-1/2,j-1/2} + h_{i-1}k_j\sigma_{i-1/2,j+1/2} + h_i k_j \sigma_{i+1/2,j+1/2} + h_i k_{j-1}\sigma_{i+1/2,j-1/2})$$

$$+ \begin{cases} 0 & \text{if } 0 < i < n, \quad 0 < j < m, \\ \dfrac{k_{j-1} + k_j}{2}\gamma_{i,j} & \text{if } i = 0 \text{ or } n, \\ \dfrac{h_{i-1} + h_i}{2}\gamma_{i,j} & \text{if } j = 0 \text{ or } m, \end{cases}$$

where

$$D_{i+1/2,j+1/2} = D\left(x_i + \frac{h_i}{2}, y_j + \frac{k_j}{2}\right),$$

etc. In (2.5) we employ the conventions that h_l is zero if $l < 0$ or $l \geqq n$, that k_l is zero if $l < 0$ or $l \geqq m$, and that $g(x, y)$ is zero if $(x, y) \notin \Omega$, $g = D$, F or σ; these conventions permit (2.5) to make sense even for $(x_i, y_j) \in \partial\Omega$.

We remark that assumptions (2.2) and (2.3) are not really essential but that if they are not made, one must resort to a homogenization procedure to define the $D_{i+1/2,j+1/2}$'s in the above derivation. In this case smearing of the interfaces occurs; we prefer not to deal with this issue in this paper.

3. A summary of the multi-grid method. In order to describe the special features of the multi-grid method for (1.1), we need to develop some notation and describe the basic philosophy behind the multi-grid method. Hence, suppose that the grid in (2.1) is denoted by G^M with mesh size h_M and that there is a sequence of grids $G^1, \cdots, G^M$ with corresponding mesh sizes $h_1 > \cdots > h_M$. For simplicity we assume that these grids are uniform square grids. This assumption implies no loss in generality since we may rewrite (2.4) on a grid with uniform $h = k$ by redefining the coefficients in (2.5). We further assume that $h_{k+1}/h_k = \frac{1}{2}$ since this ratio of mesh sizes is the most efficient [2].

Now we write (2.4) as

$$L^M U^M = F^M. \tag{3.1}$$

Let us denote an approximate solution of (3.1) by u^M and measure how good an approximation it is by examining the size of the residual $r^M = F^M - L^M u^M$. With usual iteration procedures such as SOR, for example, the first few iterations usually seem to have fast convergence, with residuals rapidly decreasing from one iteration to the next, but then the convergence rate levels off to an asymptotic value and can become very slow. A closer examination in the spirit of [2] shows that convergence is fast as long as the error $V^M = U^M - u^M$ has strong fluctuations on the scale of the grid G^M. When the convergence rate slows down, the error nearly solves the homogeneous equations, and $|r^M|$ is small compared to $|L^M||V^M|$, where $|L^M|$ is, for example, the maximal coefficient of L^M. Hence, when convergence slows down, the error V^M can be approximated on the coarser grid G^{M-1}. One does this by approximately solving the coarse-grid equation

$$L^{M-1} v^{M-1} = f^{M-1} \equiv I_M^{M-1}(F^M - L^M v^M), \tag{3.2}$$

where v^{M-1} is to be the coarse-grid approximation to V^M, $v^M = u^M$ is the last iterate on grid G^M, and where I_M^{M-1} denotes an interpolation operator from G^M to G^{M-1}. When this problem is sufficiently solved, one performs $v^M \leftarrow v^M + I_{M-1}^M v^{M-1}$, where I_{M-1}^M denotes an interpolation operator from G^{M-1} to G^M, and one expects that the v^M thus obtained is a better approximation to U^M than the old v^M. The process is now recursive, of course, since (3.2) is itself solved by relaxation and if convergence slows down for (3.2), one transfers to grid G^{M-2}, etc. On grid G^1, of course, one either solves the G^1 problem directly or iterates until the G^1 problem is sufficiently converged.

There are a number of details that need to be specified in the above description. For example, the operator L^k has so far been defined only for $k = M$, in which case it is given by (2.4); the interpolation operators I_k^p have not yet been defined; and a suitable relaxation scheme has not been selected. These issues are discussed in the following sections. To some extent they are unfortunately intertwined; for example, some remarks in § 4, on relaxation schemes, assume certain characteristics about communication between grids, that is, about L^k and I_k^p; these assumptions become clear, we hope, in § 5.

4. Choice of a relaxation scheme. In [2], [3], and [7] heavy use was made of Fourier analysis in analyzing relaxation schemes. For (2.4), Fourier analysis is not really valid because of the huge jumps in D. A review of some of the conclusions in [2] will be helpful, however. In [2] the equation

$$LU(x, y) = a\frac{\partial^2 U(x, y)}{\partial x^2} + c\frac{\partial^2 U(x, y)}{\partial y^2} = F(x, y) \tag{4.1}$$

is studied. The finite difference form of (4.1) on grid G^k is

$$(4.2)\quad L^k U^k_{\alpha,\beta} = a\frac{U^k_{\alpha+1,\beta} - 2U^k_{\alpha,\beta} + U^k_{\alpha-1,\beta}}{h_k^2} + c\frac{U^k_{\alpha,\beta+1} - 2U^k_{\alpha,\beta} + U^k_{\alpha,\beta-1}}{h_k^2} = F^k_{\alpha,\beta},$$

where $U^k_{\alpha,\beta} = U^k(\alpha h_k, \beta h_k)$, $F^k_{\alpha,\beta} = F^k(\alpha h_k, \beta h_k)$, α, β integers.

Point Gauss–Seidel relaxation consists of scanning the points (α, β) in some order, for example, lexicographic order, and replacing $u_{\alpha,\beta}$ by $\bar{u}_{\alpha,\beta}$ such that (4.2) is satisfied; thus, point Gauss–Seidel is

$$a\frac{u_{\alpha+1,\beta} - 2\bar{u}_{\alpha,\beta} + \bar{u}_{\alpha-1,\beta}}{h_k^2} + c\frac{u_{\alpha,\beta+1} - 2\bar{u}_{\alpha,\beta} + \bar{u}_{\alpha,\beta-1}}{h_k^2} = F^k_{\alpha,\beta}.$$

Let $v = U^k - u$ and $\bar{v} = U^k - \bar{u}$. To study the $\theta = (\theta_1, \theta_2)$ Fourier component of the error functions v and $\bar{v}$ before and after the relaxation sweep, put $v_{\alpha,\beta} = A_\theta e^{i(\theta_1\alpha + \theta_2\beta)}$ and $\bar{v}_{\alpha,\beta} = \bar{A}_\theta e^{i(\theta_1\alpha+\theta_2\beta)}$. Then

$$a(v_{\alpha+1,\beta} - 2\bar{v}_{\alpha,\beta} + \bar{v}_{\alpha-1,\beta}) + c(v_{\alpha,\beta+1} - 2\bar{v}_{\alpha,\beta} - \bar{v}_{\alpha,\beta-1}) = 0,$$

so that

$$(a e^{i\theta_1} + c e^{i\theta_2})A_\theta + (a e^{-i\theta_1} + c e^{-i\theta_2} - 2a - 2c)\bar{A}_\theta = 0.$$

Hence, the convergence factor of the θ component is

$$(4.3)\quad \mu(\theta) = \left|\frac{\bar{A}_\theta}{A_\theta}\right| = \left|\frac{a e^{i\theta_1} + c e^{i\theta_2}}{2a + 2c - a e^{-i\theta_1} - c e^{-i\theta}}\right|.$$

Define $|\theta| = \max(|\theta_1|, |\theta_2|)$. The quantity of interest in the multi-grid method is the smoothing factor

$$\bar{\mu} = \max_{\pi/2 \leqq |\theta| \leqq \pi} \mu(\theta),$$

since $\pi/2 \leqq |\theta| \leqq \pi$ is the range of high frequency components on the grid, that is, the range of components that cannot be approximated on the next coarser grid. For the case $a = c$, $\bar{\mu} = \mu(\pi/2, \arccos\frac{4}{5}) = 0.5$. For the degenerate case $a \ll c$,

$$\mu\left(\frac{\pi}{2}, 0\right) = \left(\frac{a^2 + c^2}{a^2 + (c+2a)}\right)^{1/2}$$

which approaches 1 as $a \to 0$. Hence, for this case, point Gauss–Seidel is not a good relaxation scheme; however, line Gauss–Seidel by lines in y is a good scheme since then

$$\mu(\theta) = \frac{a}{|2(a + c - c\cos\theta_2) - a e^{-i\theta_1}|}$$

and

$$\bar{\mu} = \max\left(5^{-1/2}, \frac{a}{a+2c}\right).$$

In the degenerate case $c \ll a$, line Gauss–Seidel by lines in x is a good relaxation scheme.

Another way of looking at the case, $a \ll c$, is as follows. The points in a given y-line are tightly coupled together, but there is weak coupling between two adjacent y-lines. Line Gauss–Seidel by lines in y relaxes the y-lines simultaneously and essentially reduces to point relaxation in x of blocks of points (the y-lines) for which the smoothing factor is good.

Let us now examine a model problem for (2.4), namely a problem with large constant $D = D_1$ in a central rectangle surrounded by a rectangle with small constant $D = D_2, D_1 \gg D_2$. Clearly point Gauss–Seidel is an acceptable scheme in each of the two regions. The problem is that the points in the D_1 region are strongly coupled compared with their coupling to the D_2 region. Hence, in analogy with the $a \ll c$ case of (1.2) above, the whole block of D_1 points should be relaxed simultaneously. Fortunately, such a drastic measure is not necessary since the D_1 region and its coupling to the D_2 region can be resolved on coarser grids. Hence, it is only on the coarsest grid of all that the strongly coupled block of points, descendents of the D_1 block on the finest grid, needs to be relaxed simultaneously; alternatively, of course, the coarse grid problem can be solved directly. On the other grids point Gauss–Seidel is sufficient. This conclusion has been verified experimentally.

One would like to conclude that for grids with uniform $h = k$, point Gauss–Seidel is always good enough for (2.4). Such is, unfortunately, not the case. Consider a problem in which there are boxes with large D along one of the diagonals of $(0, X) \times (0, Y)$ and in which D is small elsewhere. On coarser grids such a pattern can give rise to a set of strongly coupled points which are coupled together like a one-dimensioal "tail". Depending on how the coarse-grid difference equations are derived, this "tail" may not be represented on the next coarser grid. In such a case it is necessary to use block relaxation in which the "tail" is the block of points relaxed simultaneously. Such a relaxation scheme is not difficult in principle since, just as with line relaxation, only a tridiagonal system needs to be solved for each such block of points. (In case of strongly coupled sets that include two-dimensional "heads" together with one-dimensional "tails" only the "tails" need to be simultaneously relaxed.) It is also not difficult in principle to devise an algorithm to detect the one-dimensional sets of points that are strongly coupled together. We shall refer to the detection and block relaxation of one-dimensional "tails" as SCOD (strongly coupled one-dimensional) block relaxation; SCOD extends point and line Gauss–Seidel to more general situations. We have not implemented SCOD in TTDAMG since the appearance of one-dimensional "tails" in that framework is rare. An additional comment is that identification and relaxation of one-dimensional "tails" does not appear to be very amenable to implementation on a vector machine; perhaps a more attractive possibility for vector machines is local relaxation, in which the convergence is pointwise monitored and only those points (and their neighbors) which have not converged by some criterion are relaxed.

There is a final problem which falls under the heading of choice of a relaxation scheme, and this is concerned with the choice of a relaxation scheme for the coarsest grid. In neutron diffusion problems, it is quite common for σ in (1.1) to be very small in comparison with D. In the initial development of our codes such a problem exhibited the behavior of excruciatingly slow convergence on the coarsest grid. To see why this happens, assume for the moment that σ is constant. Then in (4.3) for the coarsest grid, we have

$$\mu(0, 0) = \frac{2}{2 + \sigma h_1^2}.$$

Thus if σh_1^2 is small, convergence will be slow for the coarsest grid. Actually, this use of local mode analysis is strictly valid only for boundary conditions which are not affected by adding a constant to the solution, since for a low frequency mode like (0, 0), the effects of boundary conditions cannot be ignored. Indeed, for Dirichlet boundary conditions we have never observed this phenomenon of excruciatingly slow convergence; this is because Dirichlet boundary conditions tie down the constant part of

the solution on the coarsest grid. This is not true or the boundary condition in (1.1), however. For example, if $\gamma \equiv 0$, then the addition of a constant will not affect the boundary condition, leaving the difference equation alone to determine the constant part of the solution.

There are two solutions to the above problem. One is to use a direct solution on the coarsest grid. The other, which proved quite useful in another context [4], is as follows. Consider first the case $\gamma \equiv 0$. After each iteration on the coarsst grid, compute a constant c such that $L^1(u^1+c) = f^1$ on the average. That is,

$$c = \frac{\sum (f^1_{i,j} - (L^1 u^1)_{i,j})}{\sum C^1_{i,j}}.$$

Then replace $u^1_{i,j}$ by $u^1_{i,j} + c$.

The above procedure of constant addition worked quite well in the early development stages of our codes. However, we soon found examples for which it did not work well. These examples are characterized by orders of magnitude jumps in D across interfaces, and the problem in these examples is related to one discussed above—at least one subset of coarse grid points has points strongly coupled to themselves and weakly coupled to the remaining grid points, so that the constant addition procedure is not effective. A cure is to identify such subsets and compute a separate constant or each; his is really a type of block relaxation for each subset, and the necessity for doing it for each subset was realized by de la Vallée Poussin (the grandson, not the patriarch) [10] in his quite similar procedure. If the blocks are too large, however, the constant additions will not be enough, and still coarser grids will be needed for fast convergence.

One might think that if the coarsest grid were as coarse as possible (four grid points), then all the grid points would be coupled with equal strength. Such is not the case. It is easy to give an example in which the coarsest grid has four points, three of which are strongly coupled and weakly coupled to the remaining grid point. Thus whether one is using the constant addition algorithm or solving directly for blocks of strongly coupled points, it is necessary to identify subsets of strongly coupled points on the coarsest grid. Rather than implementing an algorithm to identify subsets of strongly coupled points, we resort to direct solution on the coarsest grid; this places a constraint that the coarsst grid be coarse enough that direct solution is inexpensive in both storage and time, but we think that this constraint is not too severe.

In the current version of TTDAMG, the mesh in (2.1) is allowed. For the neutron diffusion equation in one space dimension one can derive [12, p. 72] the criterion for good truncation error that $\sqrt{\sigma/D}h < 1$; this criterion has been shown numerically to be valid also in two space dimensions. The mesh allowed in (2.1) provide at least some flexibility in adapting the mesh to a given pattern of piecewise constant D's, and σ's; typically this results in regions where $h_i \ll k_j$ or where $h_i \gg k_j$. Thus line relaxation is necessary and is, in fact, the basic relaxation scheme for that code. Sometimes it happens that $h_i \gg k_j$ in some region and $h_i \ll k_j$ in another region. This situation requires alternating line relaxation, and there is an option in our codes to use this instead of line relaxation by lines in x or lines in y alone. One could avoid the above situation and use only uniform grids by using local noncoextensive grids [2]; we have implemented this in certain simple situations, but our implementation is as yet incomplete.

5. Communication between grids. The problem remains of choosing interpolation operators I^k_{k-1}, residual weighting operators I^{k-1}_k, and difference operators L^k, $k < M$. In the early development of our codes we used bilinear interpolation for I^k_{k-1}, and we used $I^{k-1}_k = (I^k_{k-1})^*$, theadjoint operator, for residual weighting. For L^{k-1},

$k \leqq M$, we used a fixed weighting of the coefficients of L^k compatible with the weighting I_k^{k-1}, as suggested in [2]. With this approach, problems with orders of magnitude jumps in D (or problems in which Γ consisted of many line segments) exhibited unacceptably slow convergence rates, even slower than straight SOR. For these problems in general, an examination of the l^2-norm of the residuals seemed to indicate that communication between grids was good but that for some reason the smoothing rate was bad; in particular, there seemed to be no reason to suspect th interpolation procedure to be at fault. At this point another approach to generating L^{k-1} from L^k yielded some insight. In the neutron diffusion equation, $D = 1/(3\Sigma_T)$, where Σ_T is the total cross section. Hence, a reasonable approach would seem to be to generate Σ_T^{k-1} from Σ_T^k by using a fixed weighting and to define $D^{k-1} = 1/(3\Sigma_T^{k-1})$. When this was done, one observed in general huge jumps in the l^2-norms of the residuals after interpolation. This observation led to the suspicion that the wrong interpolation procedure was at the root of the problem even though it did not seem to manifest itself with the first way of definig L^{k-1}.

Indeed bilinear interpolation approximates the continuity of U_x and U_y, whereas, when D has jumps of orders of magnitude, (1.1b) makes it more natural to approximate the continuity in x of DU_x and the continuity in y of DU_y, at least at nonjuncture points of Γ. We describe this new interpolation operator, J_{k-1}^k, below. First, however, we present an alternate method of viewing the problem of communication between grids.

This alternate method is motivated by the straightforward application of the multi-grid method to the finite element method. Nicolaides [9] observed in this situation that it is automatically the case that

$$I_k^{k-1} = (I_{k-1}^k)^* \tag{5.1}$$

and that

$$L^{k-1} = I_k^{k-1} L^k I_{k-1}^k, \tag{5.2}$$

where

$$(I_k^{k-1} W^k)_i = ((I_{k-1}^k)^*) W_i^k = \frac{(I_{k-1}^k e_i^k, W^k)}{(e_i^{k-1}, 1)} = \sum_j E_{i,j} W_j^k,$$
$$E_{i,j} = \frac{(I_{k-1}^k e_i^{k-1}, e_j^k)}{(e_i^{k-1}, 1)}. \tag{5.3}$$

(Here i is the index of a point in the $(k-1)$st grid, e_i^k is the function on the kth grid which vanishes at all grid points except the ith point, where its value is 1, and 1 in the denominator of (5.3) is the function which has values 1 at each point of the $(k-1)$st grid. This denominator does not appear in [9], since the equations there are not written in the differential scale. See also the generalization in [3].)

We shall refer to (5.1) and (5.2) as the *automatic prescriptions* for residual weighting and for definition of L^k, $k < M$, respectively. The automatic prescriptions have many desirable features, as we discuss below. These features, however, are a result of the choice of I_{k-1}^k, not of the mechanics per se of performing (5.1) and (5.2). Indeed, if I_{k-1}^k is taken to be bilinear interpolation, the automatic prescriptions result in an algorithm whose performance for problems with orders of magnitude jumps in D is no better than the first algorithm described above. Hence, we turn now to the description of the new interpolation operator, J_{k-1}^k. J_{k-1}^k may be described in terms of Figs. 1 and 2, which show a portion of grids $k-1$ and k respectively. Let $v^k = J_{k-1}^k u^{k-1}$. For coarse-grid points imbedded in the fine grid, the scheme is obvious, i.e., $v_{IF,JF}^k = u_{IC,JC}^k$, etc. Along horizontal lines embedded in the coarse grid, the continuity of DU_x may be

approximated by $B^k_{IF+2,JF}(v^k_{IF+2,JF}-v^k_{IF+1,JF}) = B^k_{IF+1,JF}(v^k_{IF+1}-v^k_{IF,JF})$; i.e.,

$$v^k_{IF+1,JF} = \frac{B^k_{IF+1,JF}u^{k-1}_{IC,JC} + B^k_{IF+2,JF}u^{k-1}_{IC+1,JC}}{B^k_{IF+1,JF} + B^k_{IF+2,JF}}. \tag{5.4}$$

(Note that we have used the indices for the B's that are used in our codes, instead of the notation used in (2.4).)

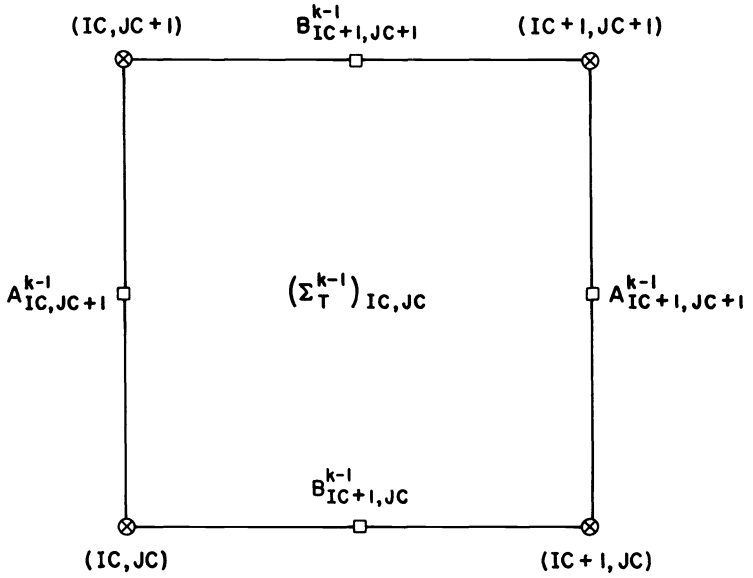


FIG. 1. *Coefficients for grid* G^{k-1}.

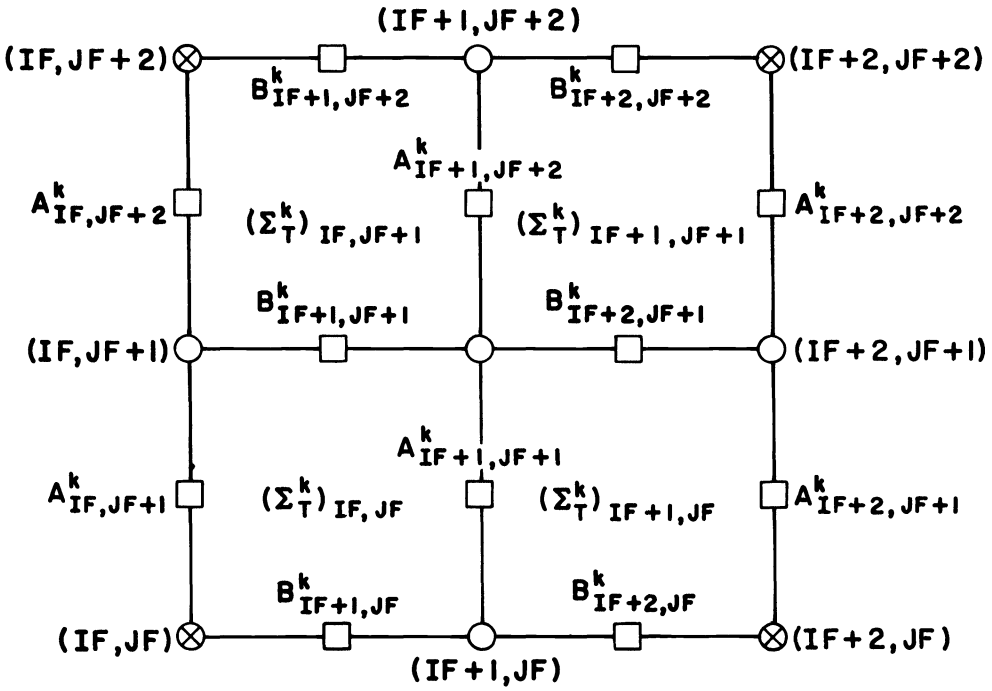


FIG. 2. *Coefficients for grid* G^k.

Similar formulae may be used for vertical lines embedded in the coarse grid. Then, at fine grid points centered in coarse grid squares, $v^k_{IF+1,JF+1}$ may be obtained by

$$v^k_{IF+1,JF+1} = \frac{\begin{aligned}&(B^k_{IF+1,JF+1}v^k_{IF,JF+1} + B^k_{IF+2,JF+1}v^k_{IF+2,JF+1}\\ &+A^k_{IF+1,JF+1}v^k_{IF+1,JF} + A^k_{IF+1,JF+2}v^k_{IF+1,JF+2})\end{aligned}}{(B^k_{IF+1,JF+1} + B^k_{IF+2,JF+1} + A^k_{IF+1,JF+1} + A^k_{IF+1,JF+2})}.$$

Having defined an interpolation operator, we now consider the coarse grid difference operator L^{k-1}. Our first effort at a definition of L^{k-1} was the following: if $\delta(a, b) = 2ab/(a+b)$, then

$$\begin{aligned}B^{k-1}_{IC+1,JC} &= \tfrac{1}{2}\delta(B^k_{IF+1,JF}, B^k_{IF+2,JF})\\ &\quad + \tfrac{1}{4}(\delta(B^k_{IF+1,JF+1}, B^k_{IF+2,JF+1}) + \delta(B^k_{IF+1,JF-1}, B^k_{IF+2,JF-1}))\end{aligned} \tag{5.5a}$$

and

$$\begin{aligned}A^{k-1}_{IC,JC+1} &= \tfrac{1}{2}\delta(A^k_{IF,JF+1}, A^k_{IF,JF+2})\\ &\quad + \tfrac{1}{4}(\delta(A^k_{IF-1,JF+1}, A^k_{IF-1,JF+2}) + \delta(A^k_{IF+1,JF+1}, A^k_{IF+1,JF+2})),\end{aligned} \tag{5.5b}$$

etc.

One way of viewing these formulae is from the point of view of homogenization theory. It is interesting that in special cases, like a two-layer problem, the coefficients in (5.5) coincide with what one would get from homogenization theory [6]. For example, in the definition of the B's it is natural from the point of view of homogenization theory for the harmonic average to appear in the x-direction and an arithmetic average to appear in the y-direction.

The definition of L^{k-1} in (5.5) works well in most situations. One case where it does not work well is in problems with four-corner junctures, that is, points at which a horizontal piece and a vertical piece of Γ cross. In the special case of piecewise constant D, four different values of D meet at such a point. (Given the example of the Four Corners Region in the American Southwest, it is tempting to call a four-corner juncture point a pollution point.) In § 7, these cases are analyzed in more detail; in this section we restrict ourselves to a single example to make plausible the replacement of (5.5) by

$$\begin{aligned}B^{k-1}_{IC+1,JC} &= \tfrac{1}{2}\delta(B^k_{IF+1,JF}, B^k_{IF+2,JF})\\ &\quad + \tfrac{1}{4}(\delta(A^k_{IF,JF+1}, B^k_{IF+1,JF+1}, B^k_{IF+2,JF+1}, A^k_{IF+2,JF+1})\\ &\qquad + \delta(A^k_{IF,JF}, B^k_{IF+1,JF-1}, B^k_{IF+2,JF-1}, A^k_{IF+2,JF}))\end{aligned} \tag{5.6a}$$

and

$$\begin{aligned}A^{k-1}_{IC,JC+1} &= \tfrac{1}{2}\delta(A^k_{IF,JF+1}, A^k_{IF,JF+2})\\ &\quad + \tfrac{1}{4}(\delta(B^k_{IF+1,JF}, A^k_{IF+1,JF+1}, A^k_{IF+1,JF+2}, B^k_{IF+1,JF+2})\\ &\qquad + \delta(B^k_{IF,JF}, A^k_{IF-1,JF+1}, A^k_{IF-1,JF+2}, B^k_{IF,JF+2})),\end{aligned} \tag{5.6b}$$

etc., where

$$\delta(a, b, c, d) = \frac{1}{\frac{1}{4}\left(\frac{1}{a}+\frac{1}{b}+\frac{1}{c}+\frac{1}{d}\right)}.$$

These formulae are already plausible from electrical network arguments. For example, there are three paths from $(IF, JF+1)$ to $(IF+2, JF+1)$; one through $(IF+1, JF+1)$; one through $(IF, JF+2)$, $(IF+1, JF+2)$, and $(IF+2, JF+2)$; and one through (IF, JF), $(IF+1, JF)$, and $(IF+2, JF)$. The diffusion coefficient for each path should be computed by harmonic averages since each path has elements connected in series; the diffusion coefficient of the three paths should be computed by an arithmetic average since the paths are connected in parallel. Other paths can be added to the scheme, but (5.6) seems good enough for the usual cases. At least in one important case (5.6) is better than (5.5); this is the case of two layers with coefficients D_1 and D_2, $D_1 \gg D_2$, where their internal boundary Γ coincides with a fine grid line which is not a coarse grid line. In such a case the difference equations obtained by (5.5) on the coarser grid would break up the strongly coupled region, creating a disconnected line of strongly coupled points (as also in Fig. 4 below) thus requring line (or more generally SCOD) relaxation. Such troubles are avoided by (5.6) (as in Fig. 5 below).

Consider now the problem with a grid point at a single four-corner juncture, the intersection of $x = X/2$ with $y = Y/2$. Assume that D is piecewise constant with values DNW, DNE, DSW, and DSE in the northwest, northeast, southwest, and southeast quadrants with respect to $(X/2, Y/2)$. Assume specifically that $DNW = DSE = 1$ and that $DNE = DSW = \varepsilon \ll 1$ and assume a uniform mesh with mesh lines passing through $x = X/2$ and $y = Y/2$. Let $(X/2, Y/2) = (x_k, y_l)$ for some k and l. Then in (2.4)

$$A_{k-1/2,l} = A_{k+1/2,l} = B_{k,l-1/2} = B_{k,l+1/2} = \tfrac{1}{2}. \tag{5.7}$$

The first issue, then, is one of discretization of the finest grid itself since the discretization (5.7) is clearly incorrect for the juncture point (x_k, y_l). It is incorrect since the northwest and southeast region should be insulated from each other—there should be only $O(\varepsilon)$ diffusion through their single point of contact (x_k, y_l); however, with (5.7) holding, there quite clearly is strong diffusion from the northwest to the southeast region. (This is discussed further in § 7.) A better discretization would be to change (5.7) to

$$A_{k-1/2,l} = A_{k+1/2,l} = B_{k,l-1/2} = B_{k,l+1/2} = \varepsilon. \tag{5.8}$$

In the discussion that follows we will assume (5.8). It is possible to deal with (5.7) with considerably more effort, but since it is a worse discretization, the effort is not worth the payoff.

There are two cases to consider for this example. The first is that the next coarser grid also has grid lines along the interfaces $x = X/2$ and $y = Y/2$. In this case with either (5.5) or (5.6), on the next coarser grid the northwest region stays insulated from the southeast region, and the connections between the northeast and southwest regions remain weak. The second case, in which the next coarser grid does not have grid lines along the interfaces $x = X/2$ and $y = Y/2$, is depicted in Fig. 3. The coarse grid is indicated with crosses, and points that are strongly coupled on the fine grid have heavy lines connecting them. When (5.5) is used to generate the coarse grid coefficients, the situation is as depicted in Fig. 4. Note that the region which should have only weak connections is broken up by two staircases of strongly connected points. This situation is bad for two reasons. First, the coarse grid does not provide a good approximation to the fine grid, and econd, one-dimensional "tails" of strongly coupled points, for which SCOD block relaxation is necessary (as warned in § 4), have arisen where they have no right to be. Figure 5 depicts what happens when (5.6) is used to generate the coarse grid coefficients; in this case the coarse grid is a good approximation to the fine grid, and the "tails" do not appear.

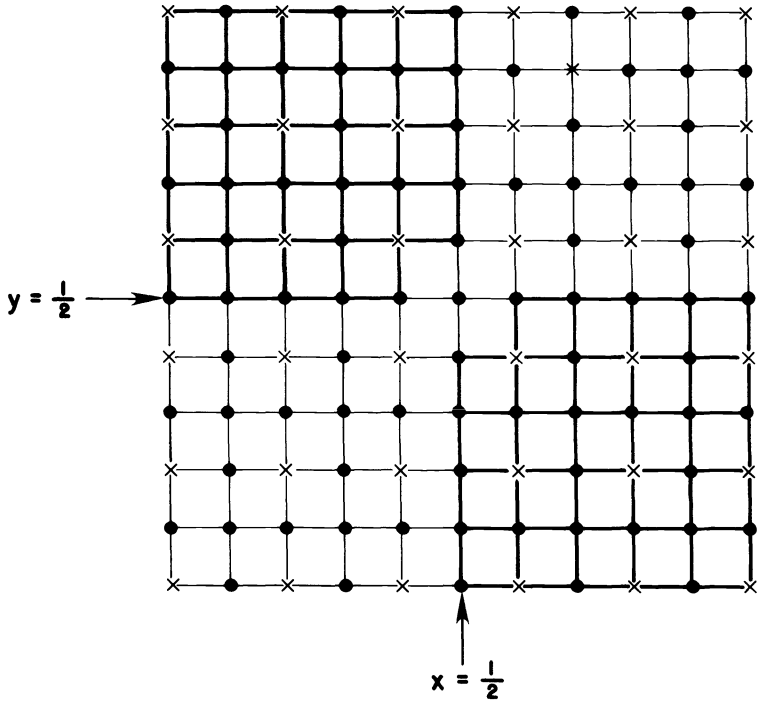


FIG. 3. *Pattern of connections when* (5.8) *is used.*

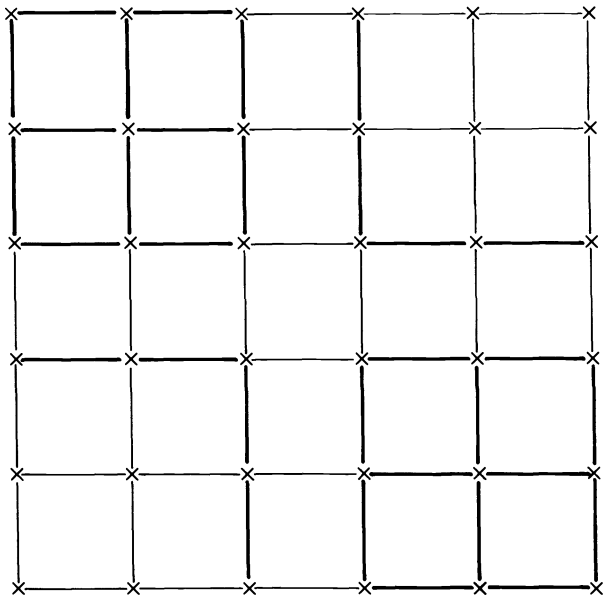


FIG. 4. *Pattern of connections on next coarser grid when* (5.5) *is used.*

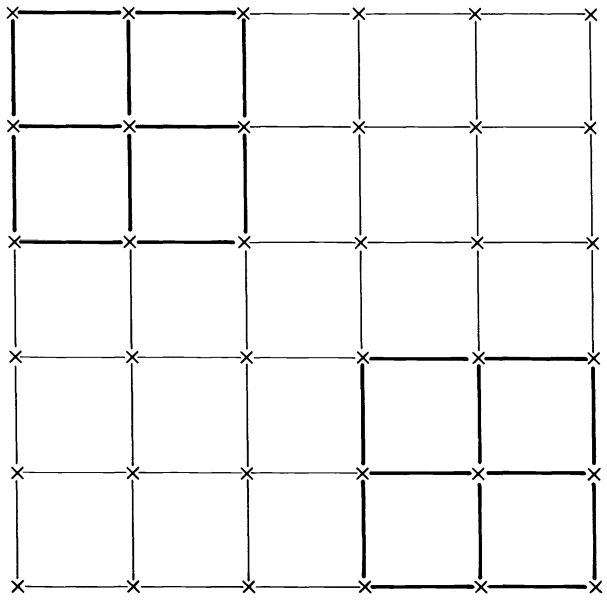


FIG. 5. *Pattern of connections on next coarser grid when* (5.6) *is used.*

Another scheme, which retains most of the good features of (5.6) but which is cheaper computationally (and is therefore the scheme currently implemented in TTDAMG), is the following. (See Figs. 1 and 2.)
Define

$$
\begin{aligned}
(\Sigma_T^{k-1})_{IC,JC} &= \tfrac{1}{4}(\Sigma_T^k)_{IF,JF+1} + (\Sigma_T^k)_{IF+1,JF+1} + (\Sigma_T^k)_{IF,JF} + (\Sigma_T^k)_{IF+1,JF},\\
B_{IC+1,JC+1}^{k-1} &= \tfrac{1}{2}([\tfrac{1}{2}(3(\Sigma_T^k)_{IF,JF+1} + 3(\Sigma_T^k)_{IF+1,JF+1})]^{-1}\\
&\qquad + [\tfrac{1}{2}(3(\Sigma_T^k)_{IF,JF+2} + 3(\Sigma_T^k)_{IF+1,JF+2})]^{-1});
\end{aligned}
\tag{5.9}
$$

$A_{IC+1,JC+1}^{k-1}$ is defined similarly. These formulae are for uniform spacing; for nonuniform spacing, they are modified by using area weighting. For the example shown in Figs. 4 and 5, this scheme gives the pattern in Fig. 5. Since the fine grid coefficients are defined by (2.5), one must still modify the fine grid coefficients by (5.8) for that problem.

The final scheme we consider for defining L^{k-1} is the automatic one (5.2) but where I_{k-1}^k is taken to be J_{k-1}^k and I_k^{k-1} is taken to be $(J_{k-1}^k)^*$, i.e.,

$$
L^{k-1} = (J_{k-1}^k)^* L^k J_{k-1}^k. \tag{5.10}
$$

The motivation for (5.10) is quite natural, since given the interpolation J_{k-1}^k, L^{k-1} is precisely the G^{k-1} operator that results from solving the minimization of a discrete approximation to a Dirichlet integral for the correction problem on G^{k-1}. (This is discussed in [2, § A.5], but in that context the interpolation is the natural one for the finite element method.) We discuss the relative merits of (5.6), (5.9), and (5.10) in §§ 6 and 9; however, two obvious disadvantages of (5.10) are as follows. The first is that L^{k-1} as defined by (5.10) is a nine point operator, while the other schemes discussed retain five point operators on the coarser grids. The second is that the implementation of (5.10) is nontrivial and in addition is very inefficient unless one can afford to store J_{k-1}^k.

If one is not using (5.10), then a related question concerns the automatic prescription of J_k^{k-1} as $(J_{k-1}^k)^*$. There is a plausible argument for this choice. Note that the

interpolation operator has exactly the features that are desirable. For example, consider (5.4) and suppose that $B^k_{IF+1,JF} \gg B^k_{IF+2,JF}$; then $u^{k-1}_{IC,JC}$ is weighted much more strongly in the interpolation than $u^{k-1}_{IC+1,JC}$. This same behavior should be true for J^{k-1}_k; that is, if we have a source (i.e., residual—the source for the correction problem) more strongly connected to one coarse grid point than to another, then the flow from the source to that coarse grid point will be proportionately stronger; hence, it will act as a (proportionately) stronger source at that point than at the other. $(J^k_{k-1})^*$ has exactly this property. If one is able to store J^k_{k-1}, then there is no reason not to take $J^{k-1}_k = (J^k_{k-1})^*$. However, in TTDAMG we cannot afford to store J^k_{k-1}. Since $(J^k_{k-1})^*$ involves divides (as does, of course, J^k_{k-1}) and since on the CDC 7600, a divide takes as much time as four multiplies, the question arises of taking $J^{k-1}_k = (I^k_{k-1})^*$. We have tried both choices, and there is not a significant difference. For some problems $J^{k-1}_k = (J^k_{k-1})^*$ is even slightly worse than $J^{k-1}_k = (I^k_{k-1})^*$ assuming equal work for both; when the work to do each is considered, $J^{k-1}_k = (I^k_{k-1})^*$ wins easily. Of course, when (5.10) is used, and J^k_{k-1} is stored, then $J^{k-1}_k = (J^k_{k-1})^*$ is the natural choice.

Given the lack of evidence for the indictment of bilinear interpolation in the first scenario described in this section, one naturally asks at this point if one can dispense with J^k_{k-1} and use bilinear interpolation in conjunction with the L^k's, $k < M$ generated by (5.6). Certainly if D jumps by orders of magnitude, then bilinear interpolation fails to be second order, as recommended in [2], but this failure is only at interfaces; hence, perhaps bilinear interpolation is good enough. It is not. Numerical experiments show it to be good enough when there are a small number of interfaces and when D does not jump by orders of magnitude. When either of these conditions is violated, bilinear interpolation can lead to divergence in the fixed mode of multi-grid or to very slow convergence in the accommodative mode. (By fixed mode, we mean that a predetermined number of relaxation sweeps is taken on each grid in each cycle, a cycle consisting of proceeding from the finest grid to the coarsest grid and back to the finest grid. The accommodative mode is described in [2].) In both modes the l^2 norm of the residuals exhibits orders of magnitude jumps after interpolation, but the accommodative algorithm accommodates by relaxing until the high frequency errors from interpolation have been smoothed out.

Up to this point, we have tacitly assumed that the interfaces in D were more important than the interfaces in σh^2. This assumption is good if the criterion for good truncation error $\sqrt{\sigma/D}h < 1$ of [12, p. 72] is satisfied. However, even if this criterion is satisfied on the finest grid, it need not be satisfied on the coarser grids, where $\sigma^k (h^k)^2$ can easily dominate the $A^k_{i,j}$'s and $B^k_{i,j}$'s. In such cases it is not surprising that a fixed weighting of the $C^k_{i,j}$'s to obtain C^{k-1} fails to work well. If one were using (5.10), then it would seem natural to define $C^{k-1} = (J^k_{k-1})^* C^k J^k_{k-1}$; however, with J^k_{k-1} based just on the diffusion coefficients A^k and B^k, even this choice does not always lead to good convergence. It seems fairly clear that on coarser grids, it is important to incorporate σ into the interpolation J^k_{k-1}. For example, consider the case that σh^2 dominates D locally on even the finest grid. Then (5.4) can be inaccurate if there are interfaces in D and σ, for $u^{k-1}_{IC,JC}$ and its weight can be large, while if $\sigma_{IF+1,JF} h^2$ is large, then $v^k_{IF+1,JF} \simeq 0$ is clearly better than (5.4) in this case.

The problem is how to incorporate σ into the interpolation. This problem is easily solved in one space dimension, where one has a tridiagonal operator and can use the operator to do interpolation. In two space dimensions the problem is trickier. For example, since $C^{k-1} = (J^k_{k-1})^* C^k J^k_{k-1}$ is a nine point operator, it is not clear how to separate out its x- and y-dependence. (The astute reader may have already noticed that this is already potentially a problem with (5.4), which ignores the cross-coupling

coefficients.) One solution is to integrate in y if the x-dependence is desired and to integrate in x if the y-dependence is desired; this gives rise to the following scheme (given for a uniform mesh). Suppose L^k has the pointwise template

$$\begin{bmatrix} -T^k_{IF,JF+1} & -W^k_{IF,JF+1} & -R^k_{IF+1,JF+1} \\ -Q^k_{IF,JF} & S^k_{IF,JF} & -Q^k_{IF+1,JF} \\ -R^k_{IF,JF} & -W^k_{IF,JF} & -T^k_{IF+1,JF} \end{bmatrix}.$$

Form $\tilde{Q}^k_{IF+1,JF} = T^k_{IF+1,JF+1} + Q^k_{IF+1,JF} + R^k_{IF+1,JF}$, $\tilde{S}^k_{IF+1,JF} = -W^k_{IF+1,JF} + S^k_{IF+1,JF} - W^k_{IF+1,JF+1}$, and $\bar{Q}^k_{IF+2,JF} = T^k_{IF+2,JF} + Q^k_{IF+2,JF} + R^k_{IF+2,JF+1}$. Then the formula corresponding to (5.4) is

$$v^k_{IF+1,JF} = \frac{\tilde{Q}^k_{IF+1,JF} u^{k-1}_{IC,JC} + \bar{Q}^k_{IF+2,JF} u^{k-1}_{IC+1,JC}}{\tilde{S}^k_{IF+1,JF}}.$$

Similar formulae may be used for vertical lines embedded in the coarse grid. Then, at fine grid points centered in coarse grid squares, $v^k_{IF+1,JF+1}$ may be obtained by

$$v^k_{IF+1,JF+1} = \frac{\begin{aligned} &Q^k_{IF+1,JF+1} v^k_{IF,JF+1} + Q^k_{IF+2,JF+1} v^k_{IF+2,JF+1} + W^k_{IF+1,JF+1} v^k_{IF+1,JF} \\ &+ W^k_{IF+1,JF+2} v^k_{IF+1,JF+2} + R^k_{IF+1,JF+1} v^k_{IF,JF} + R^k_{IF+2,JF+2} v^k_{IF+2,JF+2} \\ &+ T^k_{IF+1,JF+2} v^k_{IF,JF+2} + T^k_{IF+2,JF+1} v^k_{IF+2,JF} \end{aligned}}{S^k_{IF+1,JF+1}}.$$

We will refer to this new interpolation operator as $\tilde{J}^k_{k-1}$ and for reference display the following definition of L^{k-1} using $\tilde{J}^k_{k-1}$:

$$L^{k-1} = (\tilde{J}^k_{k-1})^* L^k \tilde{J}^k_{k-1}. \tag{5.11}$$

The numerical examples in § 9 all have $\sigma \equiv 0$; however, in subsequent work [8], these examples were repeated with nonzero σ.

We have not yet mentioned the full multi-grid algorithm for (2.4). For smooth equations like Poisson's equation it is possible, using the full multi-grid algorithm [5], to solve to within truncation error in work $O(n)$, where n is the number of grid points (without the full multi-grid algorithm, the work is $O(n \log n)$); more precisely the $O(n)$ is five to seven work units. In this process one begins on the coarsest grid and bootstraps his way up with each grid providing a good initial guess for the next finer grid; it can be shown [5] that in this process cubic interpolation should be used when a grid is visited for the first time. For (2.4), cubic interpolation is not appropriate, but the following interpolation, described for the one-dimensional case, can take its place. In Fig. 6, we

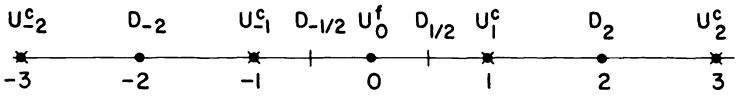


FIG. 6. *Generalization of cubic interpolation.*

know U^c_{-2}, U^c_{-1}, U^c_1, and U^c_2, and we want to interpolate a value U^f_0. We know the coarse grid diffusion coefficients D_{-1} and D_2 and the fine grid coefficients $D_{-1/2}$ and

$D_{1/2}$. We can form approximations to DU_x at -2, $-1/2$, $1/2$, and 2, namely

$$C_{-2}=\frac{D_{-2}}{2}(u^c_{-1}-u^c_{-2}), \qquad C_{-1/2}=D_{-1/2}(u^f_0-u^c_{-1}),$$

$$C_{1/2}=D_{1/2}(u^c_1-u^f_0), \qquad C_2=\frac{D_2}{2}(u^c_2-u^c_1),$$

and ask that a parabola go through C_{-2}, $C_{-1/2}$, $C_{1/2}$, and C_2. There are four degrees of freedom—the three coefficients of the parabola and u^f_0 and four constraints, and the problem can be solved; indeed, the result is

$$(5.12)\quad u^f_0=\frac{1}{D_{-1/2}+D_{1/2}}\left(-\frac{D_{-2}}{8}u^c_{-2}+\left(D_{-1/2}+\frac{D_{-2}}{8}\right)u^c_{-1}+\left(D_{1/2}+\frac{D_2}{8}\right)u^c_1-\frac{D_2}{8}u^c_2\right).$$

This process can be viewed as a generalization of (5.4); there one can fit a constant through the approximation to DU_x at two points by adjusting the desired interpolated value $v^k_{IF+1,JF}$. (Also, it is easy to see that for constant D, (5.11) reduces to cubic interpolation.) Experimental results subsequent to the first version of this paper seem to indicate that (5.12) pays off only when there are no juncture points and then only if one desires better accuracy in the derivatives of the solution. These results also indicate that the full multi-grid algorithm with $\tilde{J}^k_{k-1}$ replacing (5.12) does solve to truncation error in one full multi-grid cycle; this is exploited in [8].

6. Some details of implementation. In this section we discuss some details of implementation, many of which are specific to our application. The first is the issue of storage. On each grid the storage of u^k, A^k, B^k, C^k, and F^k requires five locations per grid point. Since we do line relaxation by lines in x, by lines in y, or both, there is a collection of tridiagonal systems to solve on every relaxation sweep. It is clearly inefficient to factor these each relaxation sweep. It pays to factor each as LU, L lower triangular bidiagonal, U upper unit triangular bidiagonal. To save on divides one can compute and store L^{-1}. Thereafter, relaxation by lines in x [y] is no more expensive than point Gauss–Seidel; this observation, of course, is not new [11, p. 199], [12, p. 22].

The additional storage is one location per grid point [12, p. 22]; however, one may use the C^k array for this purpose since it is no longer needed. The reason for this is that the diagonal of LU contains the information from C^k; and in the only other place where C^k might be needed, the calculation of residuals, C^k can also be dispensed with. To see the latter, write line relaxation by lines in x, for new values $\bar{u}$ in terms of old values u, as follows:

$$(A^k_{i,j+1/2}+A^k_{i,j-1/2})\bar{u}^k_{i,j}-B^k_{i+1/2,j}(\bar{u}^k_{i+1,j}-\bar{u}^k_{i,j})-B^k_{i-1/2,j}(\bar{u}^k_{i-1,j}-\bar{u}^k_{i,j})+C^k_{i,j}\bar{u}^k_{i,j}$$

$$=F^k_{i,j}+A^k_{i,j+1/2}u^k_{i,j+1}+A^k_{i,j-1/2}\bar{u}^k_{i,j-1}.$$

The (i,j)th residual error in $\bar{u}$ after all lines have been swept, is

$$r_{i,j}=F^k_{i,j}+A^k_{i,j+1/2}(u^k_{i,j+1}-\bar{u}^k_{i,j})+A^k_{i,j-1/2}(\bar{u}^k_{i,j-1}-\bar{u}^k_{i,j})+B^k_{i+1/2,j}(\bar{u}^k_{i+1,j}-\bar{u}^k_{i,j})$$

$$(6.1)\qquad +B^k_{i-1/2,j}(\bar{u}^k_{i-1,j}-\bar{u}^k_{i,j})-C_{i,j}\bar{u}^k_{i,j}+A^k_{i,j+1/2}(\bar{u}^k_{i,j+1}-u^k_{i,j+1})$$

$$=0+A^k_{i,j}(\bar{u}^k_{i,j+1}-u^k_{i,j+1}).$$

Thus one can dispense with storage for C^k, and the total storage needed is 5 locations per grid point (6 if alternating line relaxation is used) plus a little extra for the direct solution on the coarsest grid. The above calculation also shows that the computation of

the l^2 error and of the residuals for I_k^{k-1} is extremely cheap and can be performed during the relaxation sweep. Note that an expression analogous to (6.1) can be written down for point relaxation, enabling one to compute the residuals during the relaxation sweep for an extra two multiplies per grid point as opposed to the one multiply that (6.1) requires.

Another issue is the possible storage of the interpolation operators J_{k-1}^k. Actually this storage requirement is not bad—6 locations per G^{k-1} grid point or $1\frac{1}{2}$ per G^k grid point. It is thus an indication of how strapped for storage we are in TTDAMG when we cannot afford to store J_{k-1}^k. When the constraint on storage mentioned in § 1 is removed, however, the storage of J_{k-1}^k will become possible.

It is sometimes the case that the problem being solved is so large that the whole finest grid and its associated unknowns will not fit in memory at the same time. In such a case only a number of lines, say x-lines, of the mesh may reside in memory at once. Buffering the mesh in and out of memory once for each relaxation sweep, once for each interpolation, and once for transfer of residuals is extremely inefficient. For this situation we propose that one should instead perform these operations in a wave, as follows. For a given set of x-lines one can perform interpolation from the next coarser grid. These x-lines are now ready for a relaxation sweep, and once a relaxation sweep has been performed on a given x-line, the next relaxation sweep may be performed on the x-line below it. Once the relaxation sweeps have been performed, the residual transfer to the next coarser grid may be performed for the next lower x-line. The whole wave of operations requires the finest grid to be buffered in only once. (Similar considerations apply if even some of the coarser grids are too large to fit in memory. Using this wave of operations idea and the full multi-grid algorithm, one can obtain a solution to the level of truncation error in a process that includes only one visit (i.e., one wave of interpolation, two sweeps and residual transfer) to the finest grid. This can then be programmed without even storing the finest grid, not even externally [2]. Further reduction of required storage without using external storage is possible (segmental refinement).

The above wave of operations process is efficient for Gauss–Seidel by lines in x, but what about Gauss–Seidel by lines in y? One solution is to relax by shorter lines in y, overlapping the lines. Another suggestion is to use weighted line Jacobi relaxation by lines in y. In this case one has a collection of tridiagonal systems $A_i x_i = f_i$ to solve. Assume for the moment that A_i has been decomposed into $L_i U_i$ and that L_i^{-1} has been computed. Because L_i^{-1} is lower triangular, $L_i^{-1} f_i$ can be performed several x-lines at a time, for every i, starting at the bottom of the mesh. Then $U_i x_i = L_i^{-1} f_i$ can be solved several x-lines at a time for every i starting at the top of the mesh. Once the sweep back down is started, transfer of residuals can be performed. For alternating line relaxation, one can perform the wave of operations consisting of interpolation, relaxation by lines in x, computation of $L_i^{-1} f_i$ for all i on the way up, then solution of $U_i x_i = L_i^{-1} f_i$ for all i and transfer of residuals on the way down. The whole wave requires two bufferings of the finest grid into memory. Note, however, that only one line Jacobi relaxation by lines in y can be performed in these two bufferings. Note also the attractiveness of weighted line Jacobi relaxation for vector machines.

A final issue of implementation is whether L^k, $k < M$ is defined by (5.6), (5.9), or (5.11). If it is assumed that J_{k-1}^k is stored, 17 multiplies per fine grid point (22 if the fine grid operator is also a nine point operator) are required for (5.11). This sounds expensive, but actually (5.6) requires 15 multiplies per fine grid point because of the necessary divides (with a divide counted as four multiplies). Procedure (5.9) is the cheapest, requiring only 7 multiplies per fine grid point. Since a point (or line)

Gauss–Seidel sweep on the finest grid requires five multiplies, perhaps none of (5.6), (5.9), or (5.11) should be viewed as very expensive. Procedure (5.11) does, however, give nine point operators on the coarser grids, requiring twice as much work per relaxation sweep on the coarser grids.

7. The four-corner juncture. Consider the case where the internal boundaries Γ contain two straight segments Γ_1 (horizontal) and Γ_2 (vertical) intersecting at a point, which we will call a juncture. The neighborhood of the juncture is thus divided into four quadrants Q_1, Q_2, Q_3 and Q_4 (see Fig. 7). Let the diffusion coefficient at Q_i be D_i. We

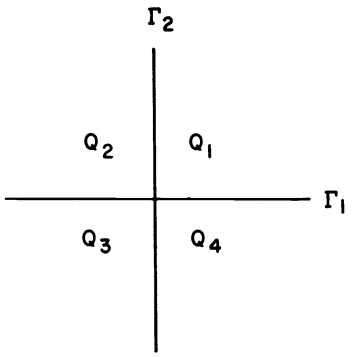


FIG. 7. *Four-corner juncture problem.*

are interested in analyzing the discretization and multi-grid procedures in the case where

$$\varepsilon = \frac{\max(D_2, D_4)}{\min(D_1, D_3)} \tag{7.1}$$

is very small. This case is of special interest since it implies a worse-than-usual singularity. Usually, when D is discontinuous, the solution exhibits discontinuous *derivatives*. When $\varepsilon \to 0$, however, the solution at the juncture becomes discontinuous itself. Indeed, when $\varepsilon \to 0$ the solutions at Q_1 and at Q_3 become completely independent of each other: Each tends to a solution of an independent boundary-value problem, with homogeneous Neumann boundary conditions along Γ_1 and Γ_2.

It is well known that near a severe singularity special care should be taken in writing the discrete equations. Similar care is needed in transferring the equations from the finest grid to coarser ones. While a bad discretization (on a fixed grid), which results in a bad approximation, is often unnoticed (since the true solution is not known), a bad discretization on coarser grids in a multi-grid context (e.g., a bad fine-to-coarse transfer of diffusion coefficients) often manifests itself in a slower multi-grid convergence. This is an advantage of multi-grid processes: they clearly expose discretization troubles.

The discretization of the diffusion equations (2.4) around a juncture is a point in case. Slower multi-grid rates led us to discover a certain conceptual defect in the discretization method. The discretization method and the fine-to-coarse transfer of diffusion coefficients should therefore be studied simultaneously.

Assume the grid is chosen so that Γ_1 and Γ_2 are grid lines; hence, the juncture is a grid point. (This is presumably a desirable choice, as placing knots at discontinuities usually yields better approximations.) Assume for simplicity that Γ_1 is the x axis, that Γ_2 is the y axis, hence that the juncture is at the origin, and that $D_1 = D_3 = 1$ and

$D_2 = D_4 = \varepsilon$. The differencing scheme of § 2 would then give discrete diffusion coefficients as described in Fig. 8. These same coefficients will also appear on coarser grids, as long as Γ_1 and Γ_2 are still grid lines, whether we produce the coarse-grid coefficients directly from the differential ones by (2.5) or we transfer them from finer grids by (5.5), (5.6), or (5.9).

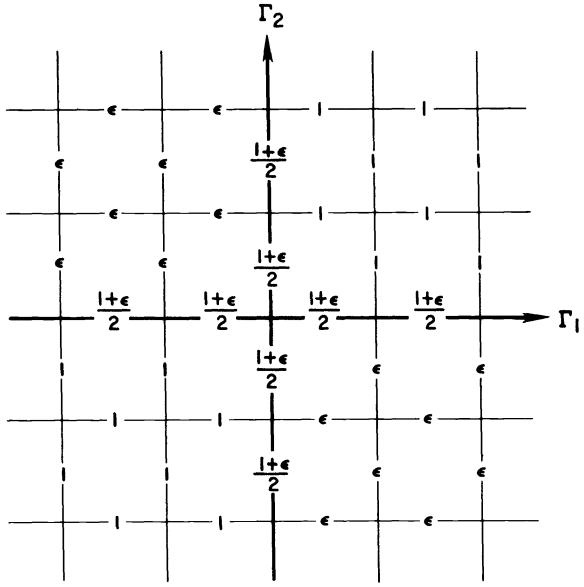


FIG. 8. *Diffusion coefficients for four-corner juncture problem.*

Numerical experiments with this discretization-coarsification method exhibit slower multi-grid convergence rates. See for example Table 3 in § 9. There is indeed an obvious weakness in this discretization: In the limit $\varepsilon \to 0$ the quadrants Q_1 and Q_3 become physically completely insulated from each other, whereas in the discrete scheme they remain coupled.

More precisely, in the limit the solution in the vicinity of the juncture tends to one value C_1 in the quadrant Q_1 and to another value C_3 in Q_3. In particular

$$U(h, 0) \approx U(0, h) \approx U(h, h) \approx U(2h, 0) \approx \cdots \approx C_1$$

while $U(0, 0)$ is not well defined. In our numerical scheme, on the other hand, we will get

$$U^h(0, 0) \sim \frac{C_1 + C_3}{2},$$

$$\frac{U^h(jh, 0) - U^h(jh - h, 0)}{U^h(jh + h) - U^h(jh)} \sim \frac{2j + 1}{2j - 1}, \qquad j \geqq 2 \tag{7.2}$$

and hence

$$\frac{U^h(jh, 0) - U^h(0, 0)}{U^h(kh, 0) - U^h(0, 0)} \sim \frac{1 + 0.66 \log j}{1 + 0.66 \log k}, \qquad j \geqq 1. \tag{7.3}$$

(The approximate coefficient 0.66 is computed numerically.) The discontinuity at the juncture is smeared by the scheme. There is thus an $O(1)$ error in U^h at points $O(h)$ distant from the juncture. Similarly, in the multi-grid process, there is an $O(1)$ error in the approximation to U^h provided by U^{2h} around the juncture. This results in a slower multi-grid convergence rate.

An obvious and simple cure is to avoid the coupling between Q_1 and Q_3 by using as our discrete diffusion coefficients around the juncture

$$D^h_{1/2,0} = D^h_{0,1/2} = D^h_{-1/2,0} = D^h_{0,-1/2} = \alpha^h, \tag{7.4}$$

where

$$\alpha^h \ll 1, \qquad \alpha^{2h} \ll 1, \cdots. \tag{7.5}$$

The precise values are not important. For this purpose it is enough to set $\alpha^h \ll 1$ and use the scheme (5.6) to produce the diffusion coefficients on coarser grids. Such a procedure has actually been implemented, with satisfactory results. See Table 3 in § 9.

Observe that using the transfer scheme (5.5) instead of (5.6) will not do here. Even with $\alpha^h \ll 1$, it will produce α^{2h} comparable to 1. Unlike (5.6), it does not preserve decoupling: blocks which are decoupled in the fine grid may become coupled in the coarser ones.

8. Pathological difference equations. It may be important to notice that the methods described above, fully efficient as they are for the cases (2.5) assumed here, are not yet general enough to solve efficiently all difference equations of the form (2.4). All kinds of "pathological" cases can easily be exhibited by diagrams such as Figs. 9a and 9b. Diagrams with much less regularity can of course be arbitrarily drawn.

In the case exhibited in Fig. 9a the coarse-grid equations, whether derived by (5.5) or by (5.6), will have no strong coupling, and thus will represent a very poor approximation and will contribute no multi-grid acceleration. In the case of Fig. 9a the situation could be corrected by using still more complicated formulae than (5.6), which take into account more possible paths between coarse-grid points. In the case of Fig. 9b even this would not help. A different approach is needed, the objective of which will be, roughly speaking, to derive coarse-grid equations in which two regions are strongly coupled if and only if they are strongly coupled on the fine grid. (Two regions are called strongly

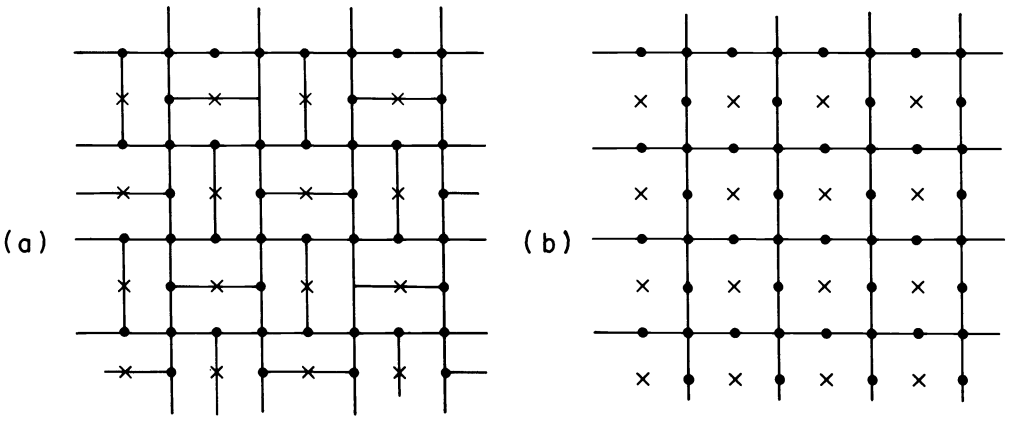


FIG. 9. *Patterns of difference equations. Part of the fine grid points is shown by dots and crosses. The crosses correspond also to coarse grid points. Links are shown between neighboring fine grid points which are strongly coupled (i.e., connected with large D).*

coupled if there is a point in one of them which is strongly coupled, through a chain of strongly-coupled intermediate points, to a point in the other.) In case 9b, for instance, all the couplings on the coarse grid need to be strong (about half as strong as on the fine grid). Methods for deriving coarse-grid equations for arbitrary fine-grid equations require further study.

Notice, however, that in discretizing diffusion problems by (2.5), we do not get arbitrary difference equations. In fact, (2.5) implies relations like

$$A_{i,j+1/2} \leqq A_{i-1,j+1/2} + A_{i+1,j+1/2},$$

$$A_{i,j+1/2} \leqq B_{i-1/2,j} + B_{i+1/2,j},$$

etc. These relations prevent pathologies of the above kind. But such relations are not necessarily maintained on coarser grids. So troubles may arise in the communication between coarser grids and still coarser ones. On those grids, however, we can afford more sophisticated procedures, since the associated computational work on coarser grids is much smaller.

9. Numerical examples. In this section we present several numerical examples indicating the behavior of various schemes described in this paper. Most examples employ a fixed algorithm, in which p indicates the number of relaxation sweeps on grid G^{k+1} after interpolation from grid G^k, $k \leqq M-2$; q indicates the number of relaxation sweeps on grid G^{k-1} after interpolation of the residuals from grid G^k, $k \geqq 3$, and r is the number of relaxation sweeps on the finest grid. All examples use an initial guess of $u \equiv 0$ and begin on the finest grid. Unless otherwise indicated, all examples use point Gauss–Seidel and solve the coarsest grid problem directly.

Example I is for the problem depicted in Fig. 10. D is $D_1 = \frac{1}{3}$ outside the shaded region and various values D_2 given in Tables 1 and 2 inside it; $f \equiv 1$ and $\sigma \equiv 0$, and the

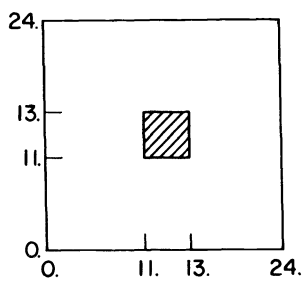


FIG. 10. *Example* I.

boundary values are given by $\partial u/\partial \nu = -1/2D\ u$ on all sides. The number of grids is $M = 4$, the finest satisfying $h_i = k_j = 1.0$. The computations for this example were done quite early in the development of our codes. At that time we were using line relaxation by lines in x, even for uniform grids, and even on the coarsest grid. Table 1 shows the result of using bilinear interpolation and the accommodative algorithm; method A uses a fixed weighting of the coefficients of L^k to obtain those of L^{k-1} while method B uses (5.5). Table 2 shows the result of using the fixed algorithm $p = q = r = 1$ and direct solution on the coarsest grid; again (5.5) is used, but the interpolation is (5.4). Table 2 shows a comparison between the multi-grid convergence factor μ_{MG} and the line SOR convergence factor μ_{SOR}. The optimum ω_{SOR} was chosen by experimentation; it is worth remarking that for large D_2/D_1 the optimum ω_{SOR} differed very little from the ω chosen by the usual power method techniques [11, pp. 203–290], yet the difference in

TABLE 1

D_2/D_1	A	B
10^{-4}	0.580	0.516
10^{-3}	0.580	0.516
10^{-2}	0.588	0.516
10^{-1}	0.586	0.516
1	0.562	0.520
10^{1}	0.710	0.644
10^{2}	0.877	0.726
10^{3}	0.931	0.768
10^{4}	0.954	0.794

TABLE 2

D_2/D_1	μ_{SOR}	ω_{SOR}	μ_{MG}
10^{-4}	0.731	1.71	0.486
10^{-3}	0.731	1.71	0.486
10^{-2}	0.731	1.71	0.486
10^{-1}	0.732	1.71	0.487
1	0.744	1.71	0.490
10^{1}	0.764	1.74	0.498
10^{2}	0.875	1.87	0.532
10^{3}	0.960	1.95	0.537
10^{4}	0.990	1.99	0.537

convergence rate was much better with the experimentally chosen ω_{SOR}; that is, in practice μ_{SOR} would be even worse than the values given in Table 5.

This first example includes ratios of D_2 to D_1 that do not occur in practical runs of TTDAMG; there D_2/D_1 ranges from 10^{-2} to 10^{2}. However, the range of D_2/D_1 can be, e.g., 10^{-4} to 10^{4} in problems arising directly from diffusion theory, and the remaining examples are chosen to explore various difficulties that can arise in such problems. (We remark that in many practical problems, the pattern of D's, f's, and σ's is a very complicated mosaic. We have tried such problems with success, but because of the difficulties in describing the mosaic, we do not present any here.) We also give results for using four grids, the finest grid again satisfying $h_i = k_j = 1$.

Example II is a problem with a four-corner junction. The region is $\Omega = (0., 24.) \times (0., 24.)$. Γ is $x = 12.$ and $y = 12.$ The values of $D[f]$ in the northwest, northeast, southwest, and southeast quadrants with respect to (12., 12.) are 1000. [1.], 1.[0.], 1.[0.], and 1000.[1.], respectively. The boundary conditions are

$$\frac{\partial u}{\partial \nu} = \begin{cases} 0 & \text{on } x = 0. \text{ and } y = 0., \\ -\dfrac{1}{2D}\, u & \text{on } x = 24. \text{ and } y = 24., \end{cases} \tag{9.1}$$

and σ is identically zero.

The number of grids for this problem is $M = 4$, the finest satisfying $h_i = k_j = 1.0$; thus, the coarse-grid problem has 16 unknowns, and the finest grid problem has 625 unknowns. Table 3 indicates the results of several methods for this problem. Method C uses (5.6) to define L^k, $k < M$ and $(I_{k-1}^k)^*$ as a residual weighting. Method D is the same

TABLE 3

p,q,r	C	D	E	F
1,1,1	*	0.56, 0.83	0.54, 0.75	0.59, 0.78
1,2,2	*	0.60, 0.75	0.59, 0.72	0.65, 0.77
1,2,3	*	0.66, 0.77	0.65, 0.74	0.69, 0.77
1,1,2	*	0.59, 0.71	0.53, 0.69	0.62, 0.75

as C but uses (5.8) to correct the bad discretization on the finest grid. Method E uses $L^{k-1} = (\tilde{J}^k_{k-1})^* L^k \tilde{J}^k_{k-1}$ and $(\tilde{J}^k_{k-1})^*$ as a residual weighting, and method F is method E with (5.8) used to correct the bad discretization on the finest grid. Each of these methods was run with various fixed algorithms; these are indicated by giving p, q, and r, where p is the number of relaxation sweeps from G^k to G^{k+1}, $k \leqq M-2$, q is the number of relaxation sweeps after interpolation from G^k to G^{k-1}, $k \geqq 2$, and r is the number of relaxation sweeps on the finest grid; the coarse-grid problem is solved directly.

For each run two numbers are given. The first indicates the convergence factor in terms of the work of relaxation sweeps alone, and the second indicates the convergence factor in terms of relaxation sweeps and interpolations, assuming the latter have been done efficiently (that is, assuming J^k_{k-1} is stored and that residuals are computed during the last relaxation sweep before interpolation from G^k to G^{k-1}). Note that methods E and F require twice the work of C and D for relaxation on the coarser grids since they use nine point operators instead of five. An asterisk in the table indicates divergence.

Note that method C diverges due to the bad discretization on the finest grid and the fact that (5.6) converts this bad discretization into bad communication between grids. (See the discussion in § 8.) The accommodative algorithm would converge for method C but with a bad convergence factor. For methods D, E, and F, the algorithm $p = 1, q = 1$, $r = 2$ is the best. It is curious that method F is slightly worse than method E.

Example III is for the problem depicted in Fig. 11. $D[f]$ is 1.[0.] outside the shaded regions and 1000.[1.] inside it; $\sigma \equiv 0$, and again the boundary values are as in (9.1). This problem uses three grids, the finest satisfying $h_i = k_j = 1$. Thus the shaded region falls between the grid lines of G^2, and with method C the set of the strongly coupled points

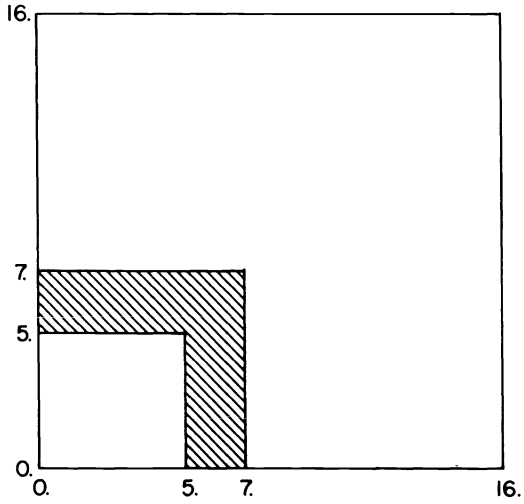


FIG. 11. *Example* III.

on G^2 is an inverted L. The results are summarized in Table 4. Method C diverges; however, method C with the accommodative algorithm should converge, though slowly. Method G uses line relaxation by lines in x and also diverges. Method H uses alternating line relaxation; the first [second] pair of numbers for H counts each sweep of alternating line relaxation as one [two] work unit[s]. Method I uses SCOD relaxation, relaxing the inverted L of points on G^2 as a block. In method E one-dimensional tails appear on G^2 also, but—unlike method C—there is a set of strong coupled points on grid G^1; in fact this set is two-dimensional and apparently acts as a good correction to grid G^1. We also give results for using four grids, the finest satisfying $h_i = k_j = 1$.

TABLE 4

p, q, r	C	E	G	H	I
1,1,2	*	0.55, 0.71	*	0.55, 0.68 0.7, 0.8	0.56, 0.69
1,1,2 (4 grids)	-	0.61, 0.75	-	-	-

Example IV is for the problem depicted in Fig. 12. $D[f]$ is 1.[0.] outside the shaded region and 1000.[1.] inside it, $\phi \equiv 0$, and again the boundary values are as in (9.1). Again the problem uses three grids, the finest satisfying $h_i = k_j = 1$. Thus again the shaded region falls between the grid lines of G^2, and with method C the set of strongly coupled points of G^2 is a one-dimensional staircase. The results are summarized in Table 5. All the methods based on (5.6) diverge, even the accommodative algorithm. (With the accommodative algorithm and 3 grids, the residual norm on G^2 is worse after interpolation from the solution from G^1 than before. Hence, the method cycles between G^1 and G^2 with the residual norm on grid G^2 getting bigger and bigger.) Method C works fine if the coarsest grid is eliminated, that is, if only two grids are used. Hence, the problem is clearly that when 3 grids are used, G^1 provides a terrible correction to G^2. With method E, the set of strongly coupled points of G^2 is

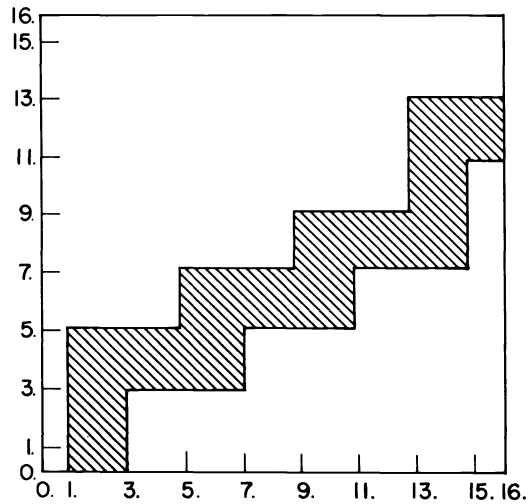


FIG. 12. *Example* IV.

TABLE 5

p, q, r	C	E	G	H	I
1,1,2	*	0.59, 0.74	*	*	*
accommodative	*	–	*	*	*
1,1,2 (4 grids)	–	0.63, 0.76	–	–	–

two-dimensional, i.e., the steps of the staircase are connected by strong cross-coupling coefficients. Again, we give results for four grids for this problem, the finest grid remaining unchanged.

These examples indicate clearly that (5.11) is more robust than (5.6) or (5.9). For problems arising in TTDAMG, (5.6) or (5.9) is suitable; however, even for these problems if one can afford to store $\tilde{J}_{k-1}^{k}$ we feel that (5.11) is to be preferred since it is more efficient than (5.6) or (5.9) for all but the simplest problems (like Laplace's equation), and even for these problems (5.11) is not much less efficient than (5.6) or (5.9).

Finally, we remark that examples II–IV have been repeated in [8] using method E and giving timings for a CDC 7600; in [8], however, one full multi-grid cycle with $\tilde{J}_{k-1}^{k}$ is used to start the calculation. Also, examples II–IV are repeated with nonzero σ and with piecewise bilinear finite elements being used as the discretization on the finest grid.

Acknowledgment. We thank Blair Swartz for advice that improved this paper.

REFERENCES

[1] R. E. ALCOUFFE, *Diffusion synthetic acceleration methods for the diamond-differenced discrete-ordinates equation*, Nucl. Sci. Engrg. 64 (1977), pp. 344–355.

[2] A. BRANDT, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp. 31 (1977), pp. 333–390.

[3] ———, *Multi-level adaptive finite element methods, I. Variational problems*, in Numerical Methods for Partial Differential Equations, S. Parter, ed. Academic Press, New York, 1979, pp. 53–147.

[4] A. BRANDT, J. E. DENDY, JR., AND H. RUPPEL, *The multi-grid method for semi-implicit hydrodynamic codes*, J. Comp. Phys. 34 (1980), pp. 348–370.

[5] A. BRANDT AND N. DINAR, *Multi-grid solutions to elliptic flow problems*, ICASE rep. 79-15, 1979.

[6] IVO BABUSKA, *Homogenization and its application*, in Mathematical and Computational Problems, Numerical Solution of Partial Differential Equations–III, B. Hubbard, ed., Academic Press, New York, 1976, pp. 89–116.

[7] A. BRANDT, *Multi-level adaptive techniques*, IBM Res. Rep. RC6026, 1976.

[8] J. E. DENDY, JR. AND J. M. HYMAN, *Multi-grid and ICCG for problems with interfaces*, in Santa Fe Elliptic Equation Conf. Proc., M. Schultz, ed., 180, to appear.

[9] R. A. NICOLAIDES, *On the l^2 convergence of an algorithm for solving finite element equations*, Math. Comp. 31 (1977), pp. 892–906.

[10] F. DE LA VALLÉE POUSSIN, *An accelerated relaxation algorithm for iterative solution of elliptic equations*, SIAM J. Numer. Anal., 5 (1968), 340–351.

[11] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ., 1962.

[12] E. L. WACHSPRESS, *Iterative Solution of Elliptic Systems and Applications to the Neutron Diffusion Equations of Reactor Physics*, Prentice-Hall, Englewood Cliffs, NJ., 1966.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0007 $01.00/0

A TWO-DIMENSIONAL MESH VERIFICATION ALGORITHM*

R. B. SIMPSON†

Abstract. A finite element mesh is usually represented in a program by lists of data, i.e., vertex coordinates, element incidences, boundary data. This paper is concerned with conditions on the list data which ensure that the lists describe a "tiling" of some planar region without overlap or gaps. For a particular format of lists, a set of such conditions is given which is proven to be sufficient to guarantee such a "tiling". These conditions have been chosen so as to be verifiable by the algorithm referred to in the title, which is described in detail and is claimed to be of reasonable efficiency.

Key words. mesh, finite element, triangulation

1. Introduction. A mesh on a region of the plane generally appears to the reader of a textbook or research paper as a diagram showing a partition of the region into finite elements of simple geometric shapes, usually triangles or quadrilaterals. Intuitively, the partition can be thought of as a tiling of the region up to its boundaries by the finite elements as tiles. On the other hand, it appears to the user of the method in the source code of his programs as lists of numbers of specific types, e.g., positive integers less than M, real numbers, etc. In general, however, if the lists are filled with arbitrary data of the correct type, they only represent some collection of elements, which may overlap each other, or leave gaps in the region's interior. Whether the collection represents a proper tiling or not is data dependent. The purpose of this paper is to make an explicit statement of this dependence in the form of a set of four conditions that the data must satisfy, these conditions being specific to two-dimensional meshes. Although the conditions are geometrically simple, their verification for the lists of a particular mesh involve nontrivial computations which have been organized in this paper into an algorithm which is referred to as the mesh verification algorithm. One consideration in the choice of these conditions, then, has been that the mesh verification algorithm be sufficiently efficient to be practically viable.

Implemented in a program, the algorithm can be used to check a mesh produced for a particular computation. It is a common practice in employing the finite element method to prepare a mesh using a mesh generation program, possibly store it in a file, and plot it to examine its correctness and suitability to the region and the problem before proceeding to the subsequent stages of the method. Often this procedure is repeated a number of times because the mesh is shown to be incorrect, typically due to errors in input data to the mesh generation program, or weaknesses in its algorithm or "bugs" in its programming. An algorithmic verification of the output lists based on the criteria in this paper is viewed as being a check on the mesh which is complementary to a graphical examination in the sense of being faster and not dependent on graphic facilities, but not providing the positive evidence of the suitability of the mesh that a visual inspection gives.

A second motivation for these conditions is to give a mathematically rigorous definition of a finite element mesh that can serve in the study and development of mesh

* Received by the editors August 27, 1980, and in revised form April 27, 1981. This research was carried out at the Brunel Institute of Computational Mathematics, Uxbridge, U.K., while the author was on sabbatical leave from the University of Waterloo, and was supported by a grant from the Natural Science and Engineering Research Council of Canada.

† Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1.

generation programs, allowing explicit specifications for algorithms and providing a debugging tool. A number of methods for mesh generation have been discussed in the published literature, e.g., [3], [6], [11], [15], which have been implemented in programs giving extensive satisfactory use. There is no doubt that mesh generation programs can be written which are pragmatically successful without a mathematical definition of a finite element mesh. However, to say whether such a program is correct or not, or to state under what conditions it fails, or to compare two such programs is difficult because of the vagueness about the specifications for output as well as input even at the algorithmic level. The programs produce meshes in their list form, but as we have mentioned above, not all lists correspond to legitimate meshes. In particular, it is believed that the four conditions of the definition can be used as verification conditions for proving mesh handling programs correct (see Van Emden [18], for a pragmatic discussion of the use of verification conditions in programming). While the question of whether, or in what measure, program correctness procedures will aid program synthesis is an area of speculation, there seems little doubt that a better understanding of the mathematics of an algorithm and its data generally leads to programs which are better in a variety of senses.

In § 2, the list representation that we will use for a collection of elements is introduced, and with it, four conditions on the collection are stated geometrically which form the proposed definition of a planar mesh. These condition are then justified in § 3, on a mathematical level by proving that a set of lists which meet the conditions describe a tiling of some region of the plane. The definition is restricted to apply to meshes for regular planar regions (possibly multi-connected, or disconnected); in particular, it does not extend to meshes for describing regions with cracks as used in some finite element applications. Readers who are not interested in the formal justification of these conditions may proceed to the development of the mesh verification algorithm in § 4 and § 5. In these sections, the conditions stated for the collection of elements represented by mesh lists are expressed as computational checks on the list data.

In these computations, information about the element sharing a common edge with a given element (i.e., an element's neighbor) is required. This information is used in a variety of other contexts in the finite element method, e.g., averaging stresses over neighboring elements [17, p. 168], improving the triangulation of a region [11], or performing local mesh refinements [13], [15]. The process of obtaining and verifying this information involves a list inversion of one of the mesh lists, and is of some independent interest, so it is discussed separately in § 4. The other aspects of checking the conditions are dealt with in § 5. At this stage, the generality of the elements' geometry becomes a significant factor, so the discussion is specialized to triangular meshes to avoid unwarranted complexity.

In composing the algorithms of § 4 and § 5, a compromise between simplicity and optimality has been sought. Some comments on the performance of a Fortran implementation are given in § 6. An inspection of the components of the verification algorithm indicates that it should run in times linear in the number of triangles in the collection being verified, subject to some restrictions on the mesh topology that are quite natural for finite element meshes.

While the author is unaware of other algorithms for verifying a mesh in this sense, several algorithms for verifying other geometric "objects" have appeared. There is a substantial literature on algorithms for verifying graph planarity (e.g., Hopcroft and Tarjan [9]). Recently, a linear time algorithm for verifying the planarity of a 2-complex has been published by Gross and Rosen [8], and in [14], Shamos and Hoey give an algorithm for verifying when a planar polygon is simple. In these references, the

algorithms tend to be described at a high level, with the primary emphasis placed on the analysis of the complexity of the process.

2. List definitions and the conditions for a mesh. The list representation for a finite element mesh which we will assume here consists of two basic lists:

the vertex coordinate list of length N_v,

the element incidence list of length N_e

and an auxiliary list:

the boundary reference table of length N_b.

The basic lists contain independent data, but the boundary reference table's information about the mesh can be obtained from the other two. The kth entry of the vertex coordinate list is the coordinates (x_k, y_k) of the kth vertex of the mesh, also denoted $P(k)$ in the sequel. The term "vertex" here refers to the points which determine the geometric shape of the element as a region (e.g., the 3 vertices of a triangle), as opposed to the term "node" which is commonly used for points associated with degrees of freedom of the element shape function. The entries in the vertex coordinate list are required to be unique, i.e., if $j \neq k$, then $P(j) \neq P(k)$. The jth entry in the element incidence list is itself a sublist of the indices in the vertex coordinate list of the vertices of the jth element. $E(j)$ will be used to denote the jth element and if it has $I(j) \geqq 3$ vertices, then the jth entry of the element incidence list consists of integers $v(1, j), v(2, j), \cdots, v(I(j), j)$ with $0 < v(i, j) \leqq N_v$, and $v(i, j) \neq v(k, j)$ if $i \neq k$. The ith vertex of $E(j)$ is $P(v(i, j))$ and i will be referred to as the local vertex number of $P(v(i, j))$. The ith edge of $E(j)$ is the directed line segment running from $P(v(i, j))$ to $P(v(\bar{i}+1, j))$, where

$$\bar{i} \equiv i \bmod I(j) \tag{2.1}$$

is a notation used in the sequel for indexing the "next" vertex around $E(j)$, and i will be referred to as the local side number of this side.

The kth entry of the boundary reference table consists of a pair of integers $(b(1, k), b(2, k))$ which described the kth boundary edge of the mesh by giving the index of the element to which it belongs, $0 < b(1, k) \leqq N_e$, and the local side number, $0 < b(2, k) \leqq I(b(1, k))$. This list is ordered first by $b(1, k)$ and within entries having the same value for $b(1, k)$ by $b(2, k)$ i.e.,

$$\begin{aligned} &k_1 < k_2 \Rightarrow b(1, k_1) \leqq b(1, k_2) \\ &\text{and if } b(1, k_1) = b(1, k_2) \text{ then } b(2, k_1) < b(2, k_2) \end{aligned} \tag{2.2}$$

This list is the least standard of the three in the literature on finite element programming. It was proposed by J. A. George in his thesis [7], and doubtless has been used independently by other implementors of the finite element method. As mentioned above, the mesh information contained in it is redundant, as it is contained implicitly in the element incidence list information. The algorithms that we discuss below could either build the boundary reference table from the element incidence list, or check that it is consistent. In view of the use of meshes in the finite element method, we have chosen to describe the latter. Typically, the table contains additional problem specific data concerning boundary conditions and is generated with this data at the time the mesh is generated (i.e., the other two lists are constructed.) The ordering of the table is done to facilitate synchronizing a scan of the element incidence list with a scan through the table. An example of this occurs in (4.7) of § 4; however, the primary instance of this, for the finite element method, occurs in the scan of the elements to generate local stiffness matrices. In some implementations, these boundary data are added as extra fields in the entries of the element incidence sublists either directly or through pointers.

A discussion of some alternative representations for meshes, and issues associated with them, may be found in [16]. While the definition given here clearly does not conform directly to a variety of implemented data structures for mesh representation, it is expected that a fairly simple identification or translation can be made between the representation used here, and the data structures of most implementations.

We now specify the four conditions on the collection of elements described by the mesh lists that qualify this collection to be a mesh.

C1. The directed polygonal curve formed by traversing the element sides in local side number order forms a simple closed curve with bounded interior (i.e., bounded region on the left of the curve). The finite element $E(j)$ is defined to be the closure of the interior of this curve.

C2. The ith edge of $E(j)$ is either the only edge joining its endpoints or there is one other element, $E(l)$, having an edge joining these vertices. In the latter case, the line segment joining these vertices must have the opposite direction as an edge of $E(j)$ to its direction as an edge of $E(l)$. In the first case of C2, the ith edge of $E(j)$ is a boundary edge and $E(j)$ is its boundary element. To be consistent, the boundary reference table must have an entry with $b(1, k) = j$, $b(2, k) = i$. In the second case of C2, the ith edge of $E(j)$ is an interior edge with $E(j)$ and $E(l)$ as neighbors across this edge. The requirement concerning the directions of the line segment as an edge of $E(l)$ and $E(j)$ will be referred to as edge consistency and if it holds then the vertex of index $v(\bar{i}+1, j)$ will be the mth vertex of $E(l)$ for some value of m which is referred to as the complementary local edge number in § 4.

C3. No interval of a boundary edge intersects an element other than its boundary element.

C4. A vertex can have at most one boundary edge directed away from it. Vertices with one such edge will be referred to as boundary vertices in this paper.

In § 3, we want to establish that these requirements are sufficient to ensure that the lists describe sets of elements which "tile" a region of the plane. However, first we will discuss some examples in which C2–C4 are violated to illustrate the sense in which they are necessary. The requirement in C1 that edges have the direction that puts the element on the left coupled with the edge consistency restriction of C2 ensures that neighbors lie on opposite sides of their common edge, i.e., that a short line segment which intersects an interior edge passes from the interior of one neighbor into that of the other. Without these requirements, for example, the configuration of Fig. 2.1 would qualify as a mesh with the list of vertices $P(1) = (1, 1)$; $P(2) = (1, -1)$; $P(3) = (-1, -1)$; $P(4) = (-1, 1)$; element incidences $v(i, j)$ given by Table 2.1 and an empty boundary reference table.

The necessity of C3 can be seen from the obvious overlap in Fig. 2.2, referred to as mesh "overspill" in [6].

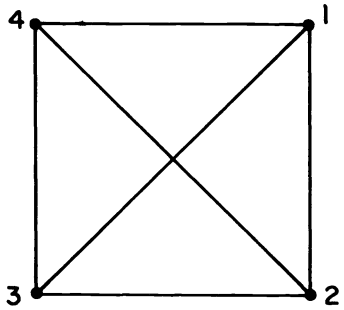


FIG. 2.1

TABLE 2.1
v(i, j) for Fig. 2.1

j \ i	1	2	3
1	3	2	1
2	4	2	1
3	3	1	4
4	3	2	4

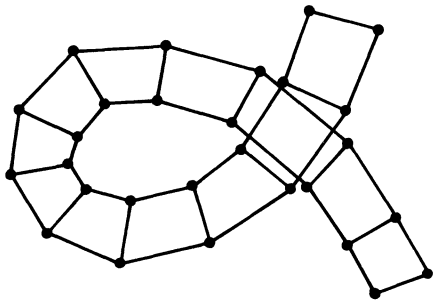


FIG. 2.2

In Fig. 2.3, an example in which the region covered by the elements of the mesh has both overlap and a "gap" is shown. The elements are the eight outer squares plus the four triangles, and the boundary reference table records the outer edges of the squares and the oblique sides of the triangles as boundary edges. The triangles share their horizontal or vertical edges with a square, but are not strictly neighbors because the edge consistency along their common edge cannot hold, or counter clockwise listing of vertices fails, i.e., C1 and C2 cannot both hold.

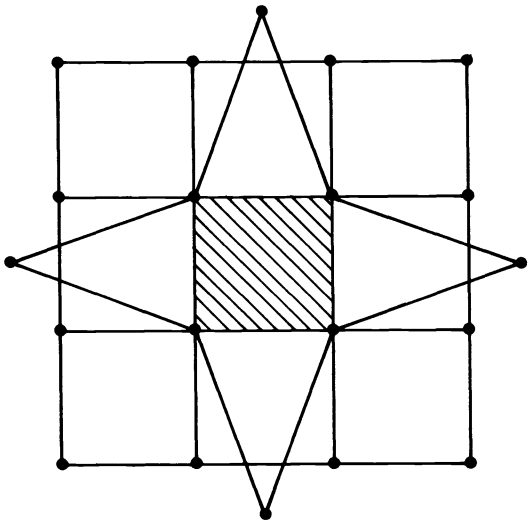


FIG. 2.3

The central square of the figure is not an element of the mesh, but is a region of the plane not separated from the elements of the mesh by a boundary edge. It is a form of "gap" in the mesh. Condition C3 and one of C1 or C2 are violated in this case.

The role of C4 seems less clear; certainly it precludes undesirable anomalies such as a mesh with every edge a boundary edge, like the black squares of a checkerboard. However, it appears that its primary implication is that the boundary curves form simple closed curves as in Theorem 1, and no example has been constructed in which C2 and C3 are satisfied, C4 is violated, and an overlap of element interiors occurs.

3. Proof of sufficiency. We turn now to establishing that the requirements C1–C4 are sufficient to ensure that the lists of (2.1) correspond in some rigorous way to the intuitive idea of the elements tiling a region of the plane. First it is shown that the set of boundary edges of the mesh form a set of disjoint simple closed curves in Theorem 1. We then show that these curves are oriented in a consistent manner, so that their interiors define bounded regions of the plane for which they are the boundaries in Lemma 3. It is then shown in Theorem 2 that this region is covered without gaps by the elements of the mesh and in Theorem 3 that it is covered without overlap. To avoid repetition of "polygonal" in this section, we shall assume henceforth that all arcs or curves are polygonal. For a helpful, if elementary, reference on curves and regions in the plane, see [1].

The following lemma concerning a type of connectivity of the mesh will be quite useful in the sequel.

LEMMA 1. *Let P be a point lying in the interior of m elements of the mesh for $m \geqq 0$ and let P be joined to a point Q by an arc which passes through no mesh vertex and intersects no boundary edge. If Q does not lie on an element edge, then Q lies in the interior of m elements of the mesh.*

Proof. As a point moves along the arc from P towards Q, it can only leave an element through an interior edge, at which point it enters its neighbor (C2). □

LEMMA 2. *A vertex is a boundary vertex if and only if it has a boundary edge directed towards it. The incoming boundary edge of a boundary vertex is unique.*

Proof. Let P be a vertex and let $\mathscr{E}$ be the set of edges having P as an endpoint. Assign $e \in \mathscr{E}$ a value 1 if it is directed towards P; -1 if it is directed away from P. Since each element for which P is a vertex contributes two edges to $\mathscr{E}$, one incoming and one outgoing, it is clear that the sum of the values of edges in $\mathscr{E}$ is zero. By requirement C2, the contribution to this sum from all interior edges in $\mathscr{E}$ is zero, and by C4 the contribution from all outgoing boundary edges is 0 or 1. Hence the contribution from all incoming boundary edges at P must be 0 or -1, from which the lemma follows. □

THEOREM 1. *The boundary edges of the mesh form a set of simple closed oriented curves, $C_1, C_2, \cdots, C_M$, which do not intersect each other.*

Proof. Let P be a boundary vertex and consider the curve traced out by following the unique outgoing boundary edge from each vertex to the boundary vertex to which it is directed (Lemma 2) starting from P. Since this process can be continued indefinitely, and there are only a finite number of boundary vertices, one must be traversed twice by this curve. If the first one to occur twice is not P, then there are two distinct boundary edges leading to it, in violation of Lemma 2. Hence the curve formed by carrying out this process until P is encountered a second time is a closed oriented curve which can be labelled C_1. To see that this curve is simple, we note that from C3 it cannot intersect itself on the line segments between vertices, and as no vertex other than P is visited twice in a circuit starting at P, it cannot intersect itself at a vertex. If not all the boundary vertices lie on C_1, then the argument can be applied to another boundary vertex to

establish a simple closed oriented curve C_2, and so on. Two of these curves cannot intersect at the interior of a line segment by C3 nor at a vertex by Lemma 2, hence they must be disjoint. □

The region on the left of a simple closed oriented curve, C, is conventionally referred to as its interior while the region on the right is referred to as its exterior. Such a curve divides the plane into a bounded region and an unbounded one, and whether the bounded region is the curve's interior or not depends on the curve's orientation, of course. Although we can expect from C1 and C4 that the elements of the mesh should lie in the interiors of these curves in some sense, Theorem 1 does not give any information about the relative orientations of these boundary curves. For example, it is conceivable that both C_1 and C_2 have bounded interiors and that C_2 lies in the interior of C_1. If no curve lies between C_1 and C_2, then there is no region for which C_1 and C_2 are oriented boundary curves, as shown in Fig. 3.1A, with the interiors shaded. However, if C_3 lies between C_1 and C_2 as in Fig. 3.1B, and is oriented so as to have an unbounded interior, then the shaded region is a (disconnected) region with C_1, C_2 and C_3 as its oriented boundaries.

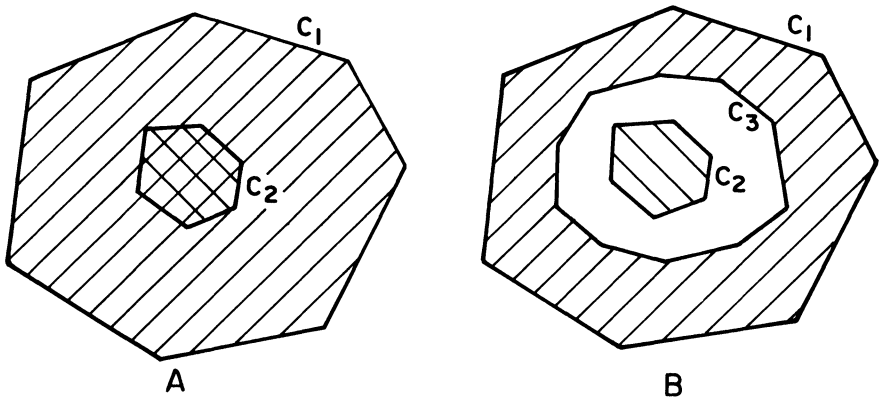


FIG. 3.1

Our aim now is to show that the boundary curves established in Theorem 1 are oriented in a consistent way, so that each curve of bounded interior defines a connected region of possibly multiple connectivity, the boundaries of the "holes" in this region coinciding with boundary curves of unbounded interior. A curve C_i will be said to be maximal in C_j if C_i lies in the bounded domain determined by C_j, but not in the bounded domain of any other curve lying in the bounded domain of C_j, (e.g., C_3 is maximal in C_1 in Figure 3.1B, while C_2 is not).[1]

LEMMA 3. (i) *Let C_j be a boundary curve of bounded interior and let C_i be any curve maximal in C_j. Then C_i is oriented so that its interior is unbounded.*

(ii) *Let C_i be a boundary curve with unbounded interior. Then there is a curve of bounded interior, C_j, such that C_i is maximal in C_j.*

Proof. (i) Let D be the domain obtained by removing from the bounded interior of C_j the bounded regions determined by all curves maximal in C_j. If there are no such

[1] This terminology is motivated by the fact that inclusion generates a partial ordering on the unoriented simple closed curves and C_i is maximal in the set of $C_k < C_j$ in the sense of this ordering.

maximal curves, then there is nothing to prove, so let us assume there is at least one, and designate it C_i. Let E be a boundary element with a boundary edge on C_i and let P be a point of the interior of E arbitrarily near C_i. Let Q be in D arbitrarily close to an edge of C_i and not lying on an element edge. Then we can join P to Q by a polygonal arc in D, which does not cross any boundary edges and can be adjusted to avoid any vertices (since D is an open, connected set). By C2, P is in exactly one element and hence by Lemma 1, Q is in one element, E'. However, Q may be arbitrarily close to C_i so E' has a boundary edge on C_i and C_i must be oriented so that D lies in its interior.

(ii) Let $\bar{C}$ be a circle sufficiently large to contain all the elements. If the statement were not true, then C_i would be maximal either in $\bar{C}$ or in C_j where C_j also has unbounded interior. We will use $\bar{C}$ to refer either to the circle or to C_j, and note that in either case, points of the bounded region of $\bar{C}$ near $\bar{C}$ do not lie in any element of the mesh. Then, as in part (i) let D be the domain formed by removing from the bounded region of $\bar{C}$ the bounded regions of all curves maximal in $\bar{C}$. Then we can pick P in D close to $\bar{C}$ so that P lies in no element and, since D is connected and open, join it to a point Q in a boundary element near C_i by an arc in violation of Lemma 1. This contradiction establishes Lemma 3 (ii). □

This lemma shows then that every mesh boundary curve of bounded interior defines a possibly multiply-connected region with mesh boundary curves as the region's boundary , and that every mesh boundary curve is a part of the boundary of such a region. While finite element problems typically require meshes for one connected region, there does not seem to be a compelling reason to restrict the definition of a mesh so that its boundary curves determine only one. However, for ease of exposition, we shall now assume that there is only one boundary curve of bounded interior and label it C_1. Then, from Lemma 3, it follows that if $K > 1$ there are $K-1$ curves $C_2, \cdots, C_K$ inside C_1 oriented to have unbounded interiors, so that the region

$$R = \bigcap_{k=1}^{K} (\text{interior of } C_k) \tag{3.1}$$

is a bounded region of connectivity $K-1$, which will be defined as the covered region of the mesh. This terminology will be justified by showing that the elements of the mesh cover this region without gaps (in Theorem 2) and without overlap (in Theorem 3).

THEOREM 2.

$$\bigcup_{i=1}^{N_e} E(i) = R.$$

Proof. To show that $\bigcup_{i=1}^{N_e} E(i) \subset R$, suppose that for some i and j, $E(i) \cap$ (the exterior of C_j) is not void. Then, since the exterior of C_j is connected, we can join $P \in E(i)$ to $Q \in C_j$ by an arc in exterior of C_j which passes through no mesh vertices. Since the exteriors of the C_k are disjoint, this arc will not pass through any boundary edges between P and Q. But near Q it lies outside every element, which contradicts Lemma 1. If there were a point $Q \in R - \bigcup_{i=1}^{N_e} E(i)$ then an arc from C_1 to Q_N could be similarly constructed which again violates Lemma 2, so $R = \bigcup_{i=1}^{N_e} E(i)$. □

To establish no overlap, we have:

THEOREM 3. *Let P be a point of R which is neither a mesh vertex nor lies on an element edge. Then P lies in the interior of one element.*

Proof. The proof proceeds inductively by constructing a finite sequence of subsets of elements $\mathcal{G}_1, \mathcal{G}_2, \cdots, \mathcal{G}_F$, such that

(3.2a) $\mathcal{G}_i$ satisfies C1, C2, C3 and C4 (see below for a minor modification of C4 for this proof), and

(3.2b) if P lies an element E of the mesh then $E \in \mathcal{G}_i$.

The sequence is constructed by starting with $\mathcal{G}_1$ = the entire mesh, and by consecutively removing elements with edges on the boundary of $\mathcal{G}_i$ but which do not contain P. It terminates when there are no such elements left, with $\mathcal{G}_F$, a subset of the elements such that each piecewise linear arc of the boundary of $\mathcal{G}_F$ is an edge of an element containing P. It is then shown that there can be only one element in $\mathcal{G}_F$, which completes the proof. □

Let $\gamma(\mathcal{G}_i)$ denote the boundary curves of $\mathcal{G}_i$ and suppose that we have constructed $\mathcal{G}_1, \cdots, \mathcal{G}_i$ and let $E \in \mathcal{G}_i$ with $\partial E \cap \gamma(\mathcal{G}_i) \equiv \partial E(b)$ not empty and $P \notin E$. Let $\partial E(\text{int}) = \partial E - \partial E(b)$.

We wish to remove E from $\mathcal{G}_i$ to get $\mathcal{G}_{i+1}$. Clearly $\mathcal{G}_{i+1}$ thus formed is a collection of elements satisfying C1 and C2. In order to see that it satisfies C3, we need to show that an edge of ∂E(int) does not intersect any other edge of the mesh. Suppose, to the contrary, that there were an element, H, for which one or more edges of H intersect ∂E(int). These intersecting edges of H must be interior edges of $\gamma(\mathcal{G}_i)$ since $\gamma(\mathcal{G}_i)$ satisfies C3. We can construct a curve which lies close to ∂E, but inside E, running in the direction of ∂E, which originates at a point Q, so close to $\partial E(b)$ that it lies only in E (by C2). This curve would have to pass through $H \cap E$ at some points, which would violate Lemma 1. Hence we conclude that ∂E(int) does not intersect any edge of the mesh. However the segments of ∂E(int), become boundary segments of $\mathcal{G}_{i+1}$, hence $\mathcal{G}_{i+1}$ satisfies C3.

We now turn to showing that C4 is satisfied by $\mathcal{G}_{i+1}$, i.e., that each boundary vertex of $\gamma(\mathcal{G}_{i+1})$ has a unique outgoing boundary segment. Actually, it is necessary to modify this condition slightly by replacing some boundary vertices by small circular arcs as indicated below. When E is removed from $\mathcal{G}_{i+1}$, the directed line segments of ∂E(int) will become boundary segments of $\gamma(\mathcal{G}_{i+1})$ with their directions reversed. ∂E(int) might be made up of several contiguous sections, but the argument supporting C4 is the same for each, so we consider one section of it, ordering its vertices in the counter clockwise direction around E. The outgoing boundary segment for the first vertex of a section of ∂E(int) is unchanged by removing E from $\mathcal{G}_i$. For the last vertex of a section, the outgoing boundary segment will switch from being a segment on $\partial E(b)$ to being its predecessor on ∂E.

For a vertex interior to a contiguous section of ∂E(int), there are two possibilities. If the vertex does not lie on $\gamma(\mathcal{G}_i)$, then the outgoing boundary segment on $\gamma(\mathcal{G}_{i+1})$ is its predecessor on ∂E. However, if it does lie on $\gamma(\mathcal{G}_i)$, as in Fig. 3.2A, then when E is removed, this vertex will have two incoming segments and two outgoing ones. To resolve the resulting ambiguity, we introduce a small circular arc which cuts off the ends of elements incident on Q, as shown in Fig. 3.2B.

If we take the circular arc sufficiently small, then the incidence relation of an element being incident on the arc will be equivalent to the relation that an element is incident on the vertex, Q, and P, the point referred to by the theorem statement, lies outside the arc.

Removing E from $\mathcal{G}_i$ breaks the arc about Q into two subarcs as shown in Fig. 3.2C, and each subarc has a unique outgoing boundary segment. Hence C4 is satisfied by $\mathcal{G}_{i+1}$ in the sense that each vertex, or small circular arc, has at most one boundary edge directed away from it. This is sufficient to allow the conclusions of the previous theorems and lemmas to apply to $\gamma(\mathcal{G}_{i+1})$. We conclude, then, that $\gamma(\mathcal{G}_{i+1})$ is a set of boundary curves for a possibly disconnected domain in the plane. If it is disconnected, then P will lie in one of the components and we can discard from $\mathcal{G}_{i+1}$ all elements not intersecting this component and the reduced $\mathcal{G}_{i+1}$ will still satisfy (3.2a, b).

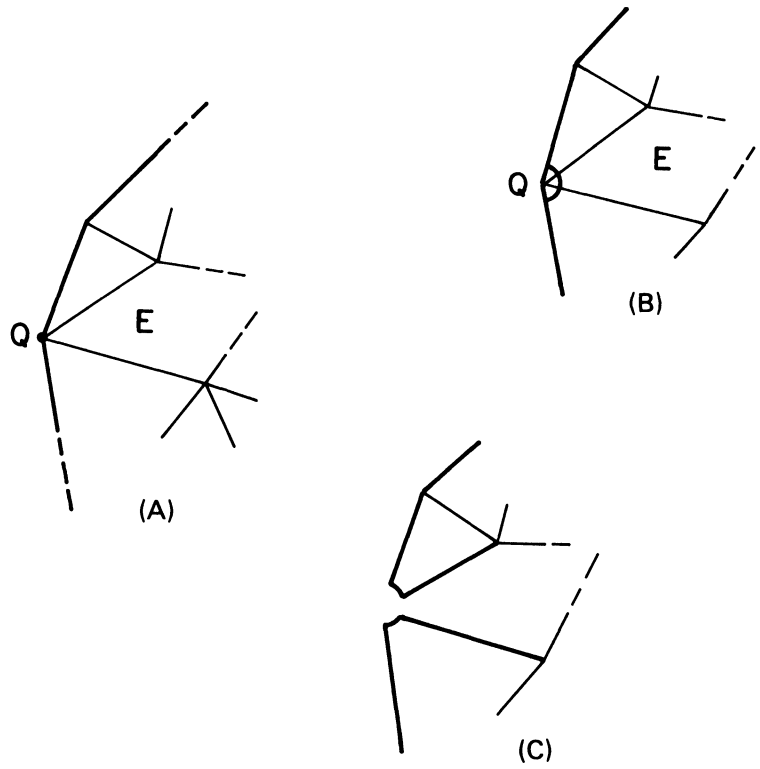


FIG. 3.2

Eventually this induction process leads to a subset of elements, $\mathcal{G}_F$, for which every element with an edge in $\gamma(\mathcal{G}_F)$ contains P. If $\mathcal{G}_F$ contains more than one element, then $\gamma(\mathcal{G}_F)$ has edges from more than one element, so suppose this latter to be the case. A point, Q, on an edge in $\gamma(\mathcal{G}_F)$ belonging to element $E \in \mathcal{G}_F$ can be joined to P by an arc lying in E. At Q, this arc lies in exactly one element, whereas at P it lies in more than one, violating Lemma 1. Hence $\mathcal{G}_F$ consists of one element and the theorem is proven. □

4. Construction of an element neighbors list and verification of C2. To verify C2, we build two lists which are useful in a variety of contexts in the finite element method, a list of elements incident on a given vertex, and a list of neighbors of each element. A convenient method for building these lists is described by Lewis and Robinson in [11] and the basic ideas have doubtless been used in many implementations of the finite element method. By adding some refinements to the basic ideas, we obtain an algorithm which serves both the purposes of building these lists and checking C2 of the mesh definition.

The method first constructs a vertex incidence list, i.e., a list whose kth entry is the sublist $e(i, k)$, $i = 1, 2, \cdots, K(k)$, where $e(i, k)$ is the index of an element incident on vertex k. $K(k)$ then is the number of elements incident at vertex k. The algorithm builds the sublists $e(i, k)$, for each k in increasing order, i.e., $e(i, k) < e(i+1, k)$, and it is convenient for searching of these sublists to add a guard entry at the end of each sublist,

$e(i, K(k)+1)$ with the value N_e+1. The algorithm is:

```
for k = 1 to N_v
    K(k) ← 0

for j = 1 to N_e
    for i = 1 to I(j)
        k ← v(i, j)
        m ← K(k) + 1
        e(m, k) ← j
        K(k) ← m

for k = 1 to N_v
    e(K(k) + 1, k) ← N_e + 1
```

Note that the success of this algorithm does not depend on the order of the vertex indices in the element incidence sublists. In particular, then, its success does not depend on the vertex orientation requirements of C1 nor the edge consistency requirements of C2.

From this list, the algorithm constructs an element neighbors list, which is a list of N_e entries in which the jth entry is a sublist, $n(i, j)$, $i = 1, 2, \ldots, I(j)$, with $n(i, j)$ being the index of the neighbor of element j on its ith edge, or zero if this edge is a boundary edge. The algorithm for constructing this list consists of a synchronized scan over the element incidence list and the boundary reference table examining the edges of elements in order. If element j has a neighbor on its ith edge, of index $l > j$ then this can be determined by searching the vertex incidence sublists of vertices $v(i, j)$ and $v(\bar{i}+1, j)$, the endpoints of the ith edge, for a common entry (recall (2.1), $\bar{i} \equiv i \bmod I(j)$). This search is described in the next algorithm, which is labelled as a procedure returning the common entry's value in a variable named l so that it can be referred to in the subsequent part of this section. The variables of our discussion are regarded as global to the procedure, and comments are enclosed in face brackets.

(4.1)

```
procedure find the common entry [l]
v1 ← v(i, j)
ī ← i mod I(j)
v2 ← v(ī + 1, j)
p1 ← 1
p2 ← 1
while (e(p1, v1) ≦ j)      p1 ← p1 + 1
while (e(p2, v2) ≦ j)      p2 ← p2 + 1
while e(p1, v1) ≠ e(p2, v2)
    if e(p1, v1) < e(p2, v2)
      then p1 ← p1 + 1
      else p2 ← p2 + 1
l ← e(p1, v1)
```

The choice of guard entry at the end of each sublist ensures that the **while** loops terminate. If l is returned with value N_e+1, then no common entry was found. If $l \leqq N_e$ is returned, then it is known that $v(i, j)$ and $v(\bar{i}+1, j)$ appear in element incidence sublist for $E(l)$, but it is not assured that $v(\bar{i}+1, j)$ immediately precedes $v(i, j)$, as is necessary for $E(l)$ to be an edge consistent neighbour of $E(j)$ according to C2. The

following algorithm checks for a local edge numbers, m of $E(l)$ returning m if the mth edge of $E(l)$ is consistent with the ith edge of $E(j)$ and $-m$ otherwise.

(4.2)
```
procedure find complementary local edge number [m]
m ← 1
ī ← i mod I(j)
while (v(m, l) ≠ v(ī + 1, j))        m ← m + 1
m̄ ← m mod I(l)
if v(m̄ + 1, l) ≠ v(i, j) then m = −m
```

The verification of C2 requires a check that each edge of each element is either an internal edge or a boundary edge. This information can then be used to verify that the boundary reference table contains references to boundary edges exclusively and exhaustively. Initially, the neighbors list is set to zero.

The verification is then carried out as in algorithm (4.3) by a scan over the mesh edges in element order. The first step of the check is to establish the presence or absence of a neighbor on this edge. Procedure (4.1), find common entry, is used to determine the presence of a neighbor of index $l > j$. If one is found, then a check is made to ensure that a second neighbor of index less than j has not already been recorded in the neighbors list. If not, l is assigned to $n(i, j)$; the edge consistency between these neighbors is checked using (4.2), and the presence of element j as a neighbor of element l is recorded. This latter assignment is made regardless of edge consistency so that the mth edge of element l will not be interpreted as a boundary edge later in the scan. When the status of a neighbor for the ith edge of the jth element has been determined, the consistency of the boundary reference table can be assessed. A scan of this table is synchronized with the scan of element edges by maintaining a pointer, k, into the table. The discovery of violation of C2, or the inconsistency of the boundary reference table, is marked in the algorithm by comments, which in an implementation would be replaced by an error recording and/or reporting mechanism. If no violation of C2 is encountered, a valid element neighbors list is constructed. If the boundary reference table were not constructed a priori, then the section of (4.3) which validates it could be replaced by a section which constructs it.

(4.3)
```
for j = 1 to N_e      {scan elements}
  for i = 1 to I(j)      {scan edge of jth element}
    {establish presence of a neighbor}
    find common entry [l]      {via (4.1)}
    if l ≦ N_e
      then
        if n(i, j) ≠ 0
          then {violation of C2, multiple neighbors}
          else {one neighbor of index l > j}
            n(i, j) ← l
            find complementary local edge number [m] {via (4.2)}
            if m > 0
              then n(m, l) ← j
              else {violation of C2, edge inconsistency}
                   n(−m, l) ← j
    {check consistency of boundary reference table}
    if b(1, k) = j   and   b(2, k) = i
```

```
then
  if n(i, j) ≠ 0
    then {brt inconsistent, contains reference to internal edge}
  if k < N_b
    then  k ← k + 1
else
  if n(i, j) = 0
    then {brt inconsistent, reference to boundary edge omitted}
```

As a rough guide to the expected running time characteristics of (4.3), it can be seen from the looping structure that if the elements have a fixed, or bounded, number of sides, and a bounded number of them can be incident on any one vertex, then the running times can be expected to be linear with respect to N_e and N_b, with dependence on N_e being the dominant effect.

5. Verification of C3 and C4. Algorithm (4.3), which checks condition C2 of the mesh definition and the consistency of the boundary reference table, is the first of three major steps in verifying the mesh. The second involves a pairwise comparison of entries in the boundary reference table to check C4 and part of condition C3 of the definition. In this step, the number of independent boundary curves is determined for the third major step, which verifies condition C1 and the remaining part of C3 in a scan over the element list. The algorithms for these steps are designed so that they can each be run independently of the outcome of the previous ones to provide as much diagnostic information as possible about invalid meshes.

Several comments are in order before we embark on a more detailed discussion of these steps. In the preceding sections, the geometric shape of the finite elements of the mesh played little role so there was no benefit to restricting this shape beyond its being a general polygonal domain. However, such generality adds considerable complexity to the discussion of this section, which seems unwarranted since most of the meshes in common use consist of triangles and quadrilaterals. We shall discuss triangular elements, pointing out that the algorithm extends directly to elements which can be triangulated conveniently.

We have assumed that the lists to be verified do represent a valid collection of triangles, e.g., that the values of the entries in the incidence list and boundary reference table fall in the correct ranges, or that the entries in the vertex coordinate list are unique. Such a check would be a useful part of a mesh verification program, but conditions that the lists represent triangles are fairly obvious and can be checked by simple inspection algorithms.

The third condition of the definition, C3, requires that no interval of a boundary edge intersect any mesh element other than the one of which it is an edge. It can be seen from the proof of Theorem 1 in § 3 that the boundary edges of a collection of triangles satisfying conditions C2 and C4 of the definition form a set of simple oriented closed polygonal curves which do not cross at the vertices of the mesh. Suppose that one vertex P, of such a curve is known to lie outside each element except those on which it is incident. We shall refer to P as the starting vertex for this boundary curve, and we shall refer to the order in which the edges appear when the boundary curve is traverse from P in the direction of its orientation as curve order. If any edge in the curve intersects an element of the collection of triangles in the manner forbidden by C3, then there must be a first such edge in the curve order or edges, and this first edge must intersect some boundary edge. The intersection may occur at the endpoint (= vertex) of one of the two

edges involved, but not both, or a violation of C4 occurs. Hence condition C3 may be divided into two subcriteria.

(5.1) C3(i) Check one point of each boundary curve to ensure that it lies outside every element on which it is not incident.

(5.2) C3(ii) Check that no two boundary edges intersect unless they are consecutive edges of a boundary curve.

The second major step of the mesh verification algorithm described below at (5.6) checks criterion C3(ii) and condition C4. The outline of an algorithm for determining when two or more of a set of N_b line segments intersect which runs in times $O(N_b \log (N_b))$ has been described by Shamos and Hoey in [14], which also shows that this behavior is asymptotically optimal for this task. We describe here a simpler algorithm which can be expected to run in times $O(N_b^2)$, noting that the first step of the verification process is generally $O(N_e)$, and for most region shapes $O(N_b^2) = O(N_e)$.

An edge will be referred to as checked when criterion C3(ii) has been examined for it, which involves comparing the edge to every other unchecked boundary edge of the collection of triangles, except its predecessor and successor in curve order. The check of a boundary edge from the vertex of index p to that of index q uses the function

$$f_{p,q}(x, y) = (y(q) - y(p))(x - x(p)) - (x(q) - x(p))(y - y(p)) \tag{5.3}$$

which vanishes only for (x, y) on the line through the vertices and takes values of opposite sign for points on opposite sides of this line. If r and s are the indices of two other vertices, then the line segment from r to s intersects that from p to q only if

$$f_{p,q}(x(r), y(r))\, f_{p,q}(x(s), y(s)) \leqq 0 \quad \text{and} \tag{5.4}$$

$$f_{r,s}(x(p), y(p))\, f_{r,s}(x(q), y(q)) \leqq 0. \tag{5.5}$$

The boundary edges are checked along each boundary curve in curve order. For the scanning of a curve, three pointer variables into the boundary reference table are used; a pointer named start which marks the starting vertex of the first edge of a curve to be checked, a pointer, k, into this table is maintained to reference the edge currently being checked, and a pointer, next, is maintained to the successor of edge k. When next coincides with start, the curve has been scanned. If unchecked boundary edges remain, it is necessary to reinitialize start on a new boundary curve.

To keep track of which edges have been checked, algorithm (5.6) uses a temporary boolean array of length N_b named "checked" with "checked (k) = true" when the edge described by the kth entry of the boundary reference table has been checked. This step of the algorithm also determines the number of distinct boundary curves, and the indices of one vertex on each curve, which is used by the third step of the verification algorithm to check criterion C3(i). This information is stored in algorithm variable "$n$curves" and array "bv(i), $i = 1$ to $n$curves" respectively.

The algorithm (5.6) requires a valid boundary reference table, as provided from algorithm (4.3) of the preceeding section. As long as C2 and C4 are not violated, then each boundary edge has a unique successor (Theorem 1 of § 3) so that the algorithm can proceed to check all edges. However, if a violation of these conditions has occurred this is no longer guaranteed, which could result in an infinite loop developing during the scanning of a boundary curve's edges. Hence, the outer control structure of algorithm (5.6) is a "while" loop with two exit criteria, (a) $n$checked $= N_b$, regarded as the "normal" exit, although it may occur with invalid as well as valid meshes, and (b) next $= k$ indicating that no successor was found for the edge referenced by the kth entry

of the boundary reference table, which can only occur for an invalid mesh. Algorithm (5.6) requires the following initializations of variables:

```
                    ncurves      ← 0
                    nchecked     ← 0
                    k            ← 0
                    next         ← 1
                    start        ← 1
                    for m = 1    to   N_b
                      checked (m) ← false

  while nchecked < N_b   and   next ≠ k
    {check if current boundary curve is complete}
    if next = start
      then {initialize next to start of new boundary curve}
        next ← 1
        while checked (next)   next ← next + 1
        start ← next
        ncurves ← ncurves + 1
        bv(ncurves) ← v(b(2, start), b(1, start))
    k ← next
    p ← v(b(2, k), b(1, k))
    q ← v(b(2, k) + 1, b(1, k))
    for m = 1   to   N_b
      if m ≠ k {check mth boundary edge against kth}
(5.6)       then
          r ← v(b(2, m), b(1, m))
          s ← v(b(2, m) + 1, b(1, m))
          if p = r
            then {violation of C4}
          if q = r
            then {mth edge is successor of kth edge}
              next ← m
            else
              if not checked (m)
                then
                  if edges intersect {using (5.4)}
                    then {violation of C3 (ii)}
    nchecked ← nchecked + 1
    checked (k) ← true
    {end of while loop}
```

The third major step of the algorithm involves a scan over the elements to verify C1, i.e., that their vertices are listed in the element incidence list in counter clockwise order, and C3(i) of (5.1), i.e., the starting vertices of the boundary curves do not lie in elements other than those of which they are incident as vertices. These checks can be conveniently implemented for triangular elements using the familiar transformation to the area coordinates of an element. For completeness, we review the formulae for this transformation, which can be found, e.g., in [19]. Let the coordinates of the jth element be

$$x_k = x(v(k, j)), \qquad y_k = y(v(k, j)), \qquad k = 1, 2, 3 \tag{5.7}$$

and let

$$D = \det \begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix},$$

(5.8)

$$A_1 = (y_2 - y_3)/D, \quad B_1 = (x_3 - x_2)/D, \quad C_1 = (x_2 y_3 - x_3 y_2)/D,$$
$$A_2 = (y_3 - y_1)/D, \quad B_2 = (x_1 - x_3)/D, \quad C_2 = (x_3 y_1 - x_1 y_3)/D,$$
$$A_3 = (y_1 - y_2)/D, \quad B_3 = (x_2 - x_1)/D, \quad C_3 = (x_1 y_2 - x_2 y_1)/D.$$

Then the area coordinates relative to the triangle $E(j)$ of a point of the plane having Cartesian coordinates (x, y) are the three numbers

$$L_i(x, y) = A_i x + B_i y + C_i, \qquad i = 1, 2, 3. \tag{5.9}$$

Their significance to our discussion is that (x, y) lies in $E(j)$ if and only if $0 \leqq L_i(x, y) \leqq 1$ for $i = 1, 2, 3$. Moreover, the vertices of $E(j)$ form a triangle and are specified in counter clockwise order if and only if $D > 0$.

The scan for verifying C1 and C3(i) can be described by the following algorithm, using data concerning the number of boundary curves and a starting vertex on each, $n$curves and $bv(i)$, $i = 1$ to curves from (5.6) and the vertex incidence list, $e(k, n)$, $k = 1$ to $K(n)$, $n = 1$ to N_v from (4.3).

(5.10)

```
for j = 1 to N_e
  compute D {via (5.8)}
  if D ≦ 0
    then {violation of C1}
  for n = 1 to ncurves
      {check if element j is incident on vertex of index bv(n)}
    for k = 1 to K(bv(n))
      if j = e(k, bv(n))
        then incident ← true
    if not incident
      then {check if vertex of index bv(n) lies in element j}
        x ← x(bv(n))
        y ← y(bv(n))
        compute A1, B1, C1, L1(x, y)  {via (5.8)}
        if 0 ≦ L1 ≦ 1
          then
            compute A2, B2, C2, L2(x, y)  {via (5.8)}
            if 0 ≦ L2 ≦ 1
              then
                compute A3, B3, C3, L3(x, y)  {via (5.8)}
                if 0 ≦ L3 ≦ 1
                  then {violation of C3(i)}
```

6. Performance of a FORTRAN implementation. A FORTRAN implementation of the mesh verification algorithm has been prepared and some tests run on regular meshes to provide some evidence of run time performance, which we discuss here; some tests have been run as well to check out the various mesh errors detected, which are not reported here. The meshes for the performance tests comprise two

families, each characterized by a parameter N which is proportional to the number of elements in the mesh. One family is a collection of valid meshes on a hollow square, as shown in Fig. 6.1, and having $3\sqrt{N}$ triangle edges along each outer side of the square, and $\sqrt{N}$ triangle edges along the inner sides of the "hollow". The second family is a collection of invalid "meshes" created by folding the meshes of the first family along the line $y = x/3$, i.e., by applying the tranformation

$$(\bar{x}, \bar{y}) = \begin{cases} (x, y) & \text{if } y \geqq x/3, \\ \frac{1}{5}(4x+3y, 3x-4y) & \text{if } y < x/3. \end{cases}$$

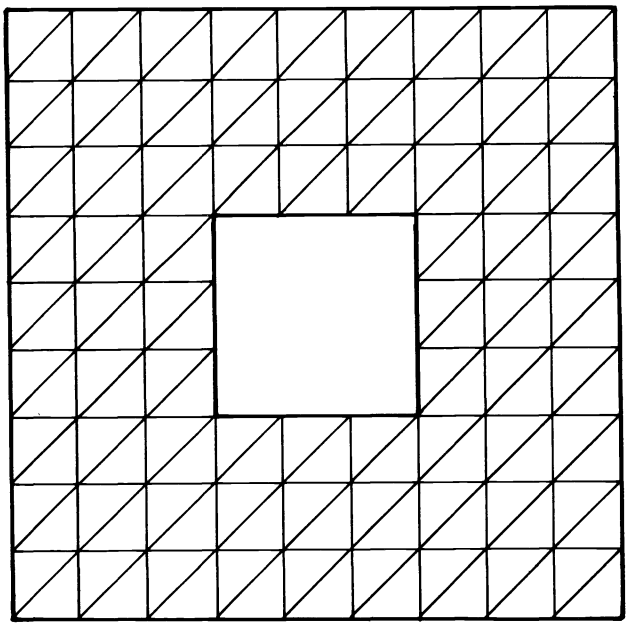


FIG. 6.1

The meshes are quite regular in the sense that 6 triangles meet at each interior vertex, and less than 6 meet at each boundary vertex; hence one would expect the algorithm's running time to be proportional to N. The Fortran implementation used for the test was compiled using the IBM optimizing compiler, Fortran H extended, and executed on the University of Waterloo's IBM 3031. For the tests involving the invalid meshes, the details of the mesh error were determined in each case, but the detailed output of these details was not performed.

The running time for the algorithm to check meshes 1600 to 4096 triangles are shown in Tables 6.1 and 6.2, along with slopes of successive line segments of the graph of running times versus N. The algorithm seems clearly to be behaving as linear in the number of triangles in the mesh, at least over this range of meshes. It can be seen that there is a slight premium to be paid for the determination of the nature of the errors in the invalid meshes.

Acknowledgment. The author, and one hopes the manuscript, have benefitted from referee's suggestions.

TABLE 6.1
Test timings for valid mesh on hollow square.

i	N_i	No. of Triangles	time = t_i (seconds)	Slope = $(t_i - t_{i-1})/(N_i - N_{i-1})$
1	100	1,600	3.7	-
2	144	2,304	5.3	.036
3	196	3,136	7.3	.038
4	256	4,096	9.5	.037

TABLE 6.2
Test timings for invalid (folded) mesh on hollow square.

i	N_i	No. of Triangles	time = t_i (seconds)	Slope = $(t_i - t_{i-1})/(N_i - N_{i-1})$	No. of errors reported
1	100	1,600	4.4	-	254
2	144	2,304	6.3	.043	375
3	196	3,136	0.5	.042	522
4	256	4,096	11.2	.045	692

REFERENCES

[1] L. V. AHLFORS, *Complex Analysis*, McGraw-Hill, New York, 1966.
[2] I. BABUSKA AND A. K. AZIZ, *On the angle condition in the finite element method*, SIAM J. Numer. Anal., 13 (1976), pp. 214–226.
[3] A. BYKAT, *Automatic generation of triangular grids*, I—*Subdivision of a general polygon into convex subregions*, II—*Triangulation of convex polygons*, Internat. J. Numer. Meth. Engrg., 10 (1976), pp. 1329–1342.
[4] J. A. CAVENDISH, *Automatic triangulation of arbitrary planar domains for the finite element method.* Internat. J. Numer. Meth. Engrg., 8 (1974), pp. 679–696.
[5] G. FORSYTHE, M. A. MALCOLM, AND C. B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, NJ., 1977.
[6] W. J. GORDEN AND C. A. HALL, *Construction of curvilinear coordinate systems and applications to mesh generation*, Internat. J. Numer. Meth. Engrg., 7 (1973), pp. 461–477.
[7] J. A. GEORGE, *Computer implementation of the finite element method*, Tech. Rep. STAN-CS-208, Stanford Univ., Stanford, CA, 1971.
[8] J. L. GROSS AND R. H. ROSEN, *A linear time planarity algorithm for 2-complexes*, J. Assoc. Comput. Mach., 26 (1979), pp. 611–617.
[9] HOPCROFT, J. AND TARJAN, R., *Efficient planarity testing*, J. Assoc. Comput. Mach., 21, (1974), pp. 549–568.
[10] C. L. LAWSON, *Software for C^1 surface interpolation*, in Mathematical Software III, J. R. Rice, ed., Academic Press, New York, 1977.
[11] B. A. LEWIS AND J. S. ROBINSON, *Triangulation of planar regions with applications.* Comp. J., 21 (1978), pp. 324–332.
[12] A. R. MITCHELL AND R. WAIT, *The Finite Element Method in Partial Differential Equations*, John Wiley, New York, 1977.
[13] E. G. SEWELL, *An adaptive computer program for the solution of Div($p(x, y)$ grad u) = $f(x, y, u)$ on a polygonal region*, in The Mathematics of Finite Elements and Applications II, J. R. Whiteman, ed., MAFELAP 1975; Academic Press, New York, 1976.
[14] M. I. SHAMOS AND D. HOEY, *Geometric intersection problems*, Proc. IEEE 17th Symposium on Foundations of Computer Science, 1976, pp. 208–215.
[15] R. B. SIMPSON, *Automatic local refinement for irregular rectangular meshes*, Internat. J. Numer. Meth. Engrg., 14 (1979), pp. 1665–1678.

[16] ———, *A survey of two dimensional mesh generation*, Proc. 9th Manitoba Conference on Numerical Mathematics and Computing, Utilitas Math. Publ., University of Manitoba, 1980.

[17] G. STRANG AND G. J. FIX, *An Analysis of the Finite Element Method*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[18] M. H. VAN EMDEN, *Programming with verification conditions*, IEEE Trans. Software Engrg., 5 (1979), pp. 148–159.

[19] O. C. ZIENKIEWICZ, *The Finite Element Method in Engineering Science*, McGraw-Hill, New York, 1971.

SIAM J. SCI. STAT. COMPUT.
Vol. 2, No. 4, December 1981

0196-5204/81/0204-0008 $01.00/0

A BIDIAGONALIZATION-REGULARIZATION PROCEDURE FOR LARGE SCALE DISCRETIZATIONS OF ILL-POSED PROBLEMS*

DIANNE P. O'LEARY† AND JOHN A. SIMMONS‡

Abstract. In this paper, we consider ill-posed problems which discretize to linear least squares problems with matrices K of high dimensions. The algorithm proposed uses K only as an operator and does not need to explicitly store or modify it. A method related to one of Lanczos is used to project the problem onto a subspace for which K is bidiagonal. It is then an easy matter to solve the projected problem by standard regularization techniques. These ideas are illustrated with some integral equations of the first kind with convolution kernels, and sample numerical results are given.

Key words. ill-posed problems, Lanczos algorithm, regularization, first kind integral equation

1. Introduction. In this paper we discuss techniques applicable to the solution of those ill-posed problems which, upon discretization, give rise to large linear systems of equations. In particular, our examples are drawn from integral equations of the first kind,

$$\int_a^b k(s,t)f(t)\,dt = g(s)$$

or

$$\min_f \left\| \int_a^b k(s,t)f(t)\,dt - g(s) \right\|^2,$$

which discretize to the linear system

$$Kf = g, \qquad K: m \times n, \quad m \geqq n,$$
$$f: n \times 1,$$
$$g: m \times 1$$

or to the minimization problem

$$\min_f \|Kf - g\|_2^2.$$

Such continuous problems are characterized by the fact that small changes in the function g can cause large changes in f. This is reflected in the discrete problem by ill-conditioning in the matrix K. Since such perturbations can be due to unavoidable noise in measurements of g or to roundoff errors in the calculation, algorithms for numerical solution of the discretized problem must be designed to minimize the effects of these perturbations.

Various techniques for solving linear ill-posed problems are discussed in a good survey paper by Björck and Eldén [3]. We do not attempt an exhaustive review of

* Received by the editors September 29, 1980.

† Mathematical Analysis Division, National Bureau of Standards, Washington, DC 20025, and Computer Science Department and Institute for Physical Science and Technology, University of Maryland, College Park, Maryland 20742.

‡ Metallurgy Division, National Bureau of Standards, Washington, DC 20025.

these techniques here, but note that two of the most popular methods are based on the following techniques:

(1) *Regularization.* We use this term in a broad sense [3] to describe methods which replace the original operator by a related one which diminishes the effects of errors in the data. For example, the function to be minimized might be replaced by

$$\min_f \|Kf-g\|_2^2+\gamma\|f\|_L^2,$$

where $\|f\|_L^2=f^*L^*Lf$ for some full rank matrix L, γ is a positive scalar parameter, and the superscript * denotes complex conjugate transpose. This is equivalent to solving the system of equations

$$(K^*K+\gamma L^*L)f=K^*g.$$

Thus the operator K^*K of the normal equations for the original problem has been replaced by an operator $K^*K+\gamma L^*L$. The choices of γ and L, guided by the physical characteristics of the problem and of the noise, give a problem for which the operator is better conditioned but which has a solution close, in some sense, to that of the original problem.

(2) *Projection.* The approximate solution f is constrained to lie in a specified subspace given by the columns of a matrix V. In this case we have a modified problem,

$$\min_h \|KVh-g\|_2^2$$

or

$$V^*K^*KVh=V^*K^*g, \qquad f=Vh.$$

The new operator, V^*K^*KV, is K^*K restricted to a subspace upon which it is better conditioned.

The technique we consider is a projection-regularization method. In the first step, the problem is projected onto a subspace defined by a bidiagonalization algorithm. The restricted operator is typically still ill-conditioned. In the second step a regularization is applied on the subspace. The reason for this approach is that regularization of the restricted problem can be less expensive and, if the subspace is chosen properly, the final results are not significantly degraded. The algorithm and its properties are presented in § 2. Potential applications of the algorithm include:

(i) Problems for which n multiplications by the operator K are significantly less expensive than factorization of the matrix.

(ii) Problems for which storage does not permit regularization of the original problem.

Thus, problems for which K is sparse or K is structured so that its storage and matrix-vector multiplication time are both less than $O(mn)$ are possible candidates for this algorithm. Examples of such problems and sample computational results are given in § 4.

2. The bidiagonalization-regularization algorithm. The algorithm proceeds in two steps. First the problem is reduced to one on a subspace which is computationally much more economical. Then standard techniques of regularization can be used on the reduced problem. The two steps are defined in §§ 2.1 and 2.2, and properties of the algorithm are discussed in § 2.3. A different algorithm based on bidiagonalization has been proposed independently in [2].

2.1 The Lanczos bidiagonalization. The subspace chosen over which to solve the problem is that generated by the "Lanczos" algorithm for bidiagonalization. This algorithm was investigated by Paige [20], named and described in block form by Golub, Luk and Overton [11] and used in a different context by Moler and Stewart [19]. It is a specific computational implementation of the bidiagonalization procedure of Golub and Kahan [10], proposed as the first step of computing the singular value decomposition of a matrix. It is related to ideas of Lanczos [16] and [17].

The Lanczos bidiagonalization algorithm takes an $m \times n$ matrix K and factors it as

$$U^*KQ = B, \qquad \begin{aligned} &U: m \times m, \\ &B: m \times n, \\ &Q: n \times n \end{aligned}$$

or

$$K = UBQ^*,$$

where U and Q are orthonormal,

$$U^*U = I, \qquad Q^*Q = I,$$

and B is bidiagonal,

$$B = \begin{bmatrix} \alpha_1 & \beta_1 & & & & \\ & \alpha_2 & \beta_2 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \beta_{n-1} & \\ & & & & \alpha_n & \\ \hline & & & 0 & & \end{bmatrix}.$$

This is done using very elementary manipulations: we need to form the product of K and K^* with various vectors and to take linear combinations and inner products of vectors. Furthermore, rather than carrying out the full algorithm, it can be terminated early to give a factorization of K as an operator over a subspace:

$$U_k^*KQ_k = B_k, \qquad \begin{aligned} &U_k: m \times k, \\ &Q_k: n \times k, \\ &B_k: k \times k, \end{aligned}$$

where

$$B_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \beta_{k-1} \\ & & & & \alpha_k \end{bmatrix},$$

$$U_k^*U_k = I, \qquad Q_k^*Q_k = I.$$

The matrix B has the same singular values as K, and the singular values of B_k can be shown to be close to certain of K's, typically its largest and smallest [11].

The algorithm can be derived from the relations $KQ = UB$ and $U^*K = BQ^*$. It proceeds as follows [11].

Given K: $m \times n$ and z_1: $n \times 1$ an arbitrary nonzero vector, set

$$q = \frac{z_1}{\|z_1\|}, \quad y_1 = Kq_1, \quad \alpha_1 = \|y_1\|, \quad u_1 = \frac{y_1}{\alpha_1}.$$

For $i = 1, 2, \cdots, k-1$,

$$z_{i+1} = K^* u_i - \alpha_i q_i, \qquad \beta_i = \|z_{i+1}\|,$$

$$q_{i+1} = \frac{z_{i+1}}{\beta_i},$$

$$y_{i+1} = Kq_{i+1} - \beta_i u_i, \qquad \alpha_{i+1} = \|y_{i+1}\|,$$

$$u_{i+1} = \frac{y_{i+1}}{\alpha_{i+1}}.$$

The vector $q_i(u_i)$ is the ith column of the matrix $Q(U)$.

Operations counts and properties of the algorithm will be discussed in § 2.3.

2.2 Regularization of the bidiagonal problem. Our original discretized problem was to solve

$$\min_f \|Kf - g\|_2^2.$$

If we decide to consider only those vectors f contained in the subspace spanned by the columns of Q_k, i.e.,

$$f = Q_k h \quad \text{for some } h\text{: } k \times 1,$$

and try to minimize the residual only in the directions spanned by the columns of U_k, we have the reduced problem

$$\min_h \|U_k^* (KQ_k h - g)\|_2^2 = \min_h \|B_k h - U_k^* g\|_2^2.$$

This problem is typically still highly ill-conditioned, because while the matrix B_k usually contains very good approximations to the large singular values of K [11], it also has singular values which are rough approximations to the small ones. Thus forming and solving the normal equations

$$B_k^* B_k h = B_k^* U_k^* g$$

will probably not give an acceptable solution. On the other hand, we have reduced our problem to one of smaller dimension having a matrix that is bidiagonal. Thus, any of the standard regularization or projection procedures [3] will be economical. For example, we can use a truncated singular value decomposition, (SVD) expressing

$$B_k = V \Sigma W^*, \quad W^* W = I, \quad V^* V = I,$$

$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_k), \quad \sigma_1 \geqq \sigma_2 \geqq \cdots \geqq \sigma_k \geqq 0,$$

defining a truncated pseudo-inverse for Σ,

$$\Sigma_r^+ = \operatorname{diag}\left(\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_r}, 0, \cdots, 0\right),$$

and then setting

$$h = W\Sigma_r^+ V^* U_k^* g = \sum_{i=1}^{r} \frac{v_i^* U_k^* g}{\sigma_i} w_i,$$

where $w_i(v_i)$ is the ith column of $W(V)$, $\Sigma_r = \operatorname{diag}(\sigma_1, \cdots, \sigma_r, 0, \cdots, 0)$, $\sigma_r > 0$, and the superscript "+" denotes the pseudo-inverse. This method might be considered either a regularization method or a projection method.

As another alternative we could take a damped SVD approach, substituting α_i/σ_i for $1/\sigma_i$ in the above expression for h, where the α_i form a sequence of decreasing positive numbers. The damping factors might be determined using generalized cross validation techniques [9], [28].

These SVD approaches are $O(k^3)$ processes. Even more efficient methods have been suggested by Eldén [6] [7] and Gander [8], basing the regularized solution directly on the bidiagonal form and producing a solution in $O(k^2)$ operations.

2.3 Properties and extensions of the algorithm. For definiteness in the discussion, we assume in this section that a truncated singular value regularization is performed.

(A) The operations count for the Lanczos bidiagonalization is $3(n+m)k$ multiplications plus $2(n+m)k$ additions plus k matrix-vector multiplications involving K and $k-1$ involving K^*. The SVD, as mentioned above, is $O(k^3)$, and the final solution is calculated in $(n+m)k$ additions and multiplications. Thus if k is small compared with n, significant savings can be realized with respect to the cost of the full singular value decomposition algorithm, an $O(n^2 m)$ process.

(B) The Lanczos bidiagonalization can easily be implemented with $2n+2m+3k$ storage locations in fast store (q, K^*u, u, Kq, α, β, U^*g), with the vectors q saved in auxiliary storage as they are computed. For the singular value decomposition, $2k^2+k$ locations are required for V, W and Σ. To compute the solution, these three matrices are used to form h, and then f is formed in $O(nk)$ operations with access to the vectors q. Thus, for efficient implementation, the storage requirement is that required for K plus $\max(2n+2m+3k, 2k^2+2k)$ in main storage, plus nk in auxiliary to be written and read once sequentially. (By rearranging the algorithm slightly the storage of V could be avoided.)

(C) If several problems are to be solved involving the same operator K, it can be more economical to also save the u vectors. In this way, the Lanczos iteration and the singular value decomposition need to be performed only once.

(D) If there is not enough storage available to store the q sequence, the q vectors can be regenerated by running the Lanczos algorithm again using the same starting vector.

(E) For very ill-conditioned problems, reorthogonalization of the q and u vectors may be necessary to preserve computational stability [22], [5], [24]; although in theory $q_i^* q_j = u_i^* u_j = 0$, $i \neq j$, in practice this may be far from being satisfied. In this case it may be necessary to perform modified Gram-Schmidt orthogonalization on the sequences. As each z_i is calculated, we would perform the iteration

$$\text{For } j = i-1, i-2, \cdots, 1, \quad z_i \leftarrow z_i - (z_i^* q_j) q_j,$$

before normalizing z_i. A similar process would be performed on the y sequence. This reorthogonalization can be done in a selective manner, depending on the magnitude of the inner product of the ith and jth vectors, or by running the loop from $i-1$ to $i-s$ for some $s<i$. In the case of full reorthogonalization the added cost is $(n+m)k^2$.

(F) There are two arbitrary parameters in the method: the number of Lanczos steps (k) and the number of retained singular values (r). The choice of these is problem dependent, and needs further investigation. Limited computational experience is summarized in § 4.

(G) For certain singular value distributions, it may be advantageous to use the block Lanczos algorithm of Palmer [23] or Golub, Luk and Overton [11]. This can improve the convergence by reducing the number of vectors q and u necessary for an acceptable solution. The number of accesses to K and K^* is reduced by forming their products with several vectors at once; thus the method is also useful, regardless of the singular value distribution, if K is stored in secondary memory and if there is room for one or more extra pairs of vectors of dimension n and m in main memory.

(H) "Preconditioning" techniques can be incorporated either to accelerate convergence or to change the character of the approximate solution. These two uses of preconditioning are described below.

(i) The problem

$$\min_f \|Kf-g\|_2^2$$

is equivalent to

$$\min_{\hat{f}} \|KN\hat{f}-g\|_2^2,$$

where N is any nonsingular $n\times n$ matrix and $f=N\hat{f}$. If the singular value spectrum of KN has better clustering of the singular values than does K, the number of Lanczos steps necessary can be significantly reduced. The price paid is an extra matrix-vector multiplication by N and by N^* at each iteration. This idea has been exploited previously in the solution of linear systems (e.g., [1], [4]) and certain least squares problems [26].

(ii) A common technique in many of these problems is "filtering" [13]. In this case K and g are both preconditioned by some operator M, changing the problem to

$$\min_{\tilde{f}} \|MK\tilde{f}-Mg\|_2^2.$$

Again the convergence rate is improved if the spectrum of MK has good clustering properties. The regularized solution $\tilde{f}$ will be different from the f above. The operator M should be chosen to filter out undesirable components of K and g, for example, highly oscillatory modes which are due to random errors in the measurements.

3. A time series deconvolution problem. We consider a special class of problems for which the algorithm in the previous section is applicable. In the case of interest, the integral equation

$$\int_a^b k(s,t)f(t)\,dt = g(s)$$

has a kernel of convolution type

$$k(s,t)=k(s-t)$$

and $b=s$.

Upon discretization, our problem becomes $Kf = g$, where

$$K = \begin{bmatrix} a_0 & & & \\ a_1 & a_0 & & \\ \vdots & \vdots & \ddots & \\ a_{n-1} & a_{n-2} & \cdots & a_0 \end{bmatrix}.$$

Thus K is lower triangular and Toeplitz. The time series involved are "causal," meaning that $k(t)$, $f(t)$ and $g(t)$ are zero for negative values of their arguments. If k is continuous, then a_j is small for small values of j, and these small numbers near the main diagonal of K cause ill-conditioning and prevent us from solving the system accurately by a simple forward substitution algorithm or by the faster algorithms for solving Toeplitz systems (e.g., [12]).

To understand the nature of this ill-conditioning, it is convenient to study a permuted version of the system, formed by reordering the equations from last to first:

$$\begin{bmatrix} a_{n-1} & a_{n-2} & \cdots & a_0 \\ \vdots & \iddots & & \\ a_0 & & & \end{bmatrix} \begin{bmatrix} f_0 \\ \vdots \\ f_{n-1} \end{bmatrix} = \begin{bmatrix} g_{n-1} \\ \vdots \\ g_0 \end{bmatrix}.$$

We denote this reordered system by $K^* f = g^*$ and note that $K = PK^*$, where P is the permutation matrix formed by writing the rows of the identity matrix from last to first. K^* is a Hankel matrix and is indefinite.

Now K is a defective matrix; it has n eigenvalues equal to a_0 but only one eigenvector, the last unit vector $e_n = [0, \cdots, 0, 1]^T$. But K^* is symmetric and thus has a full set of orthonormal real eigenvectors w_i with eigenvalues σ_i:

$$K^* w_i = \sigma_i w_i, \qquad i = 1, 2, \cdots, n$$

or

$$K^* = W \Sigma W^T,$$

where

$$W = [w_1, \cdots, w_n], \qquad W^T W = I,$$

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \sigma_n \end{bmatrix},$$

$$|\sigma_1| \geqq |\sigma_2| \geqq \cdots \geqq |\sigma_n| \geqq 0.$$

Thus $K = PK^* = (PW)\Sigma W^T$ and this, aside from the signs of the σ_i and those of the columns of PW, is the singular value decomposition of K. Therefore, we have a representation of K as a sum of n rank one matrices:

$$K = \sum_{i=1}^{n} \sigma_i w_i^* w_i^T, \qquad w_i^* = P w_i.$$

If K is nonsingular ($a_0 \neq 0$), its inverse is

$$K^{-1} = \sum_{i=1}^{n} \frac{1}{\sigma_i} w_i^* w_i^T.$$

If K is singular ($a_0 = 0$), we have the pseudo-inverse

$$K^+ = \sum_{i=1}^{r} \frac{1}{\sigma_i} w_i^* w_i^T,$$

where $|\sigma_r| > 0$, $\sigma_{r+1} = \cdots = \sigma_n = 0$.

From the decomposition above we can isolate the source of trouble in solving the linear system $K^* f = g^*$. (A similar analysis could be performed for the continuous problem.) Suppose that $a_0, \cdots, a_{s-1}$ are zero. Then

$$K^* = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}.$$

This matrix has $n - s$ eigenvalues equal to those of A, with eigenvectors of the form

$$\begin{bmatrix} \bar{w}_i \\ 0 \end{bmatrix}, \qquad \bar{w}_i : (n-s) \times 1$$

and s zero eigenvalues equal to those of A, with eigenvectors

$$\begin{bmatrix} 0 \\ e_i \end{bmatrix},$$

where the e_i are $s \times 1$ unit vectors. If $a_0, \cdots, a_{s-1}$ are small rather than zero, the eigenstructure will be similar, but all of the zeros will be perturbed slightly [29]. The unregularized solution to the problem is given by

$$f = \sum_{i=1}^{n} \frac{1}{\sigma_i} w_i^T g^* w_i^*.$$

Thus there will be large contributions to the solution from the small singular values if g^* has any components in a direction $[0, e_i]^T$, $i = 1, \cdots, s$. This means that noise in the first part of the vector g will be magnified by large factors and added to the last part of the solution vector f.

In some of these problems it was observed that the Fourier vectors are approximate eigenvectors for K^*. This can be explained by considering the problem in the frequency domain rather than the time domain. Let the matrix F be defined by

$$f_{jl} = \frac{1}{\sqrt{n}} e^{2\pi i j l / n}, \qquad l, j = 0, 1, \cdots, n-1, \quad i = \sqrt{-1}.$$

Then $F^* F = I$. Suppose the sequence $(a_{n-1}, \cdots, a_0)$ is band-limited; i.e.,

$$F^* \begin{bmatrix} a_{n-1} \\ \vdots \\ a_0 \end{bmatrix} = \sum_{j \in J} \delta_j e_j,$$

where J is an index set of cardinality m, small compared with n. Then if the norm of the vector $[a_{m-1}, \cdots, a_0]$ is small compared with the norm of the full vector $[a_{n-1}, \cdots, a_0]$, the columns of F corresponding to indices in J are an approximate invariant subspace of K; i.e., there is a subset of eigenvectors of K whose span is approximately that of those columns. To see this we form

$$F^* K^* F = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_4 \end{bmatrix}, \qquad C_1 : m \times m$$

(where F is reordered if necessary to put the columns with indices $j \in J$ first) and show that $\|C_3\|$ is small. Then, by [27, Thm. 4.11], the desired result follows. Now

$$F^*K^* = \left[\delta, \delta + \begin{bmatrix} \nu_1 \\ \mu_1 \end{bmatrix}, \cdots, \delta + \begin{bmatrix} \nu_{n-1} \\ \mu_{n-1} \end{bmatrix}\right],$$

where $\delta = (\delta_1, \cdots, \delta_m, 0, \cdots, 0)^T$ and the vectors ν_i are $m \times 1$. Now, the norms of the first few vectors μ_i are small because the first few columns of K^* differ from its first column by small perturbation terms. The norms of later vectors μ_i are small because the norm of $\delta^T + [\nu_i^T, \mu_i^T]$ is equal to the norm of the ith column of K, which is small compared with the norm of the first column. Thus F^*K^* has small entries in all rows not indexed by J, and therefore F^*K^*F is also small in those rows.

This Fourier property of the eigenvectors means that a truncated singular value decomposition often gives results indistinguishable from filtering.

In using the algorithm of § 2, we can take advantage of three simplifications:

(1) Since K^* is symmetric, we can use the Lanczos tridiagonalization procedure [17], [21] rather than bidiagonalization. This requires approximately half of the work and storage, since the u sequence is redundant, and only one matrix multiplication is needed per step. The resulting algorithm is:

Given K^* and z_1: $n \times 1$, an arbitrary nonzero vector, set

$$q_1 = \frac{z_1}{\|z_1\|}, \qquad \Psi_1 = 1.$$

For $i = 1, 2, \cdots, k-1$

$$z_{i+1} = K^* q_i - \theta_i q_i - \Psi_i q_{i-1},$$
$$\theta_i = q_i^T K^* q_i,$$
$$\Psi_{i+1} = \|z_{i+1}\|,$$
$$q_{i+1} = \frac{z_{i+1}}{\Psi_{i+1}}.$$

The algorithm performs better numerically if θ_i is calculated as $q_i^T(K^* q_i - \Psi_i q_{i-1})$.

(2) Since the resulting B is symmetric tridiagonal rather than bidiagonal, we can compute its eigendecomposition rather than its singular value decomposition. This cuts the storage required from $2k^2 + k$ to $k^2 + k$, and also involves less computation.

(3) There exist fast algorithms for forming the product of K^* with an arbitrary vector. This arises from the relation of K to a circulant matrix. Let

$$K_c = \begin{bmatrix} a_0 & & & & a_{n-1} & \cdots & a_1 \\ a_1 & a_0 & & & & & \vdots \\ \vdots & \vdots & \ddots & & & & \\ a_{n-1} & a_{n-2} & \cdots & a_0 & & & a_{n-1} \\ & a_{n-1} & \cdots & a_1 & a_0 & & \\ & & \ddots & & \ddots & & \\ & & & a_{n-1} & & \cdots & a_0 \end{bmatrix}.$$

Then K_c is a square circulant matrix of dimension $p \geqq 2n-1$. The eigenvectors of any circulant matrix are the Fourier vectors, the columns of the $p \times p$ matrix F defined

above. The eigenvalues are given by the components of the Fourier transform of its first column:

$$F \begin{bmatrix} \alpha_0 \\ \vdots \\ a_{n-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \lambda_1 \\ \cdot \\ \cdot \\ \cdot \\ \lambda_p \end{bmatrix} .$$

Thus, $K_c = F\Lambda F^*$ where $\Lambda = \operatorname{diag}(\lambda_1, \cdots, \lambda_p)$. Note that to form the product of K with an arbitrary vector x, we can form the product of K_c with the vector x padded by zeros, and take the first n components of the result. To do this, the vector

$$z = K_c \begin{bmatrix} x \\ 0 \end{bmatrix} = F\Lambda F^* \begin{bmatrix} x \\ 0 \end{bmatrix}$$

is calculated in three steps:

(a) $\hat{x} = F^* \begin{bmatrix} x \\ 0 \end{bmatrix}$,

(b) $y_i = \lambda_i \hat{x}_i$, $i = 1, 2, \cdots, p$,

(c) $z = Fy$.

Then Kx is given by the first n components of z. Step (b) costs p operations, while (a) and (c) are Fourier transforms of p-vectors and cost $O(p \log_2 p)$, for example, if p is chosen to be a power of 2.

4. Numerical examples. Two sample problems were chosen in order to demonstrate the algorithms of § 2. The experiments were run in single precision on a Univac 1108 computer. In each case the initial Lanczos vector z_1 was taken to be g. Machine storage limited the number of Lanczos vectors for the largest problem to be less than or equal to 40, so $k = 25$ and 40 were taken as representative values. The number of singular values to be dropped was guided by the uncertainty in the data, but further investigation would be needed to make the procedure automatic.

Example 1. The following integral equation has been studied, for example, in [18], [25]:

$$\int_{-6}^{6} k(s, t) f(t)\, dt = g(s), \qquad |s| \leqq 6,$$

$$k(s, t) = \begin{cases} 1 + \cos \dfrac{(t-s)\pi}{3}, & |t-s| \leqq 3, \quad |s| \leqq 6, \\ 0, & \text{otherwise}, \end{cases}$$

$$g(s) = \begin{cases} (6 - |s|)\left(1 + \dfrac{1}{2} \cos \dfrac{s\pi}{3}\right) + \dfrac{9}{2\pi} \sin \dfrac{|s|\pi}{3}, & |s| \leqq 6, \\ 0, & \text{otherwise}, \end{cases}$$

Solution: $f(s) = k(s, 0)$.

The right-hand side g and the solution f are plotted in Fig. 1. The integral equation was discretized using the trapezoidal rule with steplength $12/(n-1)$. This yields a

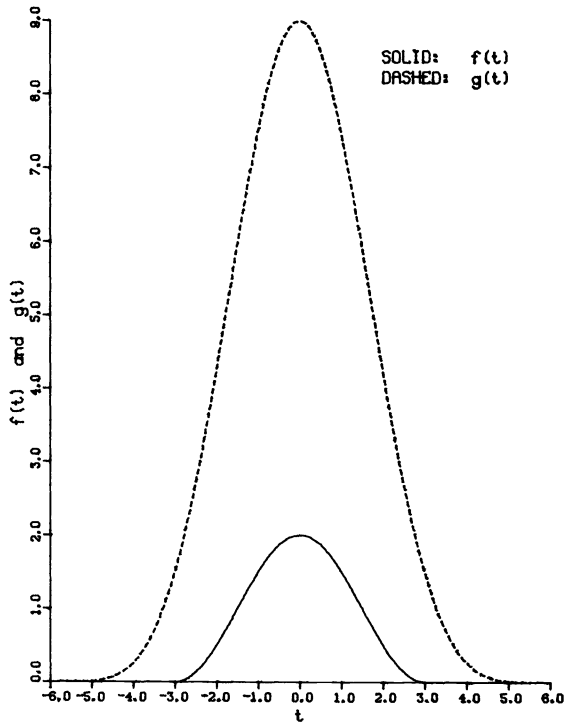


FIG. 1. *Example* 1, *functions f and g.*

symmetric banded Toeplitz matrix if the first and last unknowns are replaced by their values divided by 2; see [18] for details. The discretization of f does not satisfy the linear system exactly, but this discretization was taken as the "true" solution to the problem. The Lanczos algorithm with truncated singular value regularization was run with no reorthogonalization of the Lanczos vectors.

This problem is only mildly ill-conditioned. For $n = 25$ the calculated condition number (the ratio of the largest and smallest nonzero singular values) was $O(10^2)$, and the maximum error after one singular value was dropped was less than 3×10^{-3}. For $n = 49$, the estimated condition number based on $k = 40$ Lanczos steps was also of this order. In this case, best results for the solution (giving maximum error less than 10^{-3}) were obtained by dropping no singular values.

Figure 2 shows the results for $n = 97$ with $k = 25$ and 40. The condition number for $k = 40$ was $O(10^3)$. One singular value was dropped for $k = 40$, none for $k = 25$.

Example 2. This is a time series problem drawn from the field of acoustic emission. The kernel, plotted in Fig. 3, is the theoretical displacement response of a certain horizontal elastic plate [14] to a vertical step function force term applied at a point on one of the faces. The response is measured directly below the force, on the opposite face of the plate. The kernel was sampled at $n = 512$ points, and convolved numerically with a discretization of the function shown in Fig. 7. The resulting function, truncated to eight bits, was used as g. Elements in g ranged from 0 to 53, and the residual norm $\|Kf - g\|_\infty$ was 10^{-1}. Figure 4 shows the results of using forward substitution on the linear system $Kf = g$ to solve the time series deconvolution problem. This demonstrates the need for regularization of the solution.

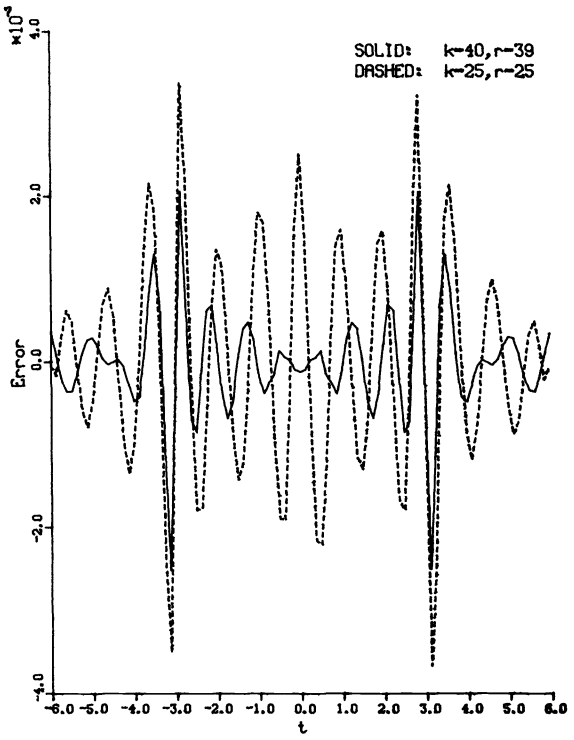


FIG. 2. *Example* 1, *error in computed solution,* $n = 97$.

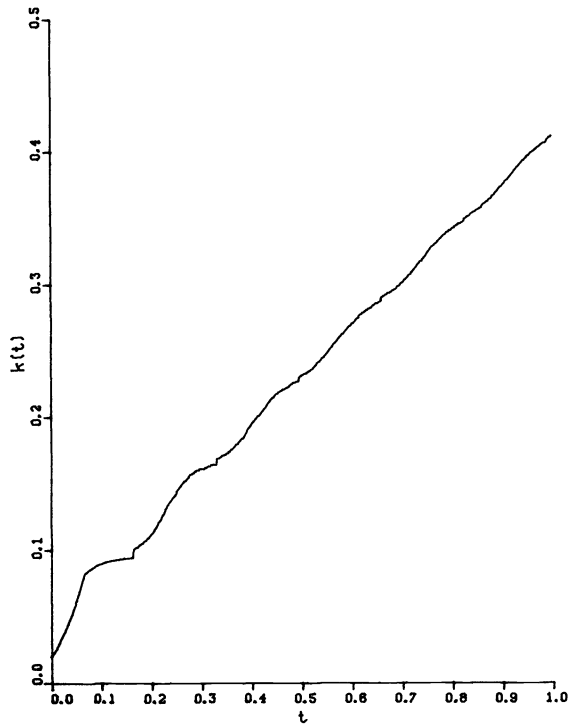


FIG. 3. *Example* 2, *kernel function.*

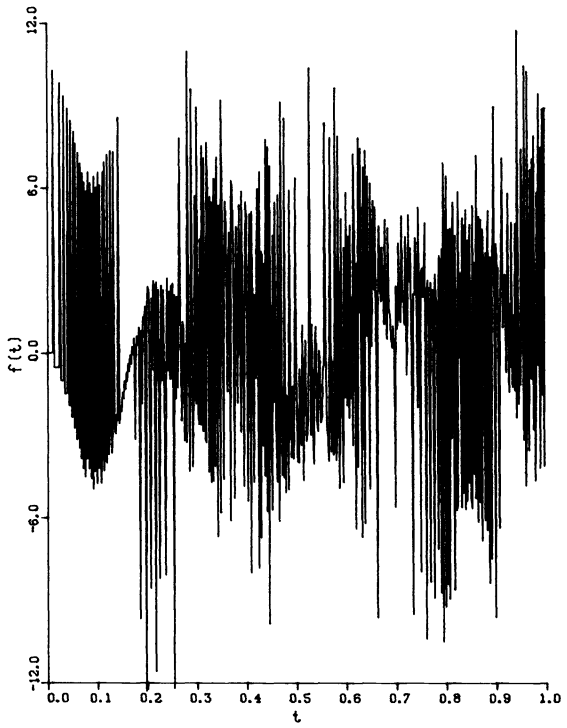


FIG. 4. *Example 2, forward substitution solution.*

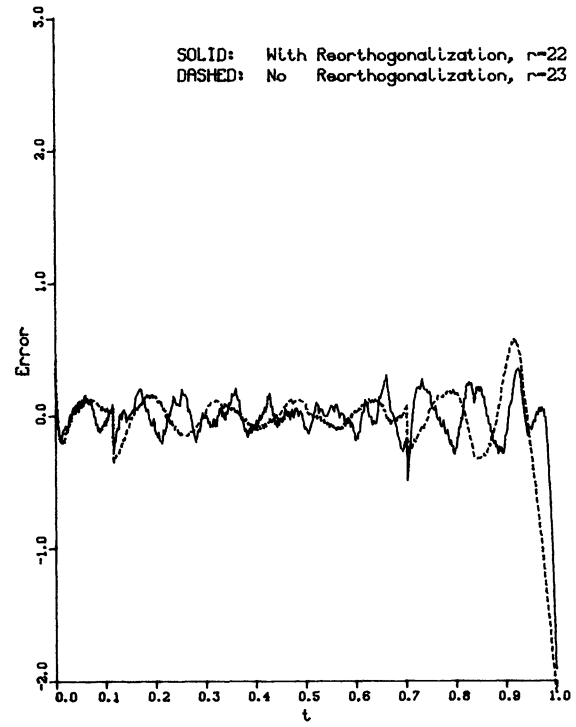


FIG. 5. *Example 2, error in computed solution,* $k = 25$.

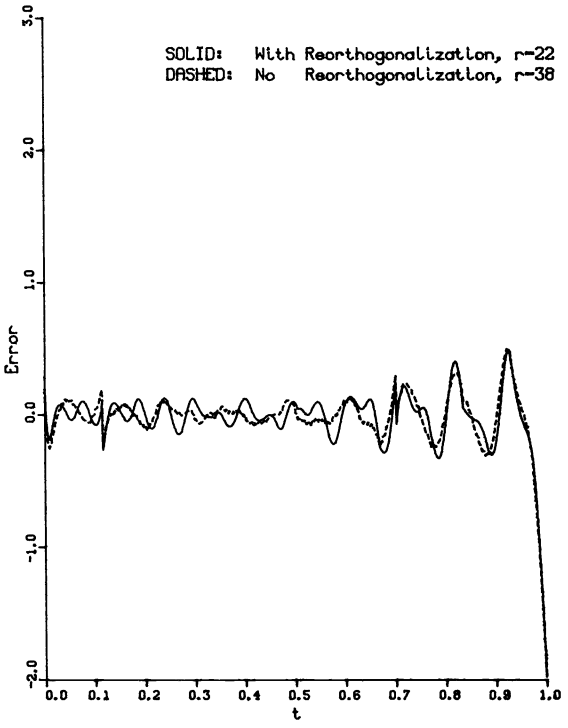


FIG. 6. *Example 2, error in computed solution,* $k = 40$.

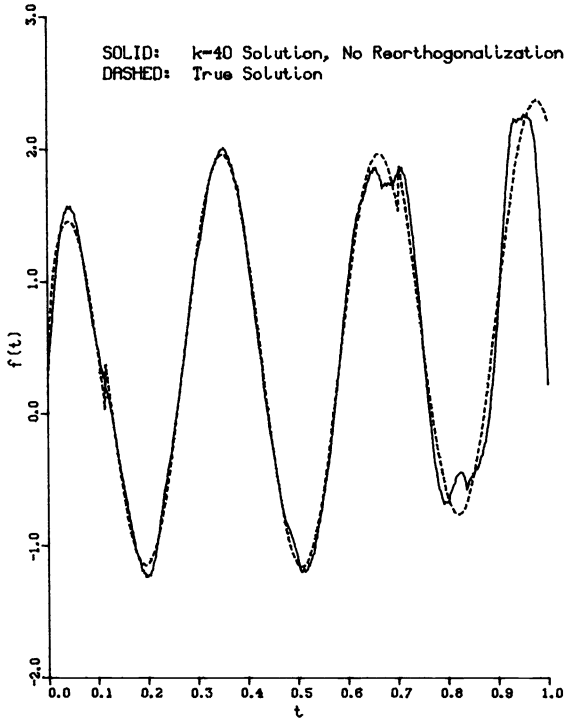


FIG. 7. *Example 2, sample solution.*

The results in Fig. 5 were obtained using $k = 25$ steps of the Lanczos algorithm with and without complete reorthogonalization. Three singular values were dropped in the calculation with reorthogonalization, two without. The estimated condition numbers were $O(10^4)$.

Analogous results for $k = 40$ are shown in Fig. 6, dropping 18 singular values when reorthogonalization is performed, and 2 without reorthogonalization. Figure 7 is a comparison of the true and computed solutions without reorthogonalization. By this stage, significant roundoff error has entered into the nonreorthogonalized process, as evidenced by the appearance of spurious copies of several singular values. See [5], [15] for an explanation of this phenomenon. This meant that fewer small singular values appeared, but despite the complete loss of orthogonality of the Lanczos vectors, a good solution vector could still be reconstructed.

Acknowledgment. We are grateful to Martin R. Cordes for very helpful discussions, for expert assistance with mathematical software and for his careful reading of the manuscript. Helpful comments were made by G. W. Stewart and Jane Cullum. Some of this work was carried out under a joint program in acoustic emission studies between the National Bureau of Standards and the Electric Power Research Institute.

REFERENCES

[1] O. AXELSSON, *Solution of linear systems of equations: iterative methods*, in Sparse Matrix Techniques, Copenhagen 1976, V. A. Barker, ed., Springer-Verlag, New York, 1977, pp. 1–51.

[2] ÅKE BJÖRCK, *A bidiagonalization algorithm for solving ill-posed systems of equations*, Rep. LiTH-MAT-80-33, Dept. Mathematics, Linköping University, Linköping, Sweden, 1980.

[3] ÅKE BJÖRCK AND LARS ELDÉN, *Methods in numerical algebra for ill-posed problems*, Rep. LiTH-MAT-R-33-1979, Dept. Mathematics, Linköping University, Linköping, Sweden, 1979.

[4] P. CONCUS, G. H. GOLUB AND D. P. O'LEARY, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–322.

[5] J. CULLUM AND R. A. WILLOUGHBY, *The Lanczos tridiagonalization and the conjugate gradient algorithms with local, ε-orthogonality of the Lanczos vectors*, Rep. RC 7152, IBM T. J. Watson Research Center, Math. Sciences Dept.,Yorktown Heights, NY, 1978.

[6] LARS ELDÉN, *Algorithms for the regularization of ill-conditioned least squares problems*, BIT, 17 (1977), pp. 134–145.

[7] ———, *A program for interactive regularization part I: Numerical algorithms*, Rep. LiTH-MAT-R-79-25, Dept. Mathematics, Linköping University, Linköping, Sweden, 1979.

[8] W. GANDER, *On the linear least squares problem with a quadratic constraint*, Tech. Rep. STAN-CS-78-697, Computer Science Dept., Stanford University, Stanford, CA, 1978.

[9] G. H. GOLUB, M. HEATH AND G. WAHBA, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.

[10] G. H. GOLUB AND W. KAHAN, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[11] G. H. GOLUB, F. T. LUK AND M. L. OVERTON, *A block Lanczos method to compute the singular values and corresponding singular vectors of a matrix*, Tech. Rep. STAN-CS-77-635, Computer Science Dept., Stanford University, Stanford, CA, 1977.

[12] F. G. GUSTAVSON AND D. Y. Y. YUN, *Fast algorithms for rational hermite approximation and solution of Toeplitz systems*, IEEE Trans. Circuits and Systems, CAS-26 (1979), pp. 750–754.

[13] R. W. HAMMING, *Digital Filters*, Prentice-Hall, Englewood Cliffs, NJ, 1977.

[14] N. N. HSU AND S. C. HARDY, *Experiments in acoustic emission waveform analysis for characterization of AE sources, sensors, and structures*, in Elastic Waves and Non-destructive Testing of Materials, Y. H. Pao, ed., AMD 29, American Society of Mechanical Engineering, 1978.

[15] W. KAHAN AND B. N. PARLETT, *How far should you go with the Lanczos process?* in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 131–144.

[16] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standards, 45 (1950), pp. 255–281.

[17] C. LANCZOS, *Solution of systems of linear equations by minimized iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.

[18] S. LJUNG, *A fast algorithm to solve Fredholm integral equations of the first kind with stationary kernels*, Rep. LiTH-ISY-I-0265, Dept. Electrical Engin. Linköping University, Linköping, Sweden, 1979.

[19] C. B. MOLER AND G. W. STEWART, *An efficient matrix factorization for digital image processing*, (manuscript) Computer Science Department, University of Maryland, College Park, MD, 1978.

[20] C. C. PAIGE, *Bidiagonalization of matrices and solution of linear equations*, SIAM J. Numer. Anal., 11 (1974), pp. 197–208.

[21] ———, *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Maths. Applics., 10 (1972), pp. 373–381.

[22] ———, *Practical use of the symmetric Lanczos process with reorthogonalization*, BIT, 10 (1970), pp. 183–195.

[23] J. PALMER, *Conjugate direction methods and parallel computing*, Ph.D. Thesis, Stanford University, Stanford, CA, 1974.

[24] B. N. PARLETT AND D. S. SCOTT, *The Lanczos algorithm with selective reorthogonalization*, Math. Comp., 33 (1979), pp. 217–238.

[25] D. L. PHILLIPS, *A technique for the numerical solution of certain integral equations of the first kind*, J. Assoc. Comput. Mach. 9 (1962), pp. 84–97.

[26] M. A. SAUNDERS, *Sparse least squares by conjugate gradients: a comparison of preconditioning methods*, Rep. SOL 79-5, Stanford University, Systems Optimization Laboratory, Stanford, CA, 1979.

[27] G. W. STEWART, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[28] GRACE WAHBA, *Numerical and statistical methods for mildly, moderately, and severely ill-posed problems with noisy data*, Tech. Rep. 595, Dept. of Statistics, University of Wisconsin at Madison, 1980.

[29] J. H. WILKINSON, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.